An Implementation of Storage-Based Synchronous Remote Mirroring for SANs

SHU Ji-wu, YAN Rui, WEN Dongchan, ZHENG Weimin

Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, China

Abstract. Remote mirroring ensures that all data written to a primary storage device are also written to a remote secondary storage device to support disaster recoverability. In this study, we designed and implemented a storage-based synchronous remote mirroring for SAN-attached storage nodes. Taking advantage of the high bandwidth and long-distance linking ability of dedicated fiber connections, this approach provides a consistent and up-to-date copy in a remote location to meet the demand for disaster recovery. This system has no host or application overhead, and it is also independent of the actual storage unit. In addition, we present a disk failover solution. The performance results indicate that the bandwidth of the storage node with mirroring under a heavy load was 98.67% of the bandwidth without mirroring, which was only a slight performance loss. This means that our synchronous remote mirroring has little impact on the host's average response time and the actual bandwidth of the storage node.

1 Introduction

Remote mirroring ensures that all data written to a primary storage are also written to a remote secondary storage to support disaster recoverability. It can be implemented at various levels, including the file system, the volume manager, the driver, the host bus adapter (HBA) and the storage control unit[1] [14]. In general, there are two locations at which mirroring are implemented: the storage control unit and the host. Each location has its own advantages and disadvantages.

IBM's Peer-to-Peer Remote Copy (PPRC)[1] and EMC's Symmetrix Remote Data FacilitySRDF[2] use a synchronous protocol at the level of the storage control unit. Today's storage control units contain general-purpose processors, in addition to special-purpose elements for moving and computing blocks of data. Therefore, remote mirroring is provided by storage subsystems as advanced copy functions. But it costs a lot and depends on the actual disk's storage subsystem. Veritas's Volume Replicator [3] is a remote mirroring solution at the level of the host's device driver. It intercepts write operations at the host device driver level and sends the changes to a remote device. So it is kept independent of the actual storage unit. However, it takes a toll on the host CPU cycles and communication bandwidth, and it is difficult to manage because it needs to interact with all the hosts where replication software is installed. NetApp's SnapMirror[4] is a remote mirroring solution using an asynchronous protocol at the level of the host's file system, but it is not suitable for block-level I/O access in the SAN environment.

In this study, we designed and implemented synchronous remote mirroring at the level of the storage control unit for Tsinghua Mass Storage Network System (TH-MSNS) [5][6], an implementation of the FC-SAN. Based on the high bandwidth and long-distance linking ability of dedicated fiber connections, this approach provides a consistent and up-to-date data copy in a remote location to meet the demand for disaster recovery. This implementation of remote mirroring has no host or application overhead, and it is also independent of the actual storage unit. In addition, we present a failover solution for disk failure. The performance results indicate that our synchronous remote mirroring does not have a significant effect on the average command response time or on the actual bandwidth of the storage node.

2 Introduction of the TH-MSNS

The TH-MSNS[5][6] is an implementation of an FC-SAN. In the TH-MSNS, the storage nodes provide storage services. A storage node is composed of a general-purpose server, SCSI disk arrays, and fibre channel adapters, and it has a software module named the SCSI target simulator [7][8] running on it. By using the SCSI target simulator to control the I/O process to access disk arrays, the TH-MSNS can implement basic functions of FC disk arrays while only using general SCSI disk arrays. Because of this, it is low cost, highly flexible and can achieve considerable performance [5]. Figure 1 shows the I/O path of the TH-MSNS. The file system and SCSI driver in the host converts the application's I/O

