

Design and Implementation of a Continuous Data

Protection System for a SAN Environment

Luo Jun, Shu Ji-wu, Xue Wei

Department of Computer Science and Technology

Tsinghua University, Beijing China

luojun03@mails.tsinghua.edu.cn

Abstract: Data is critical to enterprises. It is very important to quickly and correctly recover data in the case of an error or disaster. Traditional data protection technologies, such as remote mirroring, snapshot and backup, have their own limitations. And they can not provide continuous protection for enterprise data. In this paper, we propose a continuous data protection (CDP) method in a storage area network (SAN). We designed and implemented a CDP system based on a remote mirroring SAN system. For the remote mirroring disks, the old data is copied to the CDP disks on write. All the modified data are stored on the CDP disks. In the event that the local system fails or incorrect data is written to the disks, our CDP system can quickly recover data to any previous time. As it was implemented on the block layer, it is independent of lower SCSI cards and storage devices. It is also transparent to hosts. This paper also proposes methods for continuously storing data and quickly recovering data.

1. Introduction

Data protection is becoming increasingly more important for enterprises. The loss of data or the interruption of service results in huge losses for the enterprise. Traditional data protection technologies include remote mirroring technology, snapshot

technology and backup technology. Remote mirroring can store a copy of the local data in the remote site. When the local site fails, the remote site can provide service. Remote mirroring is an important part of the data protection system. But synchronous and asynchronous remote mirroring can only store current data. If there is an error in the current data, remote mirroring can do nothing. Snapshot can protect data of a specific time. Snapshot is very useful for online data backup and can quickly recover data to the specific time when the error occurred. Backup can protect data permanently. But the time of backup and recovery is slow. None of these three technologies can recover data to any previous time.

NetApp's SnapMirror [4] is based on the NetApp's WAFL [5] file system. It takes snapshots of data according to the time the user sets, and asynchronously transmits revised data to the remote mirroring site between two snapshots. For every WAFL Volume, SnapMirror can support up to 255 snapshots. If an error occurs between two snapshots, it can only recover data to the latest snapshot and will lose the other data between these two snapshots. EMC's SnapView and MirrorView [6] implement snapshot and remote mirroring in the disk controller. They can support up to 8 snapshots, so they

can not provide continuous data protection.

Currently several companies, such as Revivio, XOsoft etc, can provide continuous data protection for enterprise data. But they all require additional software on the host to get data. Revivio's Time Addressable Storage system [7] provides a block-based storage array to hosts, and hosts should configure it as a mirror device of the protected device. Then when the host writes data to the protected device, it will also write data to Revivio's storage array. XOsoft's XOsoft Engine [8] installs a file system filter driver on the hosts, called XOFS, which is implemented as a low-level driver to act in connection with the file system on the host. XOFS captures all changes to a file or directory and records the inverse operation rather than the operation itself. Mendocino's product [9] also requires the installation of an interceptor component on the protected server. And Mendocino uses some application-specific plug-ins to track application states. None of these products are transparent to the host. And they capture data based on the file system, the special application or the block layer. They increase the load of the host.

In this paper, we describe the design and implementation of a CDP system in a SAN environment. Our CDP system is based on the Tsinghua Mass Storage Network System (TH-MSNS) [1]. It copies all modified data to the CDP disks, so it can continuously protect data. Once incorrect data is written to disks, the CDP system can quickly and correctly recover to any previous time. And we integrate the

CDP system into the semi-synchronous remote mirroring system [2]. When the local disk or site fails, the remote mirroring site can provide service. Our CDP system can continuously protect data, unlike traditional data protection methods. It is also transparent to the host and can support any operating system and any application on the host. This paper also proposes methods for continuously storing data and quickly recovering data.

In this paper, we first introduce the TH-MSNS. Secondly we describe the CDP architecture for the TH-MSNS and its integration with the semi-synchronous remote mirroring system. Finally, we discuss the details of the design and implementation of the CDP system.

2. The Architecture of the CDP system for the TH-MSNS

2.1 Brief Introduction of the TH-MSNS

The TH-MSNS is an implementation of an FC-SAN [1]. It adds a general-purpose server as a storage server between Fabric and the storage device to provide storage services. In the storage server, the TH-MSNS has Front End Target Driver (FETD) and SCSI Target Middle Layer (STML) software modules to simulate the SCSI TARGET [3]. So hosts can use any kind of storage device, such as SCSI Disk Array or SATA Disk Array, connected to the storage server as an FC storage device. The TH-MSNS is low cost, highly scalable and can achieve considerably high performance.

2.2 The Architecture of the CDP system

In order to recover data to any previous time, the old data must be protected before new data is written to the disks. We added the CDP module between the STML module and SCSI Driver on the storage server. The CDP module is used to store all modified data, so it can recover data to any previous time.

We added redundant disks to the storage server, and shielded them from the hosts on the CDP module. During initialization, all data is directly written to the disks. After that, the user can enable the CDP module. The CDP module will intercept data, perform address mapping, and write data according to the copy on write (COW) policy.

Figure 1 shows the data write path of the CDP system. When new data arrives, the CDP module will first read old data from the protected disk, write it to the CDP disk. Then the CDP module will write new data to the protected disk.

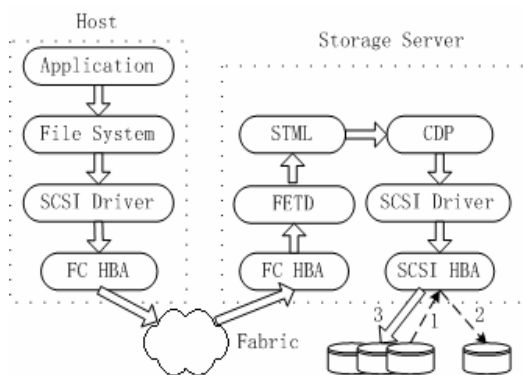


Figure 1 The data write path of CDP system

Semi-synchronous remote mirroring system of the TH-MSNS adds a mirror sub-module into the STML module on the local storage server, and

adds a remote storage server, which has the same structure as the TH-MSNS storage server. The remote storage server is connected to the local storage server through Fabric. The local storage server finds the remote mirroring disks. The hosts send data to the local storage server, and the mirror sub-module of the STML module on the local storage server mirrors the data to the remote mirroring disks. The remote mirroring system can fully meet the requirements of region disaster protection. When the local disk or the local site fails, semi-synchronous remote mirroring system of the TH-MSNS can provide service through the remote mirroring site. But if incorrect data is written to the local disk, it will be written to the remote mirroring disk at the same time. There is no way to recover data to the previous correct status.

As the remote storage server has the same software structure as the structure of the TH-MSNS storage server, we replaced the software with the CDP system on the remote storage server. In order to eliminate the delay between the local storage server and the remote storage server, we mark the time of the write commands to the remote mirroring disks in the mirror sub-module of the local storage server. The CDP module of the remote mirroring storage server will get correct time from the command, and record it on the CDP disk. When incorrect data is written to the disks, we can recover data from the CDP system at the remote mirroring site. So we do not need the CDP module on the local storage server and the CDP system at the remote mirroring site has little effect on the performance of the semi-synchronous remote mirroring

system. Figure 2 shows the data write path of the semi-synchronous remote mirroring system integrated with the CDP system. The solid arrows represent

the local write commands; the dashed arrows represent the mirroring write commands.

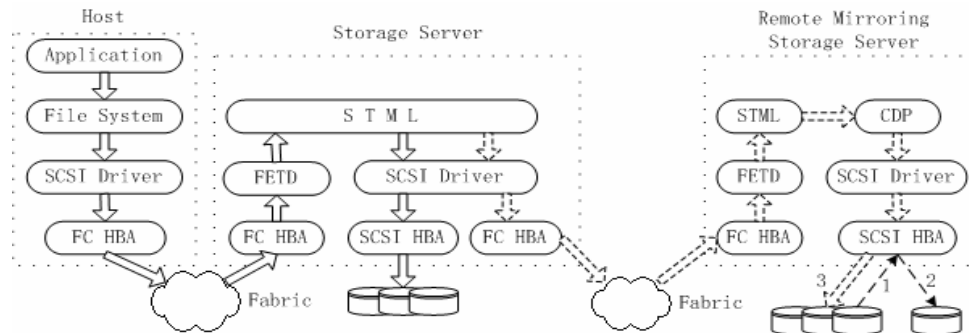


Figure 2 The data write path of the semi-synchronous remote mirroring system integrated with the CDP system

3. Design and Implementation of CDP

3.1 Design of CDP

The main goal of the CDP system is to provide continuous data protection for critical data.

FETD and STML are the two main software modules to simulate SCSI TARGET. The FETD module is mainly used to switch between the SCSI commands and the FC frames. The STML module queries the local SCSI system to get all SCSI devices information and maps them to the hosts. The STML module also transmits all the commands to their disks and obtains the response. So we added the CDP module between the STML module and the local SCSI system in the storage server. The CDP module gets all SCSI devices information and shields the CDP disks from the STML module. When a host queries the disks information, it will not get the CDP disks from the STML module. The STML module sends commands to the CDP module instead of

to the local SCSI system. Then the CDP module will get the old data, map its address to the CDP disk, write it to the CDP disk and write new data to the protected disk.

3.2 Space Management of the CDP Disks

Space can be allocated by a fixed size or by the exact size needed. Our CDP system is implemented at the block layer and is transparent to the hosts. In order to fit different applications, we allocate disk space according to the write command's requirement.

All modified data is written to the CDP disks in sequences and the released space is a comparatively large and sequential area. So we use two lists to organize the free space of the CDP disks. One is called free list, and the other is fragment list. And we use the first-fit algorithm to allocate space. When the CDP system allocates space for a write command, it will choose the first fit item from the free list. If the item does not have enough space, the CDP system will put it into the fragment list and scan the

next item. If no previous data is released on the CDP disks, all the modified data will be continuously stored on the disks. And as the CDP system stores all the modified data, we use the incremental backup method to back up data. For example, we can back up all the data that has been modified and continuously stored on the CDP disks during one week. After the data has been backed up to the tape, the space is released. The amount of backup data is large, so the released disk space will be a large continuous area and the CDP system will scan the fragment list and merge the border space. Then the first item of the free list will be comparatively large. So all modified data during one period of time has good continuity on the CDP disk. In order to provide more flexible mode, such as preserve one day's modified data at disks for analyzing application and free other data. We use the fragment list for that case.

3.3 The Data Organization of the CDP Disks

The CDP system needs to recover data quickly and correctly. Siddha and Gopinath designed a persistent snapshot device driver that used a snapshot tree to maintain all snapshots [10]. Our CDP system needs to record the time of data modification to restore data to any previous time. When a user needs to recover data to some previous time, the CDP system should recall all the modified data during this period. All previous data will be sequentially written to the CDP disks according to their modification time. We used a time list to maintain all protected data, and an index time list to accelerate the lookup

process. The first time list records the modification time and the pointer to the metadata. The metadata records the address mapping of data from the original disks to the CDP disks, the data size, etc. Figure 3 shows the structure of the metadata:

