Regular Paper

Scalability of X-NAS: A Clustered NAS System

Yoshiko Yasuda,† Shinichi Kawamoto,† Atsushi Ebata,†
Jun Okitsu,† Tatsuo Higuchi† and Naoki Hamanaka†

X-NAS (eXpandable network-attached storage), a scalable, clustered file system designed for entry-level NAS, has been developed. It enables centralized management of multiple NAS systems and virtualizes them into a single-file-system view for different kinds of clients. The core of X-NAS is a multi-protocol virtualized file system (MVFS), and its key feature — a file-handle cache — improves the system scalability while maintaining X-NAS manageability. To evaluate X-NAS scalability, an X-NAS prototype was designed and tested according to the NFSv2 implementation. These tests indicate that an eight-way X-NAS has a 10% faster response time and a 25% higher throughput than a conventional single NAS.

1. Introduction

Network-attached storage (NAS) is a network storage system——directly attached to an IP network——for efficiently managing digital data. NAS has recently been gaining general acceptance, because it can be managed easily and share files among many clients running different operating systems with different file systems. Among the various kinds of NAS, an entry-level NAS system is convenient in terms of cost and ease of management for offices and departments with no IT (information technology) experts.

However, an entry-level NAS is not scalable; therefore, if it becomes filled to capacity, clients must buy another one. This means that they then have to administer two NAS systems. Accordingly, the more NAS systems there are to be administered, the more administration costs will increase. To avoid such costs increases, it is effective to buy a midrange NAS, whose capacity can be expanded. However, the midrange NAS is much more expensive than an entry-level NAS system; therefore, clients in offices and departments cannot afford it.

To solve the above-described problems, a low-cost scalable NAS system must be developed. One approach is to make use of logical volume management (LVM), a well-known disk-expansion function. It can easily virtualize many disk drives as a unified drive by adding them to or removing them from an expansion PCI slot. In the case of the entry-level NAS with PCI connecters for adding new disk drives, clients can easily expand the system capacity at low cost. However, LVM can only be accomplished on only one NAS system. Once a part of the system, such as the CPU or operating system, suffers a fault, clients cannot access any files on the system.

Another approach is to use a clustered architecture, which virtualizes many file systems connected to the IP network as a single unified file system. Such a network-based clustered NAS system has a larger overhead than that of the LVM, because all NASs are managed via the IP network. However, the clustered NAS systems are promising in terms of reliability and performance because they have many CPUs and the network resources.

Under these circumstances, several scalable distributed file systems have been developed [1)~4)]. These scalable file systems virtualize many distributed file systems connected to the IP network as a single unified one. To specify a file system that stores file entities, they put additional information into the file handles (which are file identifiers). However, these systems can only be used under a UNIX environment and the file handles must be revalidated when the system is reconfigured.

X-NAS (where X stands for expandable) [5)~7)] is a scalable clustered NAS architecture designed for an entry-level NAS. It can be used for different kinds of clients, such as ones using UNIX or Windows. X-NAS is based on the following four goals.

- Cost reduction by using entry-level NAS as an X-NAS element
- Ease of use by providing a single-file-system view for different kinds of clients
- Ease of management by providing a centralized management function

† Hitachi Ltd., Central Research Laboratory

- Ease of scaling-up by providing several system-reconfiguration functions

To achieve the last three goals, X-NAS must manage multiple NAS systems without changing client environments, even file handles. In addition, to provide X-NAS for clients using an entry-level NAS, X-NAS has to maintain the manageability and performance of the entry-level NAS. As for manageability, centralized management is effective. However, centralized management tends to degrade performance as the system grows. To maintain the performance of the entry-level NAS and improve system scalability, a virtual file system with a low overhead must thus be developed.

To meet this demand, a virtualized file system, the core of X-NAS, has been developed. Its key feature reduces X-NAS overhead to improve system scalability while maintaining manageability. X-NAS scalability was evaluated by using an X-NAS prototype based on NFSv2. The evaluation results indicate that X-NAS incurs a lower overhead than a conventional entry-level NAS and has better cost performance and capacity scalability.

2. System Overview

Figure 1 shows an overview of X-NAS, which includes one P-NAS (parent NAS) node and many C-NAS (child NAS) nodes. A C-NAS node is equivalent to a single NAS system, which includes an NFS daemon [8] and a data partition. Each file system on the data partition has the same directories tree as that of the clients and is used to store file entities. P-NAS has two special functions: a multi-protocol virtualized file system, MVFS for short, and an X-NAS manager. MVFS is a global file system used for the virtualization and ease of management of many C-NAS nodes. It distributes each file entity among all data partitions. The X-NAS manager is responsible for X-NAS manageability features such as on-line reconfiguration, autonomous rebalancing, and automatic migration facilities. On-line reconfiguration enables administrators to add or remove X-NAS elements without stopping file-sharing services for clients. In addition, autonomous rebalancing can rebalance the available disk capacity by moving files between X-NAS members automatically and dynamically. Furthermore, automatic migration expands the capacity of the existing NAS node by keeping the existing files-and-directories tree. (Since the details of autonomous rebalancing and automatic migration are discussed in another paper [5),6)], they are not described here.) X-NAS also moves files from one data partition to the other partitions during X-NAS reconfiguration.