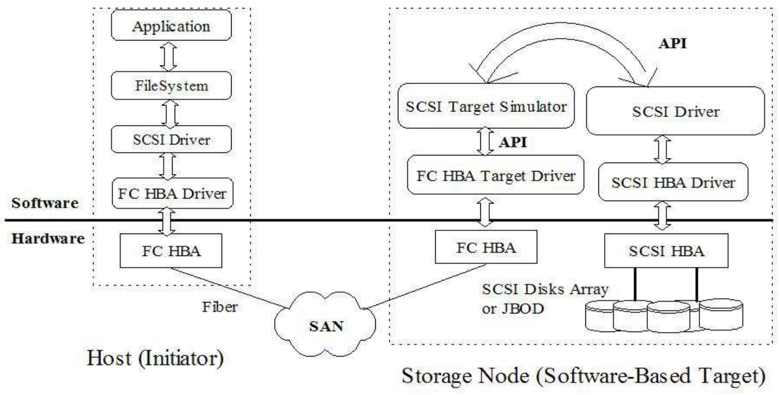

Fig. 1. The I/O path of the TH-MSNS

requests to SCSI commands and data, and then the FC HBA driver encapsulates them into FC frames using Fibre Channel Protocol (FCP)[9] and sends them to the SAN. When the FC HBA on the storage node receives the frames, the FC target driver transforms them back into SCSI commands and data. Then the SCSI target simulator, a kernel module running on the storage node, queues and fills up the SCSI requests' structures, and finally prompts the API of the SCSI driver layer to commit the SCSI requests to the SCSI subsystem. After the SCSI subsystem has completed the SCSI commands, the SCSI target simulator returns the command status or data to the host. Therefore, by the coordination of the SCSI target simulator and the FC target driver, the SCSI disk arrays of the storage node can be directly mapped to the host as its own local disks. So the storage node is equal to the storage control unit in the SAN environment at the basic function of storage service.

3 Design and implementation of remote mirroring for the TH-MSNS

3.1 The Architecture of Remote Mirroring

Figure 2 shows the architecture of remote mirroring for the TH-MSNS. We

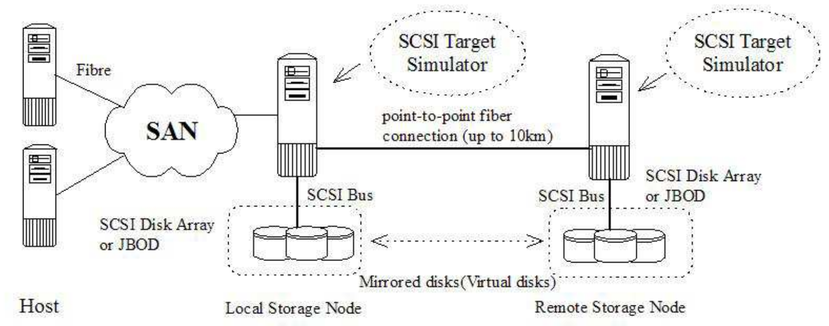

Fig. 2. The architecture of remote mirroring for TH-MSNS

added a remote storage node with the same structure and configuration as the local storage node. By adding an FC HBA in the local storage node to connect point-to-point with the remote storage node, the remote storage node's disks are regarded as the local storage node's own disks. Therefore, the storage node and remote storage node can constitute a mirrored pair. The write commands from the host can be mirrored to the remote storage node by the SCSI target simulator on the local storage node. So the data on the storage node can be mirrored to the remote storage node. The remote storage node can be located up to 10km away from the local storage node using fibre channel technology, and this distance can also been extended by using extenders.

The advantages of this approach of remote mirroring are as follows.

- It has low cost and high flexibility because the remote mirroring is implemented by software modules, not by special hardware.

- The actual storage unit is independent because the host's SCSI commands are redirected to the storage node's SCSI driver layer, which provides a common API for I/O access and hides the detail of the low-level storage subsystem.
- The mirroring process is transparent to the host, and does not consume any of the host's resource. Moreover, the remote storage node is also transparent to the host, because the SCSI target simulator can prevent the remote disks being mapped by the host.

