Regular Paper

RX-NAS: A Scalable, Reliable Clustered NAS System

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

RX-NAS (Replicated eXpandable Network-Attached Storage), a scalable, reliable clustered NAS system designed for entry-level NAS, has been developed. RX-NAS is based on X-NAS, which is a simple clustered NAS architecture for entry-level NAS, and improves the reliability of X-NAS by adding new sets of X-NASs to the original one. The core feature of RX-NAS, namely on-line replication, replicates original file objects to new sets of X-NASs for each file block in real-time without changing clients' environments. RX-NAS has other key features for maintaining the manageability of entry-level NAS; namely, new synchronization and resynchronization functions can easily replicate original files and directories to other X-NAS systems completely or partially without changing clients' environments. In addition, its health-check function can eliminate a limitation on the configuration of RX-NAS and detect and report errors that occur in the RX-NAS system. To validate the RX-NAS concept, an RX-NAS prototype was designed and tested according to the NFSv3 implementation. These tests show that the RX-NAS improves system reliability while maintaining 80% of the throughput of X-NAS.

1. Introduction

Network-attached storage (NAS) is a network storage system—directly connected 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, entry-level NAS is convenient in terms of cost and ease of management for offices and departments with no IT (information technology) experts.

However, entry-level NAS has two problems: one is that it is not scalable; the other is that it is not reliable enough. To solve the first problem, we have already proposed X-NAS [1)~7)], a simple, scalable clustered NAS architecture designed for entry-level NAS. It can be used for various kinds of clients, such as those using UNIX and Windows☆, like conventional NAS systems. X-NAS is based on the following four goals.

- Cost reduction by using entry-level NAS as one of X-NAS elements
- Ease of use by providing a single-file-system view for various kinds of clients
- Ease of management by providing a centralized management function
- Ease of scaling-up by providing several system-reconfiguration functions

To achieve these goals, X-NAS virtualizes multiple entry-level NAS systems as a unified system without changing clients' environments. In addition, to provide X-NAS for clients using entry-level NAS, X-NAS maintains the manageability and the performance of entry-level NAS [1),2)]. It can also be easily reconfigured without stopping file services or changing setting information.

As for the second problem, to improve system reliability, a file replication method is effective. Generally an entry-level NAS is composed of two to four hard-disk drives and has a software-based RAID 5 configuration [8)]. Since all data are maintained in the RAID 5 system, users can continue to access data even if one of the disk drives suffers a fault. However, this redundancy is achieved in one NAS system. Therefore, all data may be lost when the OS or the controller of the NAS suffers a fault. For that reason, the user of the entry-level NAS generally generates replicas of data on another system by using an attached backup software in addition to the redundancy provided by the RAID 5 configuration.

However, the backup software generates an archive from many files when it makes replicas. Users cannot therefore access replicas easily when they want to, because it takes a long

† Hitachi, Ltd., Central Research Laboratory

☆ Windows is a trademark of Microsoft Corporation. All other products are trademarks of their respective corporations.

time to restore the archive. Furthermore, these replicas cannot be generated in real-time. The newest files and directories on a faulty NAS system, may thus be lost when one of the X-NAS elements suffers a fault. To improve the reliability of an entry-level NAS, a file-replication function must therefore be developed.

Using a clustering software is one solution to improve system reliability and availability [9]. Generally, clustering software has a high-speed file-replication function in addition to a fail-over function. Because of these many functions, this software is about several times more expensive than entry-level NAS. It is therefore unsuitable for users of entry-level NAS: accordingly, they want another simple solution to improve system reliability.

Using a file-replication software is one such solution. It replicates file objects on the master NAS to remote NAS systems via an IP network, for entry-level up to midrange NAS [10)~13)]. However, such kinds of software are aimed at a single NAS system. Furthermore, they must use their own protocols to replicate file objects to remote NAS systems.

The goal of the present work was to introduce a new concept called RX-NAS (where R stands for replicated and X stands for expandable) in order to improve the reliability of entry-level NAS as well as to improve its scalability. RX-NAS can replicate original file objects on X-NAS to other sets of X-NASs connected to an IP network in real-time for each file I/O. It can also manage multiple X-NASs while maintaining the manageability of entry-level NAS.

In this study, the RX-NAS concept was validated by using a RX-NAS prototype based on an NFSv3 implementation. The evaluation results show that RX-NAS incurs 80% of the throughput of X-NAS and improves the reliability of an entry-level NAS while maintaining its manageability.

2. X-NAS

2.1 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 [14)] and a data partition. Each file system on the data partition is used to store file entities. It has the same directories tree as the file systems of clients. P-NAS has two special functions: a multi-protocol

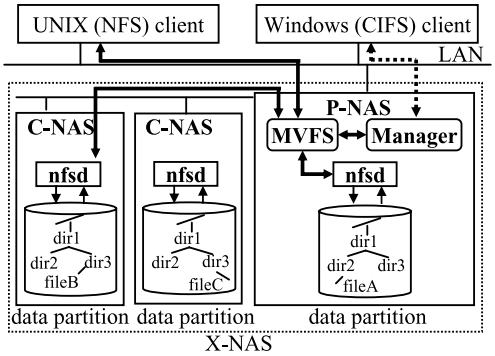

Fig. 1 X-NAS overview.

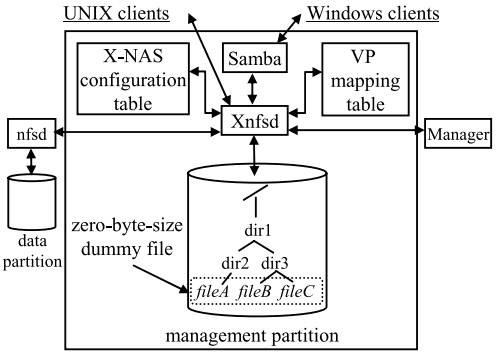

Fig. 2 Structure of MVFS.

virtualized file system, MVFS for short, and a manager. MVFS virtualizes many data partitions as a unified one and distributes each file entity to one of the data partitions. The manager is responsible for manageability features such as on-line reconfiguration, autonomous rebalancing, and automatic migration facilities (Section 2.3).

2.2 MVFS

Figure 2 shows the structure of MVFS. It consists of Xnfsd, Samba [15)], the management partition, an X-NAS configuration table, and a virtual-partition (VP) mapping table. Xnfsd is a wrapper of the NFS daemon and sends file-access requests to the NFS daemon on the P-NAS and C-NASs. The management partition is used to specify the data partitions that store the file entities. The file system on the management partition keeps the same files-and-directories tree as that of clients. However, all files on the management partition are zero-byte-size dummy files. These dummy files are used for examining the attributes of files and directories. They are also used to specify data partitions to store the file entities. The X-NAS configuration table keeps setting informa-

ID of virtual partition	1	2	3	4	…	N
ID of data partition	1	1	2	2	…	k

N: Number of virtual partitions
k: Number of data partitions

Fig. 3 Structure of VP mapping table.

tion such as hostnames of X-NAS elements and their export points. The VP mapping table is used to specify the data partition that stores files entities. These tables are updated during X-NAS reconfiguration.

2.2.1 File-distribution Policy

Xnfsd distributes all files among all data partitions by using the inode numbers of dummy files. Xnfsd determines a virtual partition by applying a hash function to the inode number of the dummy file. The virtual partition is a virtual unit for managing files and is invisible to clients. The number of virtual partitions is fixed in advance. In terms of object balancing, using a lot of virtual partitions in a few data partitions is useful [16),17)]. In X-NAS, this number is from approximately 100 to 1,000 times the number of data partitions. 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:

$$V - ID_f = Inode_f \ mod \ N.$$

All virtual partitions are correlated with the data partition. The relation between the virtual partition and the data partition is kept in a VP mapping table (**Fig. 3**). The VP mapping table is referred to in order to specify the data partition that keeps the file entity. It is updated when the X-NAS configuration is changed (described in Section 2.3).

2.2.2 Operations

NFS operations are divided into four categories: two for file object type (i.e., files and directories) and two for process type (read and write) as shown in **Table 1**. NFS operations are performed as follows:

(1) Category-1 and -2 NFS operations
NFS operations belonging to categories 1 and 2 are performed on the management partition and one of the data partitions. When a client sends a WRITE operation for file f, Xnfsd receives the WRITE operation in place of the NFS daemon. **Figure 4** shows the flow of this operation.

Table 1 Categories of NFS operations.

	File object type	Process type	NFS operations (NFSv3)
1	File	Read	READ, STATFS, GETATTR, LOOKUP,ACCESS
2	File	Write	CREATE, WRITE, REMOVE, RENAME, SETATTR, LINK, MKNOD, COMMIT
3	Directory	Read	READDIR, READLINK, STATFS, GETATTR, FSINFO, LOOKUP, READDIRPLUS, ACCESS
4	Directory	Write	MKDIR, RMDIR, SYMLINK, SETATTR, REMOVE, RENAME

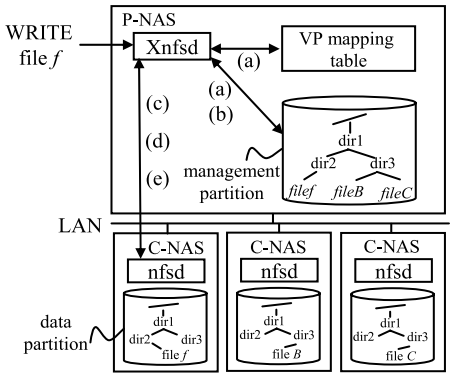

Fig. 4 Flow of WRITE operation.

(a) Xnfsd specifies the data partition that stores the file entity by using the inode number of dummy file f.

(b) Xnfsd specifies the full path name by tracing the inode number of dummy file f and the disk blocks.

(c) Xnfsd sends LOOKUP operations with the full path name to the appropriate data partition.

(d) As the response to the LOOKUP operations, Xnfsd gets the local file handle of file f on the data partition.

(e) Xnfsd then sends the WRITE operation for file f to the data partition by using this local file handle. Finally, after Xnfsd gets a response, it sends it back to the client.

(2) Category-3 NFS operations
NFS operations belonging to category 3 are performed on only the management partition. When a client sends LOOKUP operations for accessing directory to X-NAS, Xnfsd receives the operations in

place of the NFS daemon. It reads the attribute information about the directory from the file system on the management partition, and returns its response to the client.

(3) Category-4 NFS operations
NFS operations belonging to category 4 are performed on both the management partition and all data partitions. When a client sends a MKDIR operation to X-NAS, Xnfsd receives the MKDIR operation in place of the NFS daemon. First, Xnfsd creates a directory on the management partition. Next, Xnfsd gets the local file handles from all data partitions by using LOOKUP operations and the full path name. Xnfsd then sends the MKDIR operation with the local file handles to all data partitions. Finally, when Xnfsd gets the responses from all NFS servers, it makes one response from all the responses and sends it back to the clients.

2.3 Key Features

The more NAS systems there are to be administered, the more administration costs will increase. To maintain the manageability of a single NAS while providing a high level of scalability, X-NAS provides three key features: on-line reconfiguration, autonomous rebalancing, and automatic migration. These features enable X-NAS reconfiguration to be easily performed without affecting clients' environments.

2.3.1 On-line Reconfiguration

On-line reconfiguration is a strong feature of X-NAS. It can add or remove NAS nodes easily and transparently without stopping client file-sharing services. When one of the X-NAS elements is unstable, the administrator removes it by using a Web browser. After that, X-NAS automatically moves all accessible files from the unstable NAS node to the other nodes. Namely, files are moved for each virtual partition.

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 supporting foreground migration, 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 client gets no response to the file-access request, it continues to resend this request until it receives the response. As a result, file migration during on-line reconfiguration can be performed in the background, namely, without disturbing the file-access requests from clients. Note that the relationship between file-access frequency and file moving by on-line reconfiguration in the background is described in Ref. 4).

