Yao J, Shu JW, Zheng WM. Distributed storage cluster design for remote mirroring based on storage area network. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 22(4): 521~526 July 2007

Distributed Storage Cluster Design for Remote Mirroring Based on Storage Area Network

Jun Yao (姚 骏)[1,2], Ji-Wu Shu (舒继武)[1], and Wei-Min Zheng (郑纬民)[1]

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*

[2] *Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*

E-mail: shujw@tsinghua.edu.cn

Received June 29, 2006; revised April 20, 2007.

Abstract With the explosion of information nowadays, applying data storage safety requirements has become a new challenge, especially in high data available cluster environments. With the emergence of Storage Area Networks (SANs), storage can be network-based and consolidated, and mass data movements via Fiber Channels (FCs) can be of very high speed. Based on these features, this paper introduces a dual-node storage cluster designed for remote mirroring as a concurrent data replication method to protect data during system failures. This design takes full advantage of a SAN system's benefits, and it adopts a synchronous protocol to guarantee a fully up-to-date data copy on the remote site. By developing a Linux kernel module to control the I/O flow and by using the technologies of software Logic Unit Number (LUN) masking, background online resynchronization and a self-management daemon, we have achieved a reliable mirroring system with the characteristics of server-free data replication, fault tolerance, online disaster recovery and high performance. In this study, we implemented the design in a remote mirror subsystem built on a software Fiber Channel Storage Area Network (FC-SAN) system.

Keywords remote mirror, SAN (storage area network), server free, fault tolerance, online disaster recovery

1 Introduction

Mass data storage has become one of the greatest and most urgent challenges in the development of networks because of a sharp increase in data storage needs. Network storage devices can provide both information access and data sharing. As a new storage architecture, Storage Area Network (SAN), emerges, SAN-based[1~3] techniques provide solutions for information integration and data sharing, as well as easy manageability and high security. Because storage area networks are based on a network-oriented structure and ensure complete separation between data storage and computing, they have many strong points such as flexible addressing ability, long-distance transmissibility, high I/O speed and high sharing ability. However, as network storage research has progressed, safe data storage has become the new focus and will certainly become a major factor in evaluating a network system's availability. Since SANs have consolidated storage, it is possible to utilize this feature to build a system that is tolerant to great disasters.

In this paper, we introduce a new design for a remote mirroring method based on SAN's consolidated storage by adopting a dual-node storage cluster. The motivation of the design is to make an up-to-date *Data Mirroring*, so as to enhance the availability of the storage system. Unlike other data replicating products like hardware controlled IBM PPRC and XRC[4], or host software driver based Veritas Volume Replicator[5], it is a fully server-free synchronous software remote mirroring method. We take full advantages of FC-SAN's consolidated storage feature, and the mirroring process is completely transparent to the hosts. The software modules on the dual-node cluster can take a full control of the I/O streams, so that when hardware failure happens, it can easily isolate bad disks, or degrade to a one-node I/O system, and continue data storage services with the degraded system. Together with these disaster tolerant mechanisms, we also include a background online resynchronization mechanism after the system is rebuilt.

The paper is organized as follows. We first introduce our software FC-SAN system with the name TH-MSNS (Tsinghua Mass Storage Network System)[6~8] in Section 2. Based on the TH-MSNS system, Sections 3 and 4 introduce the design of the dual-node mirroring cluster and the key techniques in implementing the software controlled mirroring method. The performance analysis of synchronous mirroring can be found in Section 5 and Section 6 concludes the paper.

2 Architecture of TH-MSNS

The remote mirror designed on the TH-MSNS[6] was based on self-owned patent technologies. Fig.1 demonstrates the basic system's architecture, which is a typical FC-SAN design.

As shown in Fig.1(a), the system's hardware architecture includes devices with different functions: multiple host nodes, an I/O node, FC devices that include FC

Short Paper

This work is supported by the National Natural Science Foundation of China under Grant No. 10576018, the National Grand Fundamental Research 973 Program of China under Grant No. 2004CB318205, and the Program for New Century Excellent Talents in University.