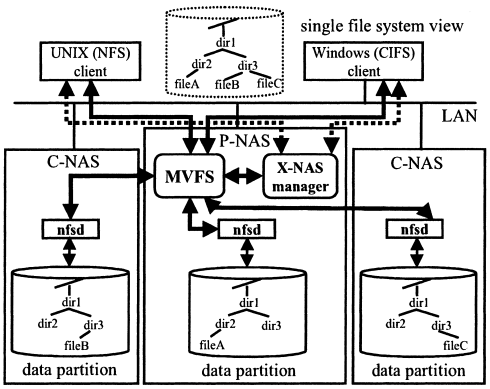

Fig. 1 X-NAS overview.

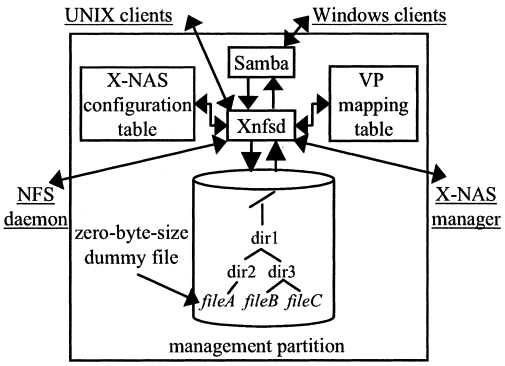

Fig. 2 Structure of MVFS.

3. MVFS

To reduce the administration cost of many NAS systems, a multi-protocol virtualized file system (MVFS) has been developed. MVFS is based on NFS and distributes all files, according to their inode numbers, to X-NAS members.

3.1 Structure

MVFS, an X-NAS core, enables the centralized management of many NAS nodes and provides a unified file system view for clients. **Figure 2** shows the structure of MVFS. It consists of Xnfsd, Samba [9], a management partition, an X-NAS configuration table, and a virtual-partition (VP) mapping table.

Xnfsd, the heart of MVFS, is a wrapper of the NFS daemon, which is the standard file system for UNIX. The wrapper daemon receives file-access requests from clients in place of the NFS daemon, and it then sends them to the appro-

priate NFS servers. Although Xnfsd emulates NFS, it has a very simple and smart structure; that is, it leaves almost all of the file-access processing to the original NFS servers. It can easily co-operate with Samba, a free software that provides seamless file services for Windows clients, because it is completely compatible with NFS.

The management partition provides a unified file system view for clients and is used to specify the C-NAS nodes that stores the file entities. The file system on the management partition keeps the same files-and-directories tree as that of the clients. However, all files on the management partition are zero-byte-size dummy files (Fig. 2). Dummy files are used for examining the attribute information in the files and directories. They are also used to specify the data partitions for storing the file entities.

The X-NAS configuration table keeps setting information such as host names of X-NAS members and their export points. The VP mapping table is used to specify the data partition that stores file entities. It keeps the correspondence between a virtual partition, which is a unit for managing files on X-NAS, and the data partition to which the virtual partition belongs. These tables are updated during X-NAS reconfiguration.

3.2 File-distribution Policy

Xnfsd distributes each file entity among all data partitions via the IP network. It therefore has to record the correspondence between each file and the data partition that keeps the file entity. One method is to record the correspondence in a table. However, this method poses a problem because the table size becomes huge as the number of files increases. As a result, such a huge table must be put on the low-speed disk drives not the high-speed memory. Furthermore, the search cost for such a huge table is high. To achieve an efficient search in the table on the disk, many subjects, which form the structure of the table and the search algorithm, should be considered.

Another method is to put additional information, which indicates the data partition that stores the file entity, into the file handles [1)~4)]. In this case, when file migration is caused by system reconfiguration, the identifier of the data partition in the file handles becomes invalid. The file handles must therefore be revalidated. However, revalidation needs modification of NFS client programs because the file handles are kept in client systems. To realize these revalidation processes, client programs must be modified. As the system grows, the management cost of modifying the client programs increases.

To solve the above-described problems, Xnfsd makes use of a file system on the management partition to record the correspondence. It distributes all files among all data partitions by using the inode numbers of dummy files that are managed by the file system on the management partition. Xnfsd determines a virtual file group, namely, a virtual partition to which a file belongs, by applying a hash function to the inode number of the dummy file. The virtual partition is a virtual unit for managing files and it is invisible to clients. The number of virtual partitions is fixed in advance. Managing all files as a unit of virtual partitions keeps the size of the mapping table compact. Xnfsd also records the correspondence between the virtual partition and the data partition in the VP mapping table.

The inode numbers of the dummy files are unique identifiers because these files are on a single file system of the management partition. When the inode number of the dummy file is $Inode_f$ and the number of the virtual partitions is N, the identifier of the virtual partition $V\text{-}ID_f$ of file f is given as follows:

$$V\text{-}ID_f = Inode_f mod N.$$

Since inode numbers of dummy files are managed by the file system on the management partition, they are not random numbers. However, the identifiers of the virtual partitions may be pseudo-random because they are calculated by applying a hash function to the inode numbers.

File-distribution using the inode number of the dummy file has two advantages. First, even if clients change the file name, the inode number of the file stays the same; file re-distribution is therefore not needed. Second, inode numbers are included in the original file handles. Since no additional information in file handles is needed, file revalidation is not needed.