2.3.2 Autonomous Rebalancing

According to the inode-based file-distribution policy, the number of files may be equally distributed among data partitions. However, each data partition does not use the same amount of disk space, because each file has a different size. Moreover, X-NAS can be used in a heterogeneous environment because the capacity of data partition may change with shipping year and cost. On-line reconfiguration, i.e., adding or deleting X-NAS elements, also incurs a capacity unbalance. Such a capacity unbalance among X-NAS elements may cause a concentration of file-access requests from clients. In response to these circumstances, if the available capacity size of each data partition becomes unbalanced, an autonomous rebalancing facility moves files among data partitions automatically and dynamically without the need for any client administration.

The manager checks the available capacity size of each data partition at constant intervals. It moves some files in this data partition when the available capacity size in any data partition is lower than the threshold chosen in advance. Since files are managed as a unit of virtual partition, it selects some virtual partitions randomly on the full data partition according to the VP mapping table. It then moves files in these virtual partitions to the other data partitions by NFS operations such as READ and WRITE. After moving them, the manager updates the VP mapping table. Since these processes are performed without disturbing the file-access requests from clients, the clients can continuously access their files on X-NAS.

2.3.3 Automatic Migration

Conventional entry-level NAS is not scalable, and it has no MVFS or on-line reconfiguration functions. Clients with existing entry-level NAS systems cannot therefore expand their systems. However, by using the automatic migration function of X-NAS, clients can expand the capacity of the existing NAS while keeping the files-and-directories tree on the existing NAS

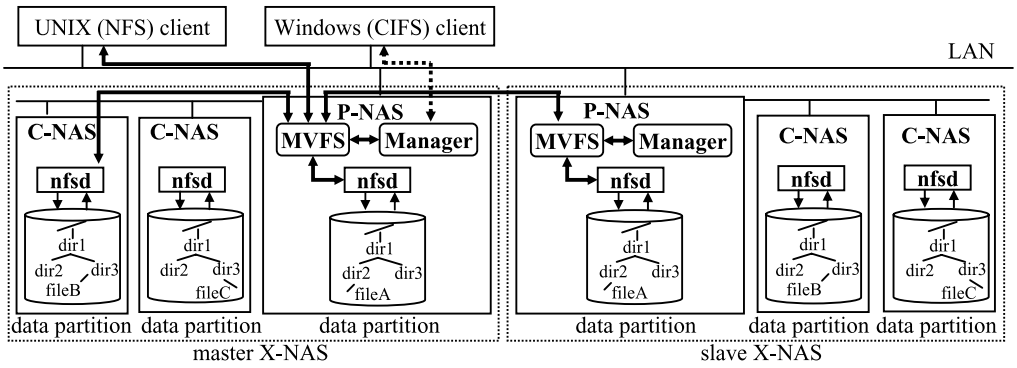

Fig. 5 RX-NAS overview.

system.

When the manager receives an automatic migration request from clients, it stops file-sharing services on the existing NAS system and mounts the file system of the existing NAS on X-NAS. After that, X-NAS traces the files-and-directories tree on the existing NAS. It then copies this tree to the file system on the management partition. Just after this copying process, the VP mapping table indicates that all virtual partitions belong to the existing NAS. After X-NAS makes the files-and-directories tree on the management partition, it restarts the file-sharing service for clients and moves some virtual partitions corresponding to the existing NAS to another data partition in the background. For example, when using a capacity-rebalancing policy, X-NAS moves the virtual partitions to balance capacity availability among all data partitions.

3. Expansion to RX-NAS

When one of the X-NAS elements suffers a fault, all file objects on the faulty NAS may be lost unless an administrator makes backups of the file objects manually. A file-replication method must therefore be developed.

There are several methods for replicating file objects to other systems via an IP network. One method is to use a block I/O [11]. Since a block I/O such as an SCSI command is fine grain, all file objects are completely consistent with their replicas. On the other hand, the system structure is limited because the logical disk blocks of the objects must be allocated to the same address between the original data and its replica. A second method is to change the client's system. DFS [10] is a simple method for replicating file objects to many NASs. It replicates files at constant intervals not in real-time. Another method is to install third-vendor software in the client systems. In this case, however, as the number of clients increases, the management cost increases.

Generally, it is difficult to apply the conventional methods to a clustered NAS system such as X-NAS. This is because all meta-data of file objects and their entities are independently managed for virtualization. For this reason, we make use of Xnfsd in order to replicate file objects on the X-NAS to remote X-NASs. In place of the NFS server, Xnfsd can receive an NFS operation and send the operation to others as described above. By extending this function, Xnfsd on the X-NAS sends the NFS operation not only to the NFS servers on the original X-NAS, which is called the master X-NAS, but also to the NFS servers on the remote X-NAS (slave X-NAS). All file objects can thus be replicated in real-time for each NFS operation.

3.1 RX-NAS Overview

Figure 5 shows an overview of RX-NAS, which includes two X-NAS nodes: one is a master X-NAS and the other is a slave X-NAS. The slave X-NAS is a replica of the master X-NAS. It has the same file-and-directories tree as that of the master X-NAS. Administrators can add a slave X-NAS via an IP network as they need. Since the slave X-NAS is a complete replica of the master X-NAS, administrators can use the slave X-NAS and users can access their files immediately when the master X-NAS suffers a fault.