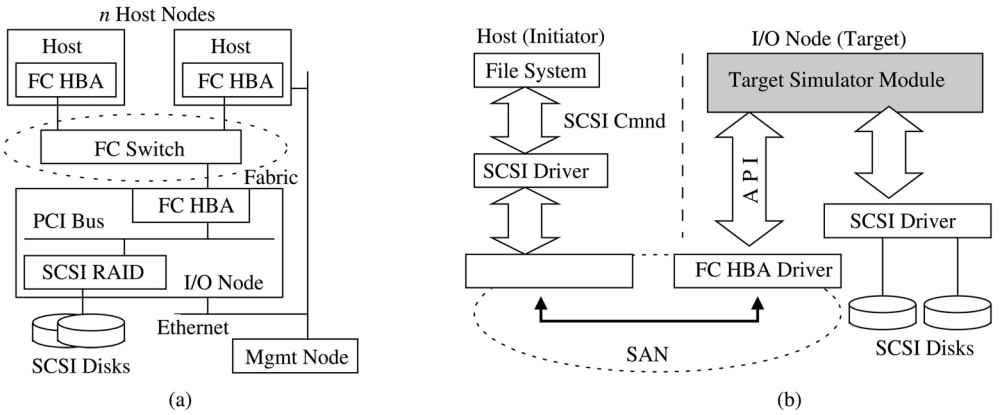

Fig.1. TH-MSNS Hardware/Software architecture. (a) Hardware architecture. (b) Software architecture.

host bus adapters (HBAs) and an FC Switch, a large capacity SCSI-RAID subsystem and a management (mgmt) node. The I/O node connects the SCSI disks and the FC switch. Its main function is to provide network disks for host nodes via the fiber channel connection. The FC devices handle the FC protocol, which is a storage protocol[9]. Because the network disk adheres to SCSI specifications, the mapping mechanism is fully transparent to the host file system according to SCSI protocol[10] on the host nodes and the network disk can be logically treated as the hosts' local physical disk, providing all the features that a local disk would have. Thus, hosts have network disks, and have the full capacity to build either a single file system or a shared file system (e.g., a global file system) on the disks with the cooperation of other nodes. Fig.1(b) depicts that the real I/O requests begin at the host file system and are transferred to the I/O node by the host SCSI protocol stack and the FC connection. The software module with the name of Target Simulator redirects these requests to the SCSI-Raid subsystem where it finally accomplishes consolidated storage.

The TH-MSNS differs from the traditional FC-SAN because it uses software rather than hardware to control the I/O flow. Because the implementation of the TH-MSNS design provides consolidated storage and the well-designed software module has full control of the I/O path, we could add a software module into the I/O node, taking the advantage of the consolidated storage to realize remote mirroring and recovery.

3 Remote Mirroring Design Based on the TH-MSNS

From the information described above, we adopted an architecture similar to the I/O node and added a new node called the mirror I/O node to the system's fabric, forming a dual-node storage cluster with the original I/O node. We were able to use the redundant storage route provided by the storage cluster to manage data replication. Unlike an ordinary cluster, the two nodes in the storage cluster were not on the same level; we used the added node to act as a secondary storage node, providing network devices for the I/O node. Then we achieved remote mirroring by replicating the I/O node's data, which is the SAN's consolidated data.

From this design, we developed a driver to handle all the I/O commands and data, so we could easily determine what the commands were and could easily replicate the data.

3.1 Hardware Architecture

Fig.2 illustrates the hardware architecture of the dual-node storage cluster.

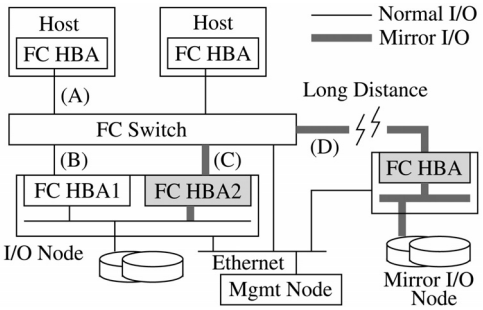

Fig.2

In this architecture, there are four different functional devices: (A) Multiple hosts, same as in Fig.1(a), are the components of the computing cluster, providing the network service or high performance computing for network users. The hosts serve as storage clients in this FC-SAN system. (B) The I/O node acts as the consolidated storage pool provider. (C) The Mirror I/O node is the storage pool for replication of the I/O node, protecting the data when the I/O node fails. (D) The management node controls all of the nodes listed above.

There are two different types of I/O operations in the mirroring system. One is the normal I/O operation, which follows the path of (A), (B) in Fig.2, from the host server to the I/O node. These normal I/O operations are used to read and write application data from and to the disks. The other is the mirror I/O operation,

which follows the path of (C), (D) in Fig.2 from the I/O node to the remote mirror I/O node. These mirror I/O operations contain the replicated data to be stored on the remote site.

3.2 Software Architecture

The key function in this remote mirror system is to produce fully redundant data replication. Fig.3 lists the mirroring system software, and demonstrates details of the extended software architecture presented in Fig.1(b). The modules indicated by grey color form the mirroring path, which is denoted as (C) and (D) in Fig.2. In addition, in the normal I/O path from the host to the I/O node, we made a few modifications before the commands and data were redirected to the SCSI-RAID array. We adopted a layer named the "cmnd/data replicating layer & response handler" to control the redundant data replication. Since cmnd/data replication and its corresponding results are handled on the I/O node, the process of mirroring is concealed behind SAN framework and thus transparent to the storage clients. The data mirroring is totally server-free, which means that the hosts can perform all operations on networked storage devices such as choosing various file systems, either single or shared, without the awareness of the existing of mirroring process. And since all the mirroring operations are in SCSI layer, it is supposed to incur small overhead by evading the participation of file I/O. Detailed performance analysis will be given in Section 5.

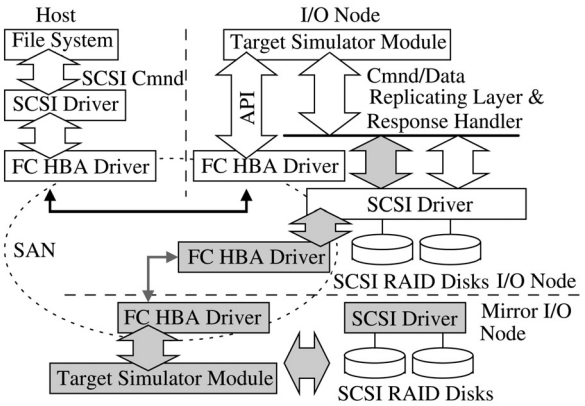

Fig.3

3.3 Mirroring Process for Write Requests

Read-like requests can be handled on the I/O node since no new data is generated. As a consequence, mirroring path only needs to handle write-like operations, which modify the data on disks. Fig.4 illustrates how a write request is handled, according to the following procedures.

1) Each write request from front-end applications are translated into SCSI Command (SCSI Cmnd)[10] and data by the operating system on the host side. The SCSI Cmnd/data can be packaged into Fiber Channel Protocol (FCP) frames[9,11] and will be transferred to the FC HBA on the I/O node side, where the packaged frames are transformed back to SCSI Cmnd/data.

2) The Cmnd replicator in the Cmnd/data replicating layer on I/O node produces a new clone of the SCSI Cmnd.

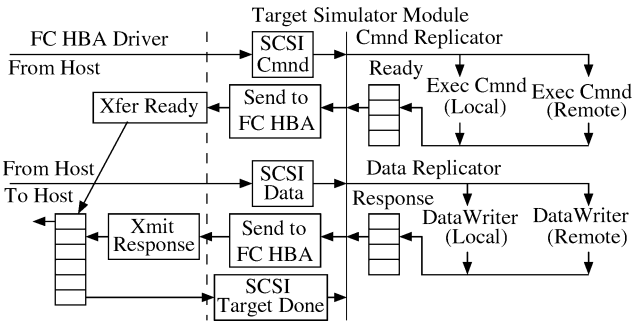

Fig.4. Processing of write commands.