3.3 Operations

NFS operations, which are emulated by Xnfsd, can be divided into four categories according to file-object type and process type as shown in **Table 1**: file objects are categorized as files or directories; processes are categorized as read or write. Category-1 and category-2 NFS operations are performed both on the management partition and on one of the

Table 1 Categories of NFS operations.

#	File object type	Process type	NFS operations (NFSv2)
1	File	Read	READ, STATFS, GETATTR, LOOKUP
2	File	Write	CREATE, WRITE, REMOVE, RENAME, SETATTR, LINK
3	Directory	Read	READDIR, READLINK, STATFS, GETATTR, LOOKUP
4	Directory	Write	MKDIR, RMDIR, REMOVE, SYMLINK, SETATTR

data partitions. Category-3 NFS operations are performed only on the management partition. Category-4 NFS operations are performed on the management partition and all data partitions, since all X-NAS members have the same directories tree.

(1) Category-1 NFS operations:
When a UNIX client sends a READ operation for file f to X-NAS, Xnfsd receives it in place of the NFS daemon. **Figure 3** shows the flow of this operation.
  (a) Xnfsd performs disk accesses to the management partition by using the file handle of the READ operation and then specifies the inode number of dummy file f.
  (b) Xnfsd calculates the virtual partition number by applying the hash function to the inode number, and then specifies the data partition that stores the file entity by referring to the VP mapping table.
  (c) Xnfsd traces disk blocks on the management partition by using the inode number of dummy file f and then gets the full path name of file f.
  (d) Xnfsd sends LOOKUP operations with the full path name of file f to the specified data partition, and it then gets a local file handle of file f on the data partition as the response to the LOOKUPs.
  (e) Xnfsd sends the READ operation to the specified data partition by using the local file handle, and then reads the entity of file f.

(2) Category-2 NFS operations:
Firstly, when a client sends a CREATE operation to make file f, Xnfsd creates dummy file f on the management partition. Secondly, it performs the above-described processes (b), (c), and (d). It then sends the CREATE operation to the specified data partition by using the local file handle and, finally, creates the entity of file f.

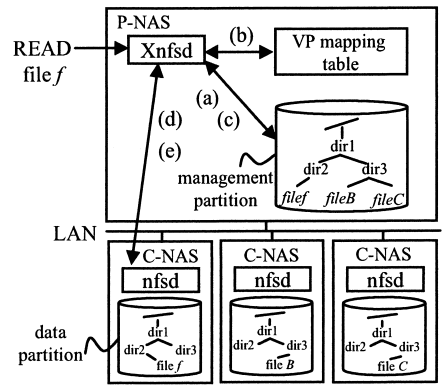

Fig. 3 Flow of READ operation.

(3) Category-3 NFS operations:
When a client sends a READDIR operation to read directory D, Xnfsd reads the entry list of directory D on the management partition.

(4) Category-4 NFS operations:
First, when a client sends a MKDIR operation to make directory D, Xnfsd creates directory D by sending the MKDIR operation to the management partition. Second, it performs the above-mentioned processes (c) and (d). In process (d), Xnfsd sends LOOKUP operations to all data partitions and then gets all of the local file handles from them. Third, Xnfsd sends the MKDIR operations with the local file handles to all data partitions. It then gets responses from all data partitions. Finally, it makes one response from all the responses and sends it back to the clients.

3.4 Interaction with X-NAS Manager

Xnfsd interacts with the X-NAS manager, which is responsible for X-NAS manageability features such as on-line reconfiguration. However, the on-line reconfiguration can add or remove NAS nodes easily and transparently without stopping the client file-sharing services.

The core architecture of on-line reconfiguration is two-layer mapping as shown in **Fig. 4**. The first-layer mapping correlates the dummy file on the management partition with the virtual partition. The second-layer mapping correlates the virtual partition with the data parti-

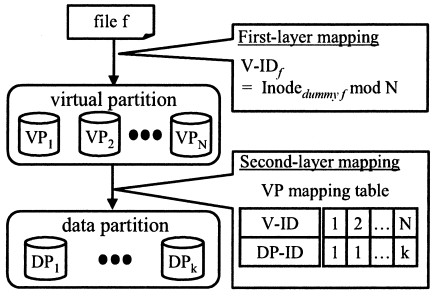

Fig. 4 File distribution by two-layer mapping.

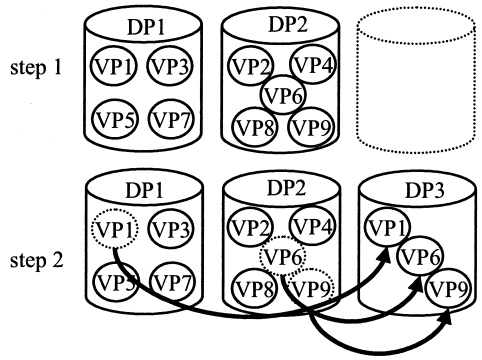

Fig. 5 Example of on-line reconfiguration.

tion. The relation between the virtual partition and the data partition is stored in a VP mapping table.