3.2 Synchronous Mirroring

Synchronous remote mirroring writes data not only to a local disk but also to a remote mirror disk at the same time. The acknowledgement is not be sent back until all the data is written to both disks. In many cases, both copies of the data are also locally protected by RAID[10]. This approach provides a consistent and up-to-date copy in a remote location to meet the demand for disaster recovery.

In the mirroring architecture we presented, the SCSI target simulator on the local storage node receives SCSI commands and data from the FC target driver. Then it converts each write command into a pair of write commands to mirrored disks, queues them into different request queues, and finally prompts the SCSI driver to process them. Actually, the local write command is sent to the local disk, and the remote write command is sent to the 'network' disk mapped by the remote storage node. The remote write command is received by the SCSI target simulator on the remote storage node through the point-to-point fiber connection. The acknowledgement is not sent back to the host until both the local and remote write commands have been completed. Figure 3 shows the local and remote I/O path of synchronous mirroring. The dashed line represents the I/O path of the remote write commands, and the solid line represents the I/O path of the local read/write commands. In order to reduce

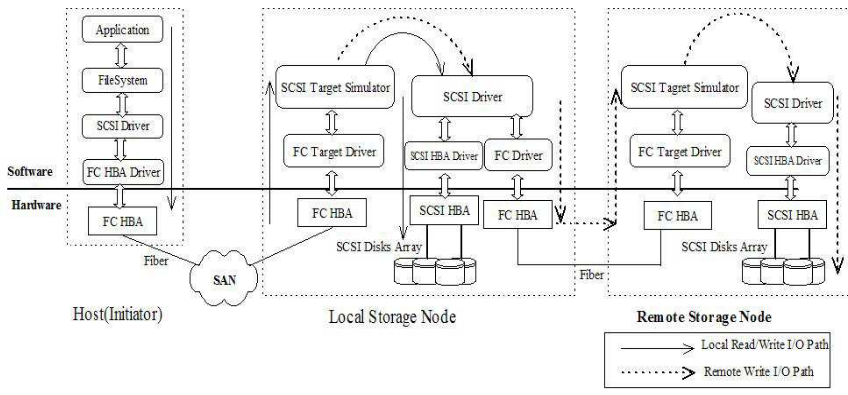

Fig. 3. The I/O path of remote mirroring for the TH-MSNS

the command processing time, local and remote write commands can share the same data buffer. It is not necessary to apply the data buffer for the remote write command; the data pointer only has to be pointed to the data buffer of the local write command. Local write and remote commands can be committed to the SCSI driver at almost the same time, and can be executed by different HBAs concurrently. In this way, commands can be executed more efficiently.

There are two device chain structures in the SCSI target simulator: the local disk chain and the remote disk chain. Figure 4 shows the local and remote disk chains. The structure of the local disk in the local disk chain contains a pointer

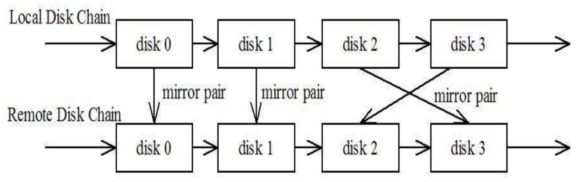

Fig. 4. The local and remote disk chains

which points to the structure of its mirrored disk. The relationship of mirrored disks can also be created between two local disks, just like the software RAID 1. When a SCSI command arrives, the SCSI target simulator analyzes the SCSI command's target disk and finds the remote mirrored disk through the pointer mentioned above. Then the SCSI target simulator fills the mirrored write request using the information of the mirrored disk such as the number of the SCSI host bus, target, or LUN and so on. Furthermore, the SCSI target simulator only maps the local disk to the host, so the remote disk is invisible to host.

3.3 Disk fail over

Although both the local and remote disks are also locally protected by RAID, many circumstances could cause a SCSI command's failure. Some examples include the power failure of the disk array, an unplugged SCSI cable, a break in the fiber connection or the failure of the RAID. Because of this, it is necessary to monitor all write commands locally and remotely. When a command is returned with an abnormal status, some actions must be taken immediately to ensure the services' continuity.