Each MVFS of the X-NASs distributes each file entity into one of the data partitions that are members of each X-NAS. In addition, MVFS of the master X-NAS sends file-access requests, whose process types are "write", to the MVFS of the slave systems, which is called an on-line replication. The manager on the mas-

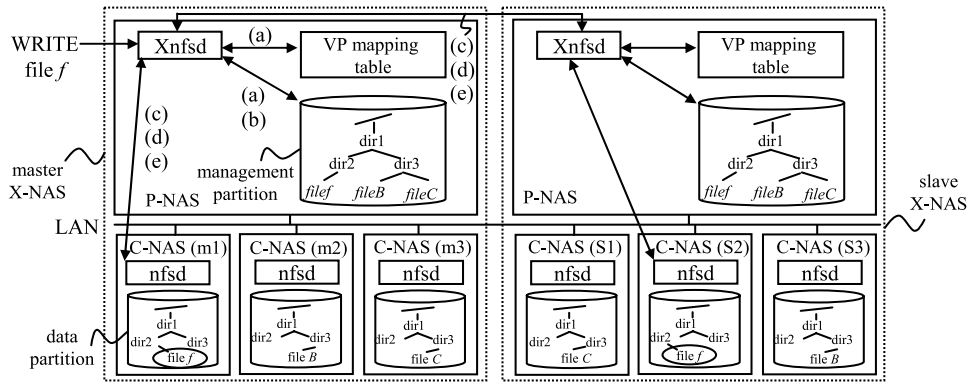

Fig. 6 Flow of WRITE operation on RX-NAS.

ter X-NAS is responsible for dependability features besides the X-NAS management features described in Section 2.3, namely, RX-NAS reconfiguration and a Health-check function. RX-NAS reconfiguration replicates file objects on the master X-NAS completely or partially to the salve X-NAS. Health-check detects the failures in the RX-NAS and reports them to the manager. It also automatically stops performing the on-line replication when Xnfsd and the manager detect an error in the slave X-NAS and/or a network.

The file system on the management partition of the master X-NAS is exported to the P-NAS and clients. However, the management partition of the slave X-NAS is exported to only the P-NAS of the master X-NAS for security. The X-NAS configuration table is extended in order to store information about the slave X-NAS, such as hostname and export point, as well as information about the X-NAS elements of the master X-NAS.

3.2 On-line Replication

Xnfsd on the master X-NAS sends NFS operations to the slave Xnfsds in place of the NFS servers on the slave systems directly. Each Xnfsd on the slave X-NASs can therefore virtualize multiple C-NASs as a unified slave X-NAS. By leaving virtualization to each slave X-NAS, each slave X-NAS can function as the master system immediately in the case that the master X-NAS suffers a fault.

Xnfsd on the master X-NAS (master Xnfsd) sends NFS operations belonging to categories 2 and 4 not only to the data partitions on the master X-NAS but also to Xnfsds on the slave X-NASs (slave Xnfsds). On the other hand, as mentioned in the previous section (Section 2.2.2), NFS operations belonging to categories 1 and 3 are performed as well. This means that only write operations are sent to the slave X-NASs.

(1) Category-2 NFS operations
When a UNIX client sends a WRITE operation for file f to RX-NAS, the master Xnfsd receives the operation in place of the NFS daemon. **Figure 6** shows the flow of this operation which consists of the following steps.
  (a) Xnfsd specifies the data partition that stores the file entity by using the inode number of dummy file f.
  (b) Xnfsd specifies the full path name by tracing the inode number of dummy file f and the disk blocks.
  (c) Xnfsd sends LOOKUP operations with the full path name not only to the appropriate data partition but also to the management partitions on the slave X-NASs.
  (d) Xnfsd gets the local file handle of file f on the data partition of the master X-NAS, and it gets the global file handle of dummy file f on the management partitions of the slave X-NASs as the response to the LOOKUP operations.
  (e) Xnfsd then sends the WRITE operation to both the NFS server and slave Xnfsds at the same time by using the local file handle and the global file handle. Finally, after Xnfsd receives all the responses from the NFS server and the slave Xnfsds, it makes one response and sends it back to the client.

(2) Category-4 NFS operations
NFS operations belonging to category 4

are performed on all partitions on the RX-NAS. When a client sends a MKDIR operation to RX-NAS, Xnfsd on the master X-NAS receives the MKDIR operation in place of the NFS daemon. First, Xnfsd creates a directory on the management partition of the master X-NAS. Next, Xnfsd gets the local file handles from all data partitions and the global file handles from management partitions on the slave X-NASs by using LOOKUP operations. Xnfsd then sends the MKDIR operation with the local file handles to all NFS servers and all other Xnfsds. Finally, when Xnfsd gets the responses from all NFS servers and the slave Xnfsds, it makes one response from all the responses and sends it back to the clients.

3.2.1 Key Features

RX-NAS replicates file objects on the master X-NAS to the slave X-NASs in real-time for each file I/O. It must therefore guarantee that the data of the file objects on the master X-NAS is consistent with that on the slave X-NASs. To achieve this consistency, Xnfsd on the master X-NAS waits for all the responses from both the NFS server of the master X-NAS and Xnfsd of the slave X-NASs for eachNFS operation. However, this waiting degrades performance. To solve this problem, the performance of the on-line replication function must be improved through two key features as follows.

(1) Multi-threaded wrapper daemon
To guarantee the consistency of data among the master X-NAS and the slave X-NASs, Xnfsd on the master X-NAS waits for all responses from these systems. This waiting incurs an overhead because of frequent accesses to the network and the disk drives. To reduce this cost, the main thread of the master Xnfsd invokes sub threads in order to send the file-access requests to the slave X-NASs. This feature enables RX-NAS to process the disk accesses in all X-NASs in parallel.

(2) Replication file-handle cache
In the case of RX-NAS, the master X-NAS sends the file-access requests to the slave X-NASs. As a consequence of this process, the cost of specifying the file handle of the dummy file, i.e., the global file handle on the slave X-NASs, becomes high. To reduce this cost, a replication file-handle cache has been developed. The replication cache works only on the master X-NAS. The structure of the replication cache is the same as that of the file-handle cache for X-NAS[2] shown in **Fig. 7**. However, the replication cache saves different information from that of the file-handle cache for X-NAS; that is, it saves the identifiers of the slave X-NASs instead of the data partition number and the global file handle instead of the local file handle. Since the lifetime of the entries saved in the replication cache is longer than that of the file-handle cache, these separated cache structures are effective.