All files are handled as a unit of a virtual partition. The number of virtual partitions N is independent of that of data partitions (as described in Section 3.2). In terms of object balancing, it is useful to use a lot of virtual partitions in a few data partitions [10),11)]. In X-NAS, N is from approximately 100 to 1,000 times the number of data partitions. Firstly, each virtual partition is assigned at each data partition equally. Even if the X-NAS manager moves the virtual partition from one data partition to another by on-line reconfiguration, the correspondence between the file and the virtual partition to which the file belongs is the same. On-line reconfiguration thus simply involves updating the VP mapping table and the X-NAS configuration table.

The simplest method of file migration during on-line reconfiguration is performed by stopping the file-sharing services of clients on purpose (foreground migration). In addition to this migration method, X-NAS supports background migration. To achieve this, Xnfsd makes use of a retrying policy of NFS clients. It ignores a file-access request when the file object corresponding to this file-access request is moving between two data partitions. Since the NFS clients get no response to the file-access request, it resends this request until it receives the response.

Figure 5 shows an example of on-line reconfiguration. Firstly, the number of X-NAS members is only two, and all virtual partitions (in this case N is nine) are mapped to DP1 and DP2 (step 1). Secondly, X-NAS adds DP3 by on-line reconfiguration. When the X-NAS manager receives an add command from an administrator, it adds a new DP (DP3) to the X-NAS member. It then updates the VP mapping table and the X-NAS configuration table. After that, the X-NAS manager selects some virtual partitions (VP1, VP6, and VP9) on both DP1 and DP2 and moves them to DP3 (step 2). Although there are several algorithms for selecting which virtual partition to move, the current X-NAS applies the simplest way, that is, random selection [6)]. Since these processes are performed without disturbing the file-access requests from clients, the clients can continuously access their files on X-NAS.

4. Scalability

4.1 Goal

Although the X-NAS architecture has the capacity to be scaled up to a 64-NAS system, it was considered that the first X-NAS target market segment should be from entry-level to midrange NAS. These reasons for this choice are that midrange NAS is much more expensive than an entry-level NAS system and that the cost-and-performance gap between these two NAS systems is wide. To lower the cost compared with the expensive midrange NAS, X-NAS uses an entry-level NAS as an X-NAS element. To cover these market segments, X-NAS provides the capacity to scale up to several terabytes, i.e., equivalent to the total capacity of ten entry-level NAS systems. Furthermore, to provide the same manageability and performance of the entry-level NAS even if the system grows, X-NAS maintains the simple-and-unified management and the performance of the entry-level NAS.

4.2 Problems with MVFS

It is difficult to achieve compatibility between simple-and-unified management and performance scalability. To reduce the management cost of multiple NAS systems, MVFS manages file-access requests from clients and re-

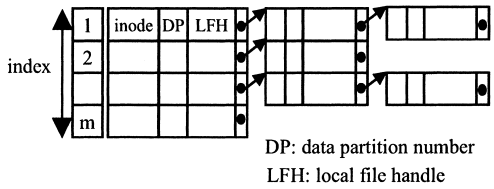

Fig. 6 Structure of file-handle cache.

configuration requests from the administrators only on P-NAS. This centralized management enables clients to use many NAS systems easily. On the other hand, a bottleneck on P-NAS tends to occur. Especially, MVFS uses existing metadata on the management partition and data partitions to reduce the X-NAS management data and to eliminate file-handle revalidation processing. However, the cost for specifying the data partition that stores the file entity is high. This is because processes such as specifying the full path name of the file and getting the local file handle of the file need many disk and network accesses (described in Section 3.3). To reduce the overhead and to improve X-NAS scalability, a file-handle cache was therefore developed.

4.3 File-handle Cache

The file-handle cache is a table that keeps the correspondence between the file handle of the dummy file, i.e., the global file handle, on the management partition and the local file handle on the data partition. It is generally kept in the memory on P-NAS. When Xnfsd sends LOOKUP operations and gets the local file handle as a response to the appropriate data partition, it registers the local file handle on the file-handle cache along with the inode number of the global file handle and the data partition number that keeps the file entity. **Figure 6** shows the structure of the file-handle cache. Xnfsd looks for the table by using the hash value of the inode number of the global file handle as an index. It then specifies the identifier of the data partition (DP) and the local file handle (LFH) corresponding to the global file handle.

NFS operations with the file-handle cache are performed as follows. When a client sends a READ operation of file f for the first time, Xnfsd performs processes (a), (b), (c), and (d). As a result, it registers the local file handle on the file-handle cache. After that, Xnfsd can eliminate processes (c) and (d) because it can reuse the local file handle on the memory of P-NAS.

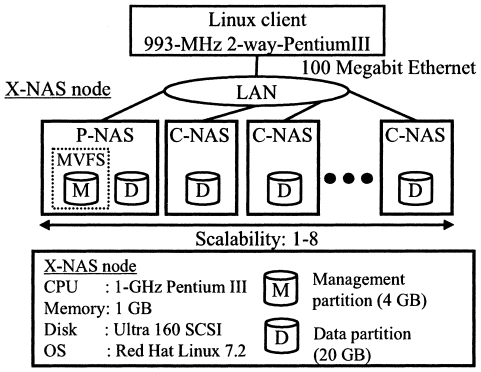

Fig. 7 Experimental environment.