3) The original SCSI Cmnd is processed by the local SCSI system on the I/O node.

4) The cloned SCSI Cmnd passes through the SAN fabric again, marked as grey-colored units in Fig.3, following a very similar way like steps 1) and 3). It is finally handled on the mirror I/O node.

5) The I/O node collects responses from steps 3) and 4). If both are correctly processed, an "Xfer ready" messages is sent back to the host via the FC connection, to indicate a ready signal for accepting data.

6) Data of this write request is handled similarly as steps 1) $\sim$ 5) illustrate. After data is written on both local and remote I/O nodes, an "Xmit response" is sent back to the host to indicate the completion of this I/O request.

This cluster architecture is suitable to adopt either synchronous or asynchronous mirroring. Fig.4 indicates a fully synchronous mirroring, in which the write request completion on both I/O node and remote mirror I/O node defines a write completion for the hosts. However, we can also change the above steps 5) and 6), to allow the I/O node to send a write completion message to the host immediately after the operation completes on the I/O node. It is an asynchronous mirroring and requires a log file to record modifications on the I/O node, which helps data replicating to the mirror node asynchronously.

The asynchronous method will introduce smaller extra latency for write I/Os, compared to the synchronous method, especially in a slow network environment. However, the synchronous method has the advantage that data on the mirror node is completely up-to-date. In this design, we chose an FC connection, which was very fast (2Gbps), thus the synchronous way is more suitable for safe data storage consideration. The performance of synchronous remote mirroring is analyzed in Section 5.

4 Key Techniques

We implemented this architecture on Linux, and the whole system worked in the kernel space, eliminating the need to copy between the kernel space memory and the user space memory. The design was implemented following several key techniques, as described below.

4.1 Software LUN Masking

The mirror I/O node, the I/O node and the hosts were located in the same SAN fabric. Since there were two principles in the system: 1) an active storage I/O node could map its SCSI disk to the hosts as a network disk; 2) as previously mentioned in [6], the mapping process is transparent to the host system, and the network disk operates identically as a local disk, logically. If no mask were added, the hosts would be able to see all the physical disks on the mirror storage cluster's two nodes. Since the hosts could not tell the difference between the disks on the I/O node and the mirror I/O node, they would probably write data on the mirroring disk, which would lead to confusion in the storage cluster. Additionally, we used a two-level (primary/secondary) architecture so that the network disks on the I/O node, provided by the mirror I/O node, could again be seen as the host's network disk on the hosts. Hence, two masks have to be added: one on the I/O node, and the other on the mirror I/O node. Only with both of the masks in place could we guarantee that the remote mirror node's disks remained unseen to the host.

4.2 Disaster Tolerance

There are three main failures that are likely to occur in the mirror FC SAN system.

1) Failure of a Single Disk in a Mirror Pair

Hardware errors can cause disk access failures or even disk disappearance on the I/O node. When a disk is lost, it returns an abnormal SCSI result after commands or data are sent to it. By checking the returned SCSI result, the system can recognize the disk error. In the design, for each SCSI disk on the I/O node, we assigned a remote disk on mirror I/O node to form a mirror pair. We built a map of the local and the remote disks on the I/O node, describing the relationship between the local and remote disks and their status. After detecting a local disk failure in the mirror pair, the Target Simulator Module in the I/O node will resend the current and incoming requests or data to the mirror disk until the failed local disk on the I/O node is recovered. Because the mirror disk contains a very up-to-date data, after redirection, data service can continue without any lost.

Since the failure detection and I/O redirection to a mirror disk are handled in the kernel module on the I/O node, it does not require any interference of the front-end hosts and thus is much faster than the traditional heartbeat based failure tolerant method.

2) Failure of the I/O Node

It is a more complicated situation when the I/O node crashes or the FC link between the I/O node and FC switch breaks. Because every FC device registers to the FC switch (actually, the name server on the FC switch), such a failure would be detected by the FC switch first. Then the FC switch alerts the management node, and a daemon on the management node starts the fail-over operations. In this case, the I/O node can no longer be accessed, and the dual-node cluster breaks, degrading to a one-node cluster that uses the active mirror I/O node to continue storage services. The whole system's architecture changes, as does the membership of the two nodes. To address this problem, we modified the software masking as previously described, allowing the mirror node disk to become directly accessible by the hosts. If we are doing a synchronous mirroring, the data on the mirror node is up-to-date. So the mirror node can be used to run the whole data storage service.

3) Failure of the Mirror I/O Node

Because the storage cluster uses primary/secondary architecture, the failure of the mirror I/O node does not have any impact on the original storage path. In such a circumstance, the architecture is degraded too, with the I/O node continuing the data service.

Any failure (of the I/O node disk, the I/O node, the mirror I/O node disk or the mirror I/O node) must be logged after the modification. The log should keep all records during the failure, and thus can provide significant speedup for the recovery process after the system recovers from the failure.

4.3 Online Resynchronization

Any failed node would need a data resynchronization after recovering because of the demand to maintain data consistency. In this framework, we implemented a module to achieve an online resynchronization. The "online" feature means that the data storage service can continuously be provided even on a disk that is currently in recovering.

If the hosts generate write requests on this recovering networked disk, there may be two writing sources issuing write operations on the same block. It is called write competition. We used the following process to solve the problem of write request competition. Suppose data is resynchronizing from disk A to disk B. The resynchronization thread works with a continuously increasing address. As an example, assume the thread is currently handling address k, and the normal write request from the host is trying to access address n. The resynchronization thread will adopt the following rules.

1) If $n > k$, the resynchronization thread will reach this address n in the future, so the write request could be handled on disk A only. Later, the resynchronization

thread will copy the modified data on address n to disk B.

2) If $n < k$, the resynchronization thread has already processed address n. To maintain data consistency, the write request should be handled on both disks A and B.

3) If $n = k$, this is the most complicated situation that causes competition. To maintain data consistency, the normal write request and data from the host are first buffered. Then, after the resynchronization thread has finished this block, the write request from the host is invoked and processed on both disks A and B.

To speed up the data recovering, we can use a log file to record newly written data during disk failures or I/O node failures, as described above. Since the log file can also be rearranged in the block address order, the above procedures to prevent write competition are still applicable.

5 Performance of Synchronous Mirroring

Using a 25-kilometer 2Gbps Fiber Channel cable as the mirroring transferring layer, we tested the performance of the dual-node remote mirroring system. Since the mirroring operation results in latency only occurring for write requests, we mainly tested the penalty of write requests after adding the mirroring operation. We designed two tests on a synchronous remote mirroring implementation, as described below.

Test 1 studied the storage system's performance under a heavy workload. In this test, we used an Ultra 160 SCSI bus on an I/O node to connect 7 SCSI disks, resulting in a maximum throughput of 160MB/s. We adopted benchmark IOZone on 7 hosts to issue 100% sequential write requests. Table 1 lists the throughput for write requests with different sizes of records, from 4KB to 256KB. The average I/O throughput for mirroring was 143.57MB/s, while the average I/O throughput without mirroring was 145.97MB/s. Both of these figures were nearly 90% of the peak throughput of the Ultra 160 SCSI bus, while the mirroring performance only resulted in a write penalty of 1.64%. This proves that the synchronous mirroring performed very well under a heavy load.

Test 2 was conducted on a single mirrored disk. In this environment, the throughput of an Ultra 160 SCSI bus was no longer the bottleneck, so the real software cost of the mirroring could be seen. We used benchmark I/OMeter on a host to issue 100% sequential write requests of 4KB. The benchmark I/OMeter also provides response time statistics.