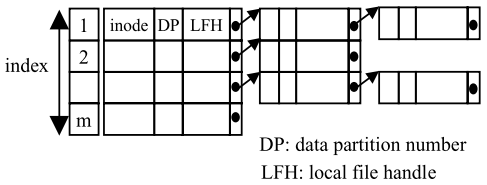

Fig. 7 Structure of file-handle cache for X-NAS.

4. Dependability

To maintain the manageability of entry-level NAS while providing more reliability, the manager has two other key features besides the X-NAS reconfiguration features described in Section 2.3: RX-NAS reconfiguration and a health-check function. These features enable RX-NAS reconfiguration and management to function easily without affecting clients' environments.

4.1 RX-NAS Reconfiguration

RX-NAS reconfiguration consists of two functions. One is "new synchronization" and the other is "resynchronization". New synchronization can improve the system reliability by adding new sets of X-NASs to the existing X-NAS and by replicating all file objects on the existing X-NAS to the new ones. Resynchronization is used to resume on-line replication. To do this, it checks the consistency of the file objects among RX-NAS members. It then replicates or removes the file objects to or from the slave X-NASs in the case that the file systems of the X-NASs are different.

4.1.1 New Synchronization

Firstly, when an administrator wants to configure RX-NAS, he provides the NEWSYNC command, a hostname, and an export point of the new (slave) X-NAS by using a Web browser. Secondly, the manager of the master X-NAS

mounts the file system of the slave X-NAS and registers it as a replicating target. Thirdly, the manager traces the files-and-directories tree on the management partition of the master X-NAS. Finally, the manager copies the files-and-directories tree to the slave X-NAS. This copying process is performed by using READ operations for the master X-NAS and WRITE operations for the slave one. Generally, after processing WRITE operations, the execution completion times are set on the slave X-NAS as the file attributes. As a result, the file attributes on the master X-NAS are different from those on the slave X-NAS. Accordingly, the consistency of all data among X-NASs is not guaranteed. To solve this problem, the manager issues SETATTR operations to the slave X-NAS after executing the WRITE operations. These SETATTR operations can set the generation times for all files of the slave X-NAS at the same generation times on the master X-NAS.

4.1.2 Resynchronization

When the manager of the master X-NAS receives the RESYNC command from an administrator, it then traces the files-and-directories tree on the management partition of the master X-NAS and those of the slave X-NASs. It checks the consistency of the file attributes between the master X-NAS and the slave X-NASs. The manager gets the attribute information of the slave X-NASs by issuing the LOOKUP operations for the file objects. If there is an inconsistency among the file objects of these systems, the manager removes an old one on the slave X-NASs and creates a new one on the slave X-NASs. This enables the consistency of all files-and-directories trees between X-NASs to be guaranteed. Accordingly, on-line replication can be resumed.

4.2 Health-check Functions

RX-NAS has a health-check function that performs error detecting and error reporting between the master X-NAS and the slave X-NAS. It also has an automatic-disconnection function, which automatically stops performing the on-line replication when Xnfsd and the manager detect errors.

Figure 8 shows the software structure of the health-check functions. The UI agent receives administration commands such as new synchronization and resynchronization. The manager consists of the main thread that executes the commands from the administrator, the checker, and the HB (heartbeat) function. The health check includes the checker and the HB function. The checkers of the master X-NAS and the slave X-NAS manage the disk capacity and the disk usage. Furthermore, they perform the error detection and the error reporting by using STATFS procedures between the master X-NAS and the slave X-NAS. The HB functions monitor the connectivity between the master X-NAS and the slave X-NAS. They check the heartbeat requests and the heartbeat response at constant intervals.

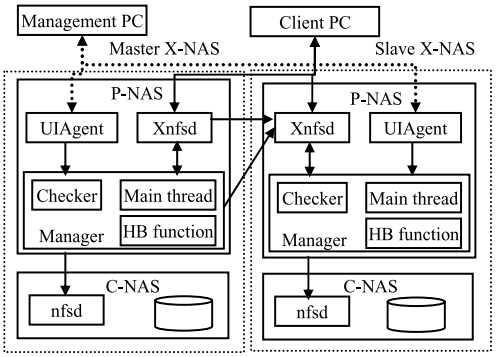

Fig. 8 Software structure of health check.

	RX-NAS	master X-NAS	slave X-NAS
Total disk capacity	150 GB	300 GB	150 GB
Used disk capacity	20 GB	20 GB	20 GB

Fig. 9 Structure of RX-NAS capacity table.

4.2.1 Checker

The checker uses the RX-NAS capacity table to handle the disk capacity of each X-NAS (**Fig. 9**). The table keeps the total capacity and used capacity of each X-NAS. It also stores the total and used disk capacities of RX-NAS, that is, the smallest ones of the X-NASs. To get the disk-usage ratio of each X-NAS, the manager on the master X-NAS sends FSSTAT operations both to all C-NAS nodes on the master X-NAS and to the slave Xnfsds at constant intervals. It then gets all the responses from them and updates the table. When an administrator sends a request to RX-NAS to get the disk capacity of RX-NAS, the manager returns the total capacity of the RX-NAS according to the RX-NAS capacity table. When a client intends to write a file over the total capacity of RX-NAS, the manager detects that the client cannot write the file by referencing the RX-NAS capacity table, and it returns an error to the client.

Table 2 Details of errors detected by master and slave X-NASs.

Error	Meaning
PRPCError	Network failure or data partition failure in master X-NAS
PServerError	Server error in master X-NAS
SRPCError	Network failure between master and slave X-NAS or the failure in slave X-NAS
SServerError	Server error in slave X-NAS
NoHB	Failure in master X-NAS

The checker also detects errors that occurred between the master X-NAS and the slave X-NAS. **Table 2** lists the errors that can be detected by the checker and the HB function. These errors are managed by the flags. If the checker detects an RPC error in the master X-NAS, it sets the PRPCError to TRUE. In the case of an NFS server error such as *NFS3ERR_IO*, which means a hardware error occurs during processing the NFS request, the checker sets the PServerError to TRUE. It then records the error to the log file and reports to the main thread of the manager and stops the Xnfsd process. (In the case that the checker fails, the HB function of the slave X-NAS detects the error.) If the checker detects an RPC error in the slave X-NAS, it sets the SRPCError to TRUE. In the case of an NFS server error, it sets the SServerError to TRUE. The checker then records the error to the log file and reports to the main thread of the manager.

If the checker of the slave X-NAS detects an RPC error, it records the error to the log file and reports the main thread of the manager and stops the Xnfsd process. In the case of an NFS server error, it records the error to the log file and reports the main thread of the manager and stops the Xnfsd daemon.

4.2.2 Heartbeat Function

The heartbeat (HB) function monitors the connectivity between the master X-NAS and the slave X-NAS. The master X-NAS issues a NULL procedure at constant intervals to the slave X-NAS. When the HB function of the master X-NAS gets the response to the NULL procedure, it sleeps at constant intervals and repeats this procedure after these intervals. The HB function of the master X-NAS judges that the slave X-NAS may suffer a fault when it gets no responses to the NULL procedures at multiple times. After detecting an error, the HB function sets the SRPCError flag (Table 2) to TRUE and records the error to the log file. And it reports to the main thread of the manager that the slave X-NAS may be fault.

On the other hand, the HB function of the slave X-NAS monitors the NULL request at constant intervals. When the Xnfsd of the slave X-NAS receives the NULL procedure from the HB function of the master X-NAS, it checks whether the procedure is from the master X-NAS or not. If the NULL procedure comes from the master X-NAS, the HB function of the slave X-NAS records the arrival time of the procedure and returns the response to the sender. The HB function of the slave X-NAS monitors the arrival time of the NULL procedure at constant intervals. It compares the difference between the arrival time and the monitoring time. If there is no request from the HB function of the master X-NAS at multiple monitoring time, the HB function of the slave X-NAS judges that the master X-NAS may be fault. It changes the NoHB flag to TRUE and records the error to the log file and reports to the main thread of the manager that the master X-NAS may be fault.

4.2.3 Automatic Disconnection

Since RX-NAS is connected with an IP network, it can suffer a LAN failure or an intra-LAN failure. If RX-NAS continues to replicate file objects while there are some troubles in the RX-NAS environment, these troubles will probably be propagated to the normal X-NAS systems. When Xnfsd on the master X-NAS (master Xnfsd) detects an error by processing NFS procedures or performing the health check, it automatically stops performing the on-line replication by using the automatic-disconnection function.

In the following example, it is assumed that a LAN failure is incurred after a client sends a CREATE operation to the master X-NAS while performing on-line replication. The master Xnfsd sends the CREATE operation to the specified data partition on the master X-NAS and to the slave X-NAS at the same time. However, in the case of a LAN failure, the master X-NAS gets no response to the CREATE operation for the slave X-NASs, which is called TIMEOUT. When the master Xnfsd detects TIMEOUT, it judges that RX-NAS has suffered a fault and it then stops performing the on-line replication. After that, the master Xnfsd does not send anymore write operations to the slave X-NAS until it has finished performing new synchronization or resynchronization.

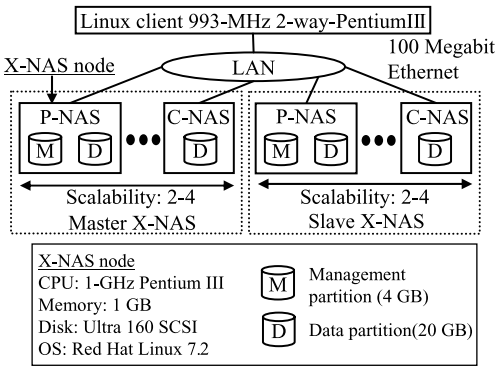

Fig. 10 Experimental environment of RX-NAS.

In the case that the master X-NAS suffers a fault, the manager on the slave X-NAS detects the error and sends an e-mail alert to the administrator. The administrator receives the e-mail alert and then changes the settings to make use of one slave X-NAS as an alternative.

5. Performance

To validate the RX-NAS concept, we measured the performance of RX-NAS by using a RX-NAS prototype implemented on several 1U Linux servers. All RX-NAS functions were provided as a user-mode process on Linux.

To evaluate the overhead of RX-NAS, the industry-standard SPECsfs97 benchmark program [18], which measures the performance of the NFS server, was run on the RX-NAS prototype. The evaluation indexes of SPECsfs97 are throughput and response time. The throughput is the number of NFS operations per second when the same number of NFS operation is offered. Since implementation of the RX-NAS prototype is based on NFSv3, an NFSv3 working set was used in the test. Note that the number of mount points was only one.

5.1 Experimental Environment

Figure 10 shows a schematic of the experimental environment, in which three kinds of RX-NASs, namely, RX-NAS (P1, C1), RX-NAS (P1, C2), and RX-NAS (P1, C3), were used to evaluate the RX-NAS performance. Each RX-NAS has two sets of X-NAS, whose elements are changed from two to four. For example, RX-NAS (P1, C2) means that it has two sets of X-NAS, which are a master X-NAS and a slave X-NAS, and each X-NAS consists of three elements, i.e., one P-NAS and two C-NASs. In the experimental environment, Xnfsd was multi-threaded for replication and the file-handle cache had sufficient entries [2].

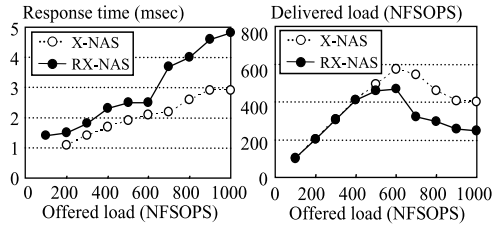

Fig. 11 Average response time and throughput in the case of RX-NAS (P1, C1).

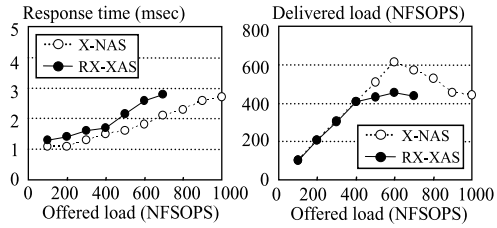

Fig. 12 Average response time and throughput in the case of RX-NAS (P1, C2).

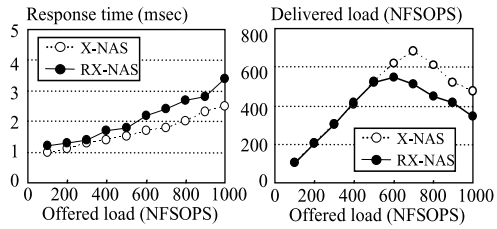

Fig. 13 Average response time and throughput in the case of RX-NAS (P1, C3).

5.2 Experimental Results

Figures 11, **12** and **13** show the average response of all NFS operations and the relationship between offered load and delivered load. In RX-NAS configurations, the files-and-directories trees on the slave X-NAS were completely consistent with those on the master X-NAS. In the case of RX-NAS (P1, C2), however, the checker on the master X-NAS detected the disk I/O error, which was occurred by the access to bad blocks on the data partition of the slave X-NAS, and stopped performing on-line replication when the offered load reached 800 NFSOPS. Therefore, the average response time and throughput of RX-NAS (P1, C2) are not plotted when the offered load exceeded 800 NFSOPS. The average response time for RX-NAS is from 1.25 to 1.39 slower than that for X-NAS. The larger the number of the X-NAS elements is, the smaller the performance gap between X-NAS and RX-NAS is. The throughput for RX-NAS is an 80% of that of X-NAS.

Figure 14 shows the average response time for X-NAS and RX-NAS in the case of changing

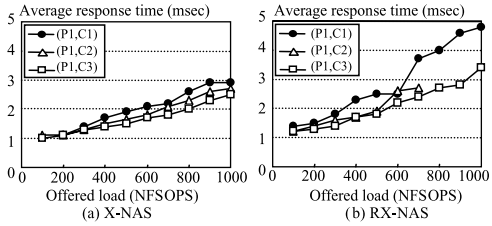

Fig. 14 Average response time of X-NAS and RX-NAS.

the number of X-NAS elements. The gap between the average response times for X-NASs is 0.4 milliseconds, while the gap for RX-NASs is twice that for X-NASs. Accordingly, RX-NAS is more sensitive to performance than X-NAS for a given number of X-NAS elements.

5.3 Discussion

To investigate the reason that the gap between RX-NAS and X-NAS is larger in the case of a lower number of the elements and the gap becomes small as the number of elements increases, the response times for READ and WRITE operations were analyzed. These operations were chosen because that the processing time for READ is about 40% of the total testing time, and that for WRITE is 30% of the total testing time. **Figure 15** shows the average response time for WRITE operations, and **Fig. 16** shows that for READ operations in the case of X-NAS and RX-NAS. In the case of WRITE operations in X-NAS, the average response time for each X-NAS is the same (Fig. 15 (a)). On the other hand, the average response time for WRITE operations in the case of RX-NAS is different for the three kinds of RX-NAS. The performance gap between RX-NAS (P1, C1) and RX-NAS (P1, C3) is 40%. Furthermore, the response time for RX-NAS (P1, C3) is twice that for X-NAS (P1, C3). This is because the master X-NAS of RX-NAS sends the WRITE operations to the slave X-NAS and waits for the response from the slave X-NAS.

In the case of READ operations, there is a 40% performance gap between X-NAS (P1, C1) and X-NAS (P1, C3) (Fig. 16 (a)). The performance gap for RX-NAS becomes larger than that for X-NAS and the response time for RX-NAS (P1, C1) is 1.5 times longer than that for X-NAS (P1, C1) (Fig. 16 (b)). However, when the number of the X-NAS elements is four, the response times for X-NAS and RX-NAS are the almost same.

The above results are explained as follows. In

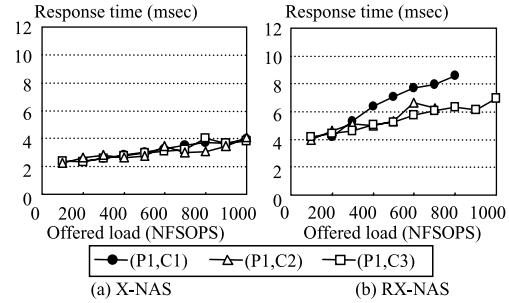

Fig. 15 Average response time for WRITE operations in the case of X-NAS and RX-NAS.

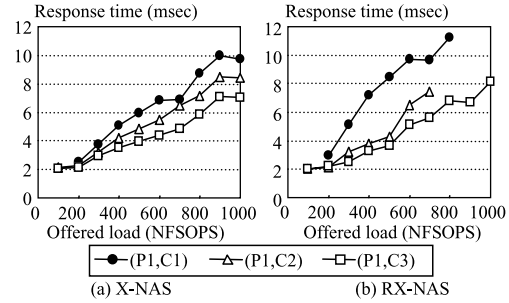

Fig. 16 Average response time for READ operations in the case of X-NAS and RX-NAS.

regards to the SPECsfs97 benchmark, a client issues two outstanding requests for READ operations and two for WRITE. In the case of RX-NAS (P1,C1), the disk-access time for the data partitions is longer than that in the other cases because the requests are distributed to only two data partitions. Moreover, the later outstanding READ operation must wait until the completion of the previous disk-access request because the disk access is a sequential access. The response time for the READ therefore becomes longer. As the number of the X-NAS members increases, the disk-access time for data partitions becomes shorter because disk distribution widens. Consequently, the waiting time for the following outstanding READ operation is reduced and the response time is shortened.

The results of the performance evaluation described above demonstrate that the performance gap between the RX-NAS and the X-NAS is large because the response time for READ operation dominates the total performance when the number of X-NAS elements is small. On the other hand, the performance gap can be reduced because the responses times for READ operations of X-NAS and RX-NAS are the same in the case of a larger number of X-NAS elements.

6. Related Work

Several scalable distributed file systems based on NFS [19)~22)] have been developed, but they have no file-replication function. On the other hand, there are several file-replication methods for replicating file objects between several NAS systems via an IP network. For example, DFS [10)] is a simple and easy file-replication function on Windows systems. It replicates files and folders on one Windows system at constant intervals to other systems by using DFS clients and DFS servers. DRBD [11),12)] is a kernel module for building a two-node HA cluster under Linux. It replicates files and directories for each block I/O by using a proprietary protocol. Because of the block-I/O, the files and directories on the slave server are updated in real time. In addition, Double Take [13)] is a third-vendor software for replicating file objects on the master NAS to the slave NAS. It also uses its own protocol to communicate between the master and slave systems.

RX-NAS is a scalable and reliable NAS architecture based on a standard NFS. It can replicate file objects on the master X-NAS to the slave X-NASs in real-time for each NFS operation without changing clients' environments. It can also use a general NFS server as the slave NAS because it sends general NFS operations. Furthermore, RX-NAS has several management functions to improve its manageability. These features can improve the reliability and dependability of entry-level NAS.

7. Conclusions

A new concept named RX-NAS, a scalable and reliable clustered NAS system, has been developed. RX-NAS is based on X-NAS—a simple, clustered NAS architecture for entry-level NAS—and it improves the reliability of X-NAS by adding new sets of X-NASs to the original one. On-line replication, which is the core function of RX-NAS, replicates file objects on the master X-NAS to slave X-NASs in real-time for each NFS operation. A multi-threaded wrapper daemon and the replication file-handle cache, which keeps the correspondence between the global file handle of the master X-NAS and that of the slave ones, reduce the overhead for accessing remote X-NASs.

To improve the manageability of RX-NAS, administrators can easily start or restart on-line replication by using the RX-NAS's new-synchronization and resynchronization features without changing clients' environments. A health-check function can perform error detection and error reporting when X-NAS and/or a network suffer a fault. Automatic disconnection can automatically stop performing the on-line replication when the health-check function and/or Xnfsd detect an error. Furthermore, the function guarantees the total disk-capacity size of RX-NAS for writing file objects while performing on-line replication.

An RX-NAS prototype, which is based on an NFSv3 implementation running the SPECsfs97 program, attains 80% of the performance of X-NAS. It enables the reliability of entry-level NAS to be improved while maintaining its manageability.

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

References

1) 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).
2) Yasuda, Y., et al.: Scalability of X-NAS: a Clustered NAS System, *IPSJ Transactions on Advanced Computing Systems*, Vol.44, No.SIG11 (2003).
3) 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).
4) Kawamoto, S., et al.: Autonomic File Rebalancing Policy of X-NAS: a Clustered NAS System, Technical Report of IEICE CPSY2003-6 (2003).
5) Yasuda, Y., et al.: Disk Quota Method Designed for a Clustered NAS System (X-NAS), *IPSJ Transactions on Advanced Computing Systems*, Vol.45, No.SIG11 (2004).
6) Yasuda, Y., et al.: An On-line Backup Function of X-NAS: a Clustered NAS System, *Proc. 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies* (*MSST 2004*) (2004).
7) Ebata, A., et al.: Consideration of Synchronized Backup of X-NAS: a Clustered NAS System, *Proc. FIT 2003* (2003).
8) Patterson, D.A., et al.: A Case for Redundant Arrays of Inexpensive Disks (RAID), *Proc. SIGMOD'88* (1988).

9) Vogels, W., et al.: The Design and Architecture of the Mircrosoft Cluster Service, *Proc. FTCS'98* (1998).
10) Microsoft Corporation: *Deploying Windows Powered NAS Using Dfs with or Without Active Directory* (2001). http://www.microsoft.com
11) Reisner, P.: DRBD, *Proc. 7th International Linux Kongress* (2000).
12) Reisner, P.: DRBD Distributed Replicated Block Device, *Proc. 9th Linux Kongress* (2002).
13) NSI software: *Double-Take Theory of Operations* (2001). http://www.nsisoftware.com
14) Callaghan, B.: *NFS Illustrated*, Addison Wesley, Reading, Massachusetts (2000).
15) Eckstein, R., et al.: *Using Samba*, O'Reilly and Associates, Inc (1999).
16) Litwin, W., et al.: LH*: a scalable, distributed data structure, *ACM Transactions on Database Systems*, Vol.21, No.4 (1996).
17) Honicky, R.J. and Miller, E.L.: An optimal Algorithm for Online Reorganization of Replicatd Data, *Proc.17th International Parallel and Distributed Processing Symposium* (2003).
18) Standard Performance Evaluation Corporation: *SFS3.0 Documentation Version 1.0* (2002). http://www.spec.org
19) Karamanolis, C., et al.: DiFFS: a Scalable Distributed File System, Technical Report HPL-2001-19, HP Laboratories Palo Alto (2001).
20) Karamanolis, C., et al.: An Architecture for Scalable and Manageable File Services, Technical Report HPL-2001-173, HP Laboratories Palo Alto (2001).
21) Anderson, D.C., et al.: Interposed Request Routing for Scalable Network Storage, *ACM Transactions on Computer System*, Vol.20, No.1 (2002).
22) Yamakawa, S., et al.: NAS Switch: NFS Server Virtualization, Technical Report CPSY2002-36, The Institute of electronics, Information and Communication Engineers (2002).

(Received November 4, 2003)
(Accepted October 4, 2004)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.1, pp.1–14.)

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 SR2201 parallel computer. She is currently involved in research and development for clustered network attached storage systems and I/O architecture of blade server systems. She is a member of the IPSJ and IEEE.

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 and now is a senior researcher. He is currently involved in research and development for high available application platform. 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 developed the mainframe computer. He is currently involved in research and development for network attached storage systems.

Jun Okitsu received his B.E. and M.E. degrees from the University 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 for enterprise blade server 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 now is a senior researcher. His current research interests are architecture of distributed and parallel computer architecture 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.