The overhead incurred by accessing the management partition and the network can therefore be reduced.

5. Performance

To evaluate the system scalability, we measured the performance of MVFS by using an X-NAS prototype implemented on several 1U Linux servers. (Note that the X-NAS prototype is now based on the NFSv2 implementation.) Since all X-NAS functions were provided as a user-mode process on Linux and X-NAS was independent of the Linux kernel, it is portable.

To evaluate the overhead and scalability of X-NAS, the industry-standard SPECsfs97 benchmark program [12], which measures the performance of the NFS server, was run on the X-NAS prototype. The evaluation index SPECsfs97 was used as throughput, that is, the number of executed NFS operations per second when the same number of NFS operations is offered. Since implementation of the X-NAS prototype is based on NFSv2, an NFSv2 working set and the modification were used in the test. Note that the number of mount points was only one.

5.1 Experimental Environment

Figure 7 shows a schematic of the experimental environment, in which the maximum number of X-NAS elements is fixed to eight in our target segment. This is because the capacity of a disk will be increased before X-NAS can be expanded since the cost of disks is becoming much cheaper. Since a C-NAS only has one data partition, it is the equivalent to the original NFS server. On the other hand, P-NAS has a management partition in addition to a data partition. The client, C-NASs and P-NAS are connected by 100-Megabit Ethernet because most offices and departments that are

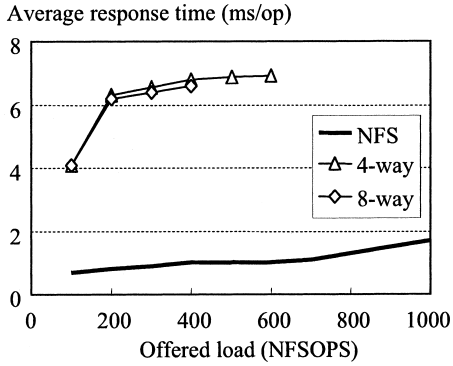

Fig. 8 Average response time of X-NAS without file-handle cache.

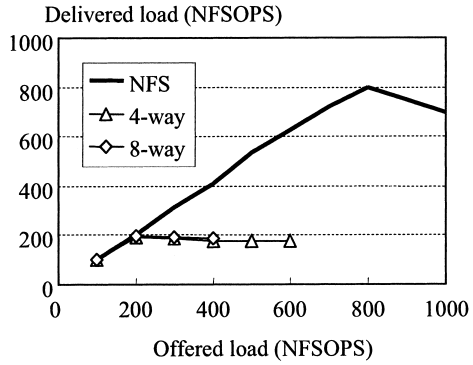

Fig. 9 Throughput of X-NAS without file-handle cache.

the targets of X-NAS still use this type of LAN.

5.2 Experimental Results

5.2.1 X-NAS without File-handle Cache

The evaluated performance, i.e., the relationship between the offered load and the average response time for all NFS operations, is shown in **Fig. 8**. The average response time of X-NAS is from five to eight times slower than that of the original NFS server. **Figure 9** shows the relationship between offered load and delivered load. The delivered load indicates whether the system can process all NAS operations of an offered load within a benchmarking time. If delivered load is equal to offered load, there is no bottleneck in the system. The maximum delivered load for the single NFS is 700 NFSOPS (Fig. 9). On the other hand, the throughput of the X-NAS prototype is one seventh compared to that of the single NFS server. Furthermore, the average response time and the throughput are the same in the case of both four- and eight-way X-NASs. These results suggest that X-NAS without a file-handle cache has no scalability (as described in Section 4.2).

It was assumed that the X-NAS without the file-handle cache has no scalability because the overhead to specify the data partition that stores the file entities is very high. To validate this assumption, relative response time for each NFS operation that appears in the SPECsfs97 benchmark program was determined.

Figure 10 shows the relative response time of NFS operations when the average response time for each NFS operation on a single NFS server is equal to one. Clearly, a longer relative response time means a higher overhead. The figure shows that the average response times for directory-access requests such as READDIR and READLINK (category-3) in X-NAS are the same as those for requests in the single NFS server. On the other hand, the average response times for file-access requests such as LOOKUP and CREATE (category-1 and category-2) are from about three to twenty-five times slower than those for these requests in the single NFS. Furthermore, the average response times for directory-access requests such as MKDIR and RMDIR (category-4) are much slower than those for requests in the single NFS. Because of these overheads, a bottleneck occurs on P-NAS. As a result, the X-NAS system without the file-handle cache has no scalability and much lower performance than the single NFS server.

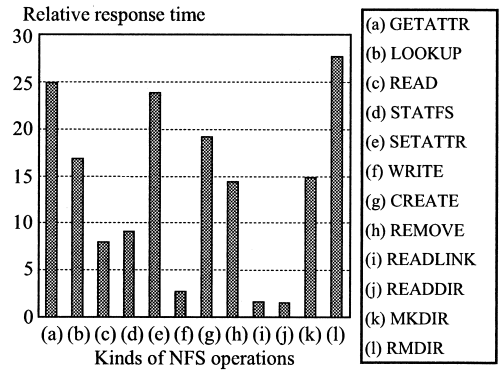

Fig. 10 Relative response time for NFS operations in the case of eight-way X-NAS.

5.2.2 Effects of File-handle Cache

Figure 11 shows the average response time for the eight-way X-NAS with a file-handle cache. The total throughput is shown in **Fig. 12**. The average response time and the throughput were measured by changing the number of file-handle cache entries. As shown in Fig. 11, as the number of the file-handle cache

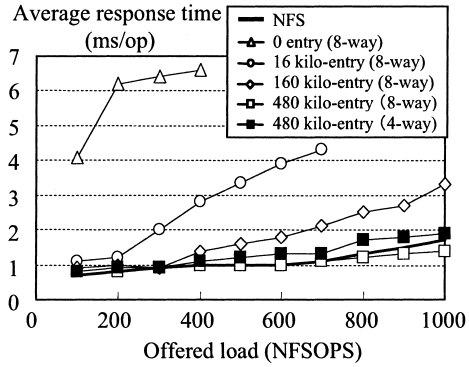

Fig. 11 Average response time of X-NAS with file-handle cache.

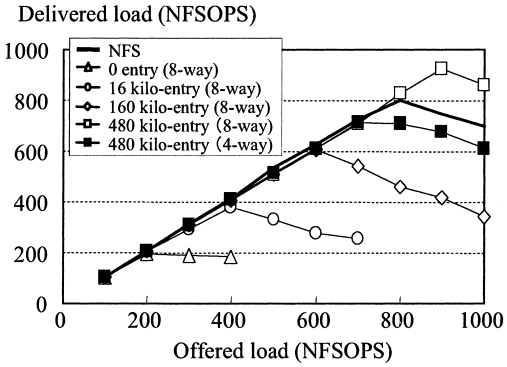

Fig. 12 Throughput of X-NAS with file-handle cache.

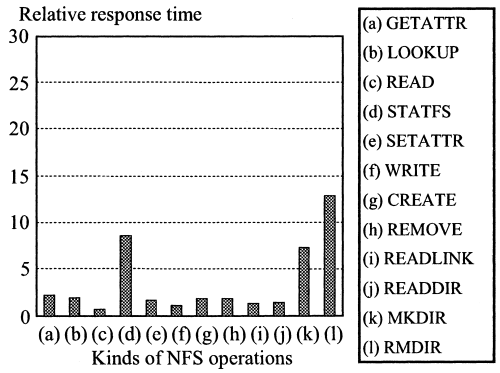

Fig. 13 Relative response time for NFS operations in the case of eight-way X-NAS.

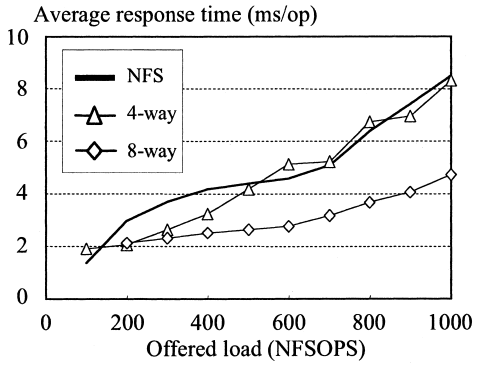

Fig. 14 Average response time for READ operations.

entries increases, the average response time becomes quicker. When the file-handle cache has 480 kilo-entries, the average response time for the eight-way X-NAS is 10% faster than that for the single NFS server. Furthermore, the throughput of eight-way X-NAS is 25% higher than that of the single NFS (Fig. 12). **Figure 13** shows the relative response time of NFS operations when the file-handle cache has 480 kilo-entries. Clearly, almost all relative response times are much faster than those shown in Fig. 10.

When the offered load is 1,000 NFSOPS, the delivered load is sustained (i.e., load is saturated) even though the number of the cache entries increases. On the other hand, in the case of four-way X-NAS, the maximum delivered load is 11% lower than that of the single NFS even though the number of cache entries increases.

6. Discussion

To investigate the above-mentioned load saturation in the case of four- and eight-way X-NASs with a file-handle cache, the average response times for READ operation in the case that the file-handle cache has 480 kilo-entries were analyzed (**Fig. 14**). This is because that the total response time for READ operations occupies about half of the benchmarking time. The average response times for READ operations of the four-way X-NAS dramatically increase when the offered load is 800 NFSOPS. This result indicates that a bottleneck in the disk accesses on P-NAS occurs. If the offered loads on both the four- and eight-way X-NASs are the same, the number of disk accesses on one data partition of the four-way X-NAS is double, that on the eight-way X-NAS. In addition, P-NAS has to process the disk access on the management partition. This means that the response time for READ operations on the four-way X-NAS becomes much slower than that on the eight-way X-NAS at the same offered load.

The SPEC client issues NFS operations at constant intervals determined by the offered load. It also issues a new NFS operation after receiving the result of the previous operation. However, even if the response time for NFS operations became slower because of such a con-

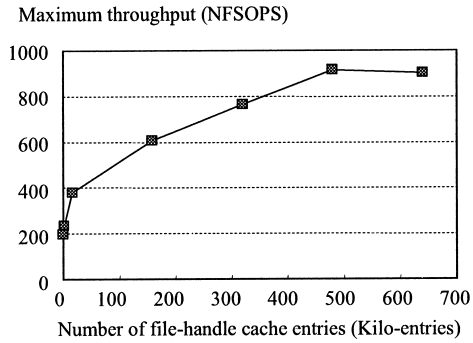

Fig. 15 Relation between maximum throughput and file-handle cache entries in the case of eight-way X-NAS.

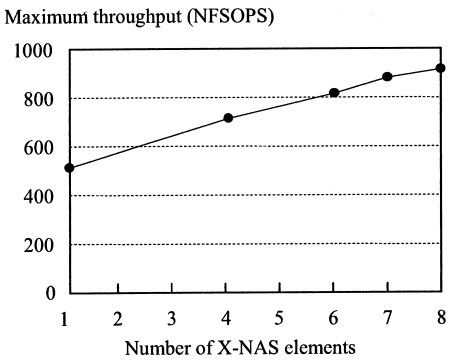

Fig. 16 Maximum throughput of X-NAS.

centration of disk accesses, the delivered load is ensured as long as the response delays are within the allowance range of the offered load. In the case of four-way X-NAS, when the offered load is 800 NFSOPS, the response delays becomes out of range. As a result, the response time for READ operations increases dramatically, and then the delivered load is saturated. However, if P-NAS had no data partition, this saturation might be eliminated.

On the other hand, the average response time for READ operations on eight-way X-NAS gets gradually longer as the load increases (shown in Fig. 14). It can be said that in the case of eight-way X-NAS, the bottleneck is not caused by disk access concentration. The transfer rate of the network was calculated by using the workload mix and the average file size to be accessed on the SPECsfs97 benchmark program. As a result, it was found that the bottleneck is caused by low network transfer rate. (Note that the evaluation environment consisted of a 100-Megabit Ethernet as the network.) However, if the Gigabit Ethernet is used, the network bottleneck can be reduced.

The relation between the maximum throughput and the number of the file-handle cache entries in the case of eight-way X-NAS is shown in **Fig. 15**. When the file-handle cache has about 480 kilo-entries, the maximum throughput is sustained. This means that the number of file-handle cache entries is sufficient on the SPECsfs97 benchmark program. According to our previous research, the average file size of office documents for an entry-level NAS is about from 100 to 300 kilobytes [6]. Since the average file size to be accessed on this benchmark program is much smaller than that, this number of file-handle cache entries is sufficient in practical use.

Figure 16 shows the scalability of X-NAS: the more the number of X-NAS elements, the higher the X-NAS throughput. The results of the performance evaluation described above show that X-NAS provides a quicker response time and a higher throughput than a single NFS server. This result verifies that X-NAS has better cost performance and capacity scalability while maintaining the performance and the manageability of entry-level NAS.

7. Related Work

There have been several studies on scalable distributed file systems based on NFS [1)~4)]. In particular, DiFFS [2),3)] and Slice [4)] resemble X-NAS in terms of design principles. DiFFS delivers high performance because it is distributed among many partitions. It puts additional information, i.e., the partition ID, into a file handle. Slice, developed at Duke University, is also close to X-NAS in terms of its partition approach and use of directory servers. The partition used in Slice is a single server. Slice also puts a file ID as a routing key in each file handle, so it can be easily reconfigured. Furthermore, the NAS switch [1)] uses the same approach as DiFFS and Slice in that it puts an identifier of NAS, which stores the file entity, into each file handle. File handles must therefore be revalidated in the case of on-line reconfiguration. All the above-described systems need modification of client programs in order to support file-handle revalidation processes.

X-NAS is a scalable clustered architecture based on a standard NFS file system. It can provide file-sharing services for many kinds of client OS. To specify the file location, it uses the inode number instead of additional information in each file handle. In the case of sys-

tem reconfiguration, revalidation of file handles is therefore not needed. Furthermore, its file-distribution method is completely invisible to a client's environment. The file handle cache therefore enables locations of files, which are frequently used, to be kept in the memory. As a result, the overhead incurred when specifying the file locations can be significantly reduced and X-NAS scalability can be improved.

8. Future Work

When the number of NAS elements is more than ten, the performance of the current X-NAS is not sufficient, because the file-access requests on the current X-NAS are managed by one P-NAS. To solve this problem, hardware and software improvements to deal with the network bottleneck should be considered. One hardware improvement is to use a Gigabit Ethernet and a software one is to change the transmitting policy, which directly returns the responses of the file-access requests from the data partitions to the client. As for Gigabit Ethernet, parts of many 100-Megabit LANs are being upgraded to Gigabit Ethernet systems. The benefit of Gigabit Ethernet will be manifested when all of such networks (such as routers, switching hubs and Ethernet) are replaced. Generally, 100-Megabit Ethernet is still the mainstream technology in the offices and departments that are the X-NAS targets. We thus that the evaluation of X-NAS using Gigabit Ethernet is an important future work.

If the network bottleneck were eliminated, a CPU bottleneck would occur. However, the CPU usage is less than 30% on the current P-NAS when the number of X-NAS members is eight. The CPU bottleneck can thus be reduced by raising the CPU frequency and by using an on-chip multiprocessor to deal with the CPU bottleneck. Moreover, using many NASs as the entry points for file-access requests is also effective. However, in terms of the ease of management and ease of use of the multiple NAS systems, to handle multiple entry points is hard for entry-level NAS users. Current X-NAS therefore manages file-access requests and administration requests on one entry point. The authors thus consider that some virtualization methods of multiple entry points must be developed in the future.

Regarding the reliability of X-NAS, a single point of failure is also a problem. However, it can be solved by X-NAS clustering, details of which will be discussed in another paper.

9. Conclusions

X-NAS, a scalable clustered NAS designed for an entry-level NAS, has been developed. The X-NAS core, a so-called multiple-protocol virtualized file system (MVFS), enables centralized management of many NAS nodes and provides a single file-system view for clients running different operating systems.

Xnfsd, which is the heart of MVFS, is a smart-code wrapper daemon that completely emulates the NFS daemon and distributes all file-access requests into many NFS servers. Its file-distribution policy based on inode numbers is useful for renaming of files and directories as well as for system reconfiguration. The file-handle cache, which keeps the correspondence between the global file handle on the management partition and the local file handle on each data partition, can improve the X-NAS cost-performance and scalability.

An X-NAS prototype, an eight-way clustered system based on NFSv2 running the SPECsfs97 benchmark program has a 10% faster response time and 25% higher throughput than a single NFS server. In other words, X-NAS not only improves the performance of the single NAS system but also provides a unified-file-system view for both UNIX and Windows clients.

Acknowledgments We would like to thank the reviewers for their helpful comments and suggestions.

References

1) Yamakawa, S., et al.: NAS Switch: NFS Server Virtualization, Technical ReportCPSY2002-36, The Institute of electronics, Information and Communication Engineers (2002).
2) Karamanolis, C., et al.: DiFFS: a Scalable Distributed File System, Technical Report HPL-2001-19, HP Laboratories Palo Alto (2001).
3) Karamanolis, C., et al.: An Architecture for Scalable and Manageable File Services, Technical Report HPL-2001-173, HP Laboratories Palo Alto (2001).
4) Anderson, D.C., et al.: Interposed Request Routing for Scalable Network Storage, *ACM Trans. Comput. Syst.*, Vol.20, No.1 (2002).
5) Yasuda, Y., et al.: Concept and Evaluation of X-NAS: a Highly Scalable NAS System, *Proc. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies* (*MSST 2003*) (2003).
6) Kawamoto, S., et al.: Expandable NAS (X-

NAS) and Its Autonomic File Redistribution Policy, Technical Report 14th Data Engneering Workshop (DEWS2003), The Institute of electronics, Information and Communication Engineers (2003).
7) Yasuda, Y., et al.: Scalability of X-NAS: A Clustered NAS System, *Proc. Symposium on Advanced Computing Systems and Infrastructures* (*SASCIS2003*) (2003).
8) Callaghan, B.: *NFS Illustrated*, Addison Wesley, Reading, Massachusetts (2000).
9) Eckstein, R., et al.: *Using Samba*, O'Reilly and Associates, Inc (1999).
10) Litwin, W., et al.: LH*: a scalable, distributed data structure, *ACM Transactions on Database Systems*, Vol.21, No.4 (1996).
11) Honicky, R.J. and Miller, E.L.: An optimal Algorithm for Online Reorganization of Replicatd Data, Technical ReportUCSC-CRL-02-36, University of California, Santa Cruz (2002).
12) Standard Performance Evaluation Corporation: *SFS3.0 Documentation Version 1.0*, http://www.spec.org (2002).

(Received January 23, 2003)
(Accepted May 15, 2003)

Yoshiko Yasuda received her B.E. degree from Waseda University in 1991. In 1991, she joined Hitachi Ltd., Central Research Laboratory, where she designed the inter-network architecture for Hitachi's SR2201 parallel computer. She is currently involved in research and development on clustered network-attached storage systems. She is a member of the IPSJ and IEEE-CS.

Shinichi Kawamoto received his B.E., M.E. and Ph.D. degrees from Tohoku University in 1991, 1993 and 1998, respectively. From 1996 to 1998, he was a research associate at Tohoku University. In 1998, he joined Hitachi Ltd., Central Research Laboratory. He is currently involved in research and development on network-attached storage systems. He is a member of IPSJ.

Atsushi Ebata received his B.E. and M.E degrees from the University of Tsukuba in 1993 and 1995. He joined Hitachi Ltd., Central Research Laboratory in 1995, where he helped develop mainframe computers. He is currently involved in research and development on clustered network-attached storage systems.

Jun Okitsu received his B.E. and M.E. degrees from Univ. of Tokyo Institute of Technology in 1999 and 2001. He has been working in the Platform System Research Department of Hitachi Central Research Laboratory since 2001. He is currently involved in research and development on clustered network-attached storage systems.

Tatsuo Higuchi received his B.E. and M.E. degrees from The University of Tokyo, Japan in 1988 and 1990, respectively. He has been working in Central Research Laboratory, Hitachi Ltd., since 1990 and is now a senior researcher. His current research interests are distributed and parallel computer architectures and clustered network-attached storage systems. He is a member of IEICE.

Naoki Hamanaka received B.S. and M.S. degrees from The University of Tokyo in 1983 and 1985. He has been working in Central Research Laboratory, Hitachi Ltd., since 1985. He is currently a department manager in the Platform System Research Department. He is a member of IPSJ and IEICE.