For the accidents mentioned above, SCSI commands will timeout. The SCSI driver layer will try to recover it or retry command. If these actions fail, the SCSI driver will return the SCSI command with the timeout status to the SCSI target simulator. The SCSI target simulator analyzes the status of the local and remote command, and adopts different measures to meet different instances.

- The remote write command fails, but the local command is ok. In this case, the remote disk is assumed to be defunct. The mirror relationship needs to be broken; the corresponding information should be recorded for later resynchronization.

- The local write command fails, but the remote command is ok. In this case, the local disk is assumed to be defunct and the mirror relationship must be severed. The corresponding information must been recorded for later resynchronization. The most important action is to redirect the read and write commands on the local disk to the remote mirrored disk.
- The local read command fails. In this case, the local disk is assumed to be defunct. The mirror relationship is broken, and the corresponding information is recorded for later resynchronize. The most important action is to redirect the read and write commands on local disk to the remote mirrored disk.

In order to perform the disk failover, the local disk's structure contains a status identifier to record the status of the local disk. Table 1 shows the possible status of the local disks. In addition, some key remote mirroring implementation techniques, such as software LUN masking, online resynchronization and disaster tolerance, have been introduced in another paper [11].

Device's state	Description
DEVICE_OK	Disk is ok.
DEVICE_DEFUNCT	Disk is defunct. If disk has mirrored, it means that both local and remote disks are defunct.
DEVICE_MIRRORED	Disk has mirrored, both local and remote disks are ok.
DEVICE_LOCAL_DEFUNCT	Disk has mirrored. Remote disk is ok, but local disk is defunct.
DEVICE_MIRROR_DEFUNCT	Disk has mirrored. Local disk is ok, but remote disk is defunct.
DEVICE_SYNCING	Disk is synchronizing.

Table 1. The status of the local disk

4 Performance evaluation

The synchronous remote mirroring system was tested, and its performance was evaluated and analyzed. Because the read command is executed locally in the process of mirroring, our testing only used the write command. Table 2 shows the test configuration of the host and storage nodes.

Host A

CPU	Intel Xeon 700MHz $\times$ 4
Memory	1G
OS	Linux (kernel: 2.4.18)
FC HBA	Emulex LP9822Gb/s

Host B

CPU	Intel Itanium 2 1GHz $\times$ 2
Memory	2G
OS	Linux (kernel:2.4.18-e12)
FC HBA	Emulex LP982(2Gb/s)

Storage node and its storage subsystem

CPU	Intel Xeon 2.4GHz × 1
Memory	1G
OS	Linux (kernel: 2.4.18)
FC HBA	Emulex LP982 (Initiator mode2Gb/s) Qlogic ISP 2300 (Target mode2Gb/s)
RAID Controller	Adaptec Ultra160 RAID Controller 2110S
SCSI Disks	Seagate Cheetah (73GB 10KRPM) × 7 configured as JBOD

Table 2. Test configuration of hosts and storage node

4.1 Comparison of average command response times

The average response time of the command is a very important factor to evaluate the performance and quality of services. In this test, a host issues commands with different data block sizes to its one 'network' disk, which is provided by the local storage node. The goal is to compare the average response time of each command both with mirroring and without. The Iometer[12] benchmarking kit was used. The host issues sequential write commands with different block size ranging from 64KB to 2048 KB. This test adopts a host configured as host A (given above) and two storage nodes: a local storage node and a remote storage node. Table 3 shows the test results. The results show that synchronous mirroring has little impact on the average command response time for different block sizes.

Block Size	Command Average Response Time	
	No mirror(ms)	Mirror(ms)
64KB	1.718	1.795
128KB	3.482	3.573
192KB	5.335	5.357
256KB	7.083	7.202
512KB	13.573	14.469
768KB	21.214	21.561
1024KB	24.797	28.610
1536KB	42.245	44.048
2048KB	58.141	58.323

Table 3. Comparison of average command response time on different block sizes

4.2 The total execution time for replication in a large amount of data

In this test, we used command dd on the host to replicate a large amount of data (10GB-50GB) to the network disk. By comparing the total execution time with and without mirroring, we evaluated the performance of synchronous mirroring. Command dd is able to directly generate read/write block-level requests. This test adopts a host configured as host A and two storage nodes. Figure 5 shows the test results. The results show that synchronous mirroring has little impact on the total execution time for replicating a large amount of data.

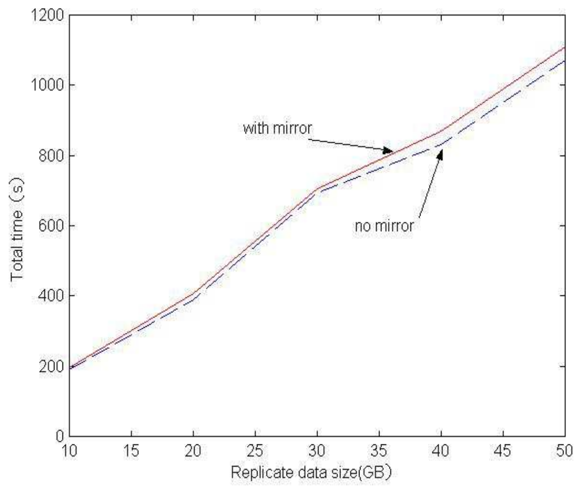

Fig. 5. The total execution time for replication of a large amount of data

4.3 The storage node bandwidth with heavy loads

In this test, we adopted seven hosts, local and remote storage nodes which both have 7 SCSI disks and are mapped to the hosts. Each host accesses its one network disk respectively. We ran IOzone [13]benchmarking kit on each host to offer the local storage node a heavy load. The test used sequential 100% write requests with different record sizes ranging from 4 KB to 4096 KB. The hosts file system were ext2, and the test files size was 15 GB. We compared the bandwidth of local storage node with mirroring and without mirroring. The seven hosts were configured as host B. Figure 6 shows the results. The results show the bandwidth of storage node with different record sizes. With the host cache, the results appear higher than actual bandwidth of the storage node. In the figure, the black bars represent the results without mirroring, with an average bandwidth of 144.314MB/s, while the white bars represent the results with mirroring, with an average bandwidth of 142.399MB/s. The bandwidth of storage node with mirroring is 98.67% of that without mirroring. The performance loss is very little.

In addition, the SCSI HBA we adopted in the test was an Ultra 160 card, so the max throughput in theory was only 160Mbyte/s. Furthermore, the storage node is a server which is only responsible for I/O operations without caring other services, so its CPU has not heavily load. Actually, its CPU utilization is less than 20% at most time.

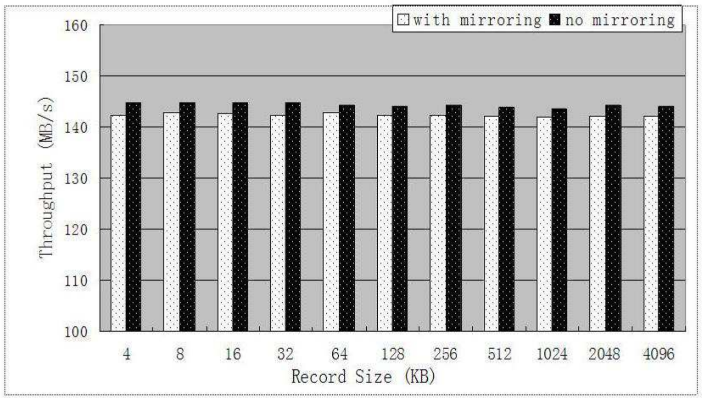

Fig. 6. A comparison of storage node bandwidth with heavy loads

5 Conclusion

In this study, we designed and implemented a storage-based synchronous remote mirroring system for the TH-MSNS. Based on the high bandwidth and long-distance linking ability of dedicated fiber connections, this approach provides a consistent and up-to-date data copy in a remote location to meet the demands of disaster recovery. This system is independent of the actual storage unit, and the process of mirroring is transparent to the hosts. In addition, we present a disk failover solution. In the performance evaluation, we compared the average command response time with different block sizes, the total execute time in replicating a large amount of data, and the bandwidth of storage node under heavy loads, both with and without synchronous mirroring. The performance results indicate that the bandwidth of storage node with mirroring under heavy loads is 98.67% of the bandwidth of storage node without performing mirroring, a result showing only slight performance loss. This means that our synchronous remote mirroring has little impact on the host's average response time and the actual bandwidth of the storage node.

Acknowledgements

The work described in this paper was supported by the National High-Tech Research and Development Plan of China under Grant No. 2001AA111110.

References

1. A.C.Azagury,M.E.Factor,W.F.Micka. Advanced Functions for Storage Subsystems: Supporting Continuous Availability, IBM SYSTEM Journal VOL 42 ,NO 2,2003.

2. Using EMC SnapView and MirrorView for Remote Backup, Engineering White Paper,EMC Corporation(April 2002).
3. VERITAS Volume Replicator Successful Replication and Disaster Recovery,Veritas Software. Corporation .See http://eval.vertias.com/downloads/pro/volume_replicator_whitepaper.pdf
4. Patterson, R.H., et al. SnapMirror: File-System-Based Asynchronous Mirroring for Disaster Recovery. In First USENIX conference on File and Storage Technologies. 2002. Monterey, CA, USA
5. Shu Ji-wu, et al. A Highly Efficient FC-SAN Based On Load Stream. Xingming Zhou, Stefan Jahnichen, Ming Xu, and Jiannong Cao eds., The Fifth International Workshop on Advanced Parallel Processing Technologies, LECTURE NOTES IN COMPUTER SCIENCE 2834, pp.31-40,2003,
6. Technical Report: Design and Implementation of the TH-MSNS (In Chinese), Computer Science Department, Tsinghua University, P.R. China, 2003, http://storage.cs.tsinghua.edu.cn/
7. Ashish Palekar, Narendran Ganapathy. Design and Implement of A LINUX SCSI Target for Storage Area Networks. Proceedings of the 5th Annual Linux Showcase & Conference, 2001.Oakland ,USA.
8. Jae-Chang Namgoong ,Chan-Ik Park . Design and Implement of a Fibre Channel Network Driver for SAN-Attached RAID Controllers. IEEE Parallel and Distributed Systems, 2001.
9. Fibre Channel Protocol for SCSI (FCP) , ANSI X.272, rev4.5, 1995
10. D.Patterson, G.Gibson, and R.Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", ACM SIGMOD, June 1998.
11. YAO Jun, SHU Ji-wu, ZHENG Wei-minA Distributed Storage Cluster Design for Remote Mirroring based on Storage Area NetworkIEEE Transactions on computers(Submitted), also in http://storage.cs.tsinghua.edu.cn
12. Jerry Sievert. Iometer: The I/O Performance Analysis Tool for Servers. http://www.intel.com/design/servers/devtools/iometer/index.htm
13. IOzone Filesystem Benchmark. http://www.iozone.org/
14. Richard Barker, Paul Massiglia ,Storage Area Network Essentials: a complete guide to understanding and implementing SANS. Publish by John Wiley&Sons, Inc.,New York, 2001.
15. Draft, T10 Project 1561-D, SCSI Architecture Model - 3 (SAM-3), Revision 3 16 September 2002, http://www.t10.org/scsi-3.html