Table 1. Comparison of Storage Node Bandwidth with Heavy Loads

Record Size (KB)		4	8	16	32	64	128	256
Throughput (MB/s)	Mirror	143.64	143.62	143.52	143.70	143.66	143.36	143.46
	No Mirror	145.26	145.42	145.66	146.16	146.25	146.43	146.60

Table 2. Comparison of the Throughput and Average Response Time

	Mirror	No Mirror
Throughput	26.13 (MB/s)	26.55 (MB/s)
Avg. Resp. Time	0.1489 (10^{-3}s)	0.1465 (10^{-3}s)

Table 2 shows the throughput and average response time, with mirroring and without mirroring. The comparison shows that the throughout penalty was 1.58%, and the average response time increased 1.64%. This proves that the synchronous mirroring introduces very small overhead under a pure writing environment, by adopting a 2Gbps Fiber Channel as the transferring layer.

These two tests were conducted with 100% sequential write requests. Since read requests do not introduce any additional latency for mirroring, when using mixed read and write requests as the testing sequence, this mirroring method will perform even better. Actually, few applications issue 100% write requests continuously, so this mirroring method promises to perform well in a real application environment.

6 Conclusions

This paper introduced the design and implementation of a dual-node remote mirror storage cluster based on an FC-SAN storage system. Compared to other systems, the new system has the following features. 1) It achieves remote mirroring based on consolidated storage. The mirroring process is fully transparent to the host system. Thus it is a server-free mirroring method, requiring no host resources. 2) It uses a dual-node cluster to provide a data mirror, preserving all the advantages of the software-controlled features of the I/O node. It ensures continued data service when disaster happens, and provides an online recovery mechanism that runs in the background. 3) All the software works in the kernel space, developed as kernel modules, which reduces the number of memory copy operations between user space and kernel space, thus achieving high performance. 4) The dual-node cluster connects by a relevantly simple method and can be easily extended. Since the mirroring process is totally in SCSI layer, it incurs very small overhead by avoiding the involvement of file system. We tested the performance of this dual-node cluster under a fully synchronous mirroring circumstance, the results show that even under a very heavy write workload, the performance penalty is within 2%, compared to the system without mirroring.

References

[1] Molero X, Silla F, Santonja V *et al.* Modeling and simulation of storage area networks. In *Proc. 8th Int. Symp. Modeling,*

Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, USA, 2000, pp.307~314.

[2] Palekar A. Design and implementation of a Linux SCSI target for storage area networks. In *Proc. the 5th Annual Linux Showcase & Conference*, Oakland, USA, 2001, http://www.usentx.org.

[3] Barker R, Massiglia P. Storage Area Network Essentials: A Complete Guide to Understanding and Implementing SANs. Wiley, 2001.

[4] IBM SAN solutions. http://www.storage.ibm.com/ibm-san/products/sansolutions.html.

[5] Veritas Volume Replicator^{TM}. http://www.veritas.com/products/category/ProductDetail.jhtml?productId=volumereplicator.

[6] Shu Ji-Wu, Li Bi-Gang, Zheng Wei-Min. Design and implementation of a SAN system based on the fiber channel protocol. *IEEE Trans. Computers*, 2005, 54(4): 439~448.

[7] Shu Ji-Wu, Yan Rui, Wen Dong-Chan *et al.* An implementation of storage-based synchronous remote mirroring for SANs. In *Proc. IFIP Int. Conf. Network and Parallel Computing* (*NPC*), Wuhan, China, *LNCS 3222*, 2004, pp.463~472.

[8] Yan Rui, Shu Ji-Wu, Wen Dong-Chan. An implementation of semi-synchronous remote mirroring system for SANs. In *Proc. GCC 2004 Workshop on Storage Grid and Technologies*, Wuhan, China, *LNCS 3252*, 2004, pp.229~237.

[9] Information Technology. Fibre channel protocol for SCSI. Second Version (FCP-2), T10 Project 144D Revision 7a, Nov. 1, 2001.

[10] T10 Project 1561-D, SCSI Architecture Model — 3 (SAM-3). Draft, Revision 3, September 16, 2002, http://www.t10.org/scsi-3.html.

[11] Namgoong J C, Park C L. Design and implementation of a fibre channel network driver for SAN-attached RAID controllers. In *Proc. 8th Int. Conf. Parallel and Distributed Systems*, Kyongju City, Korea, 2001, pp.477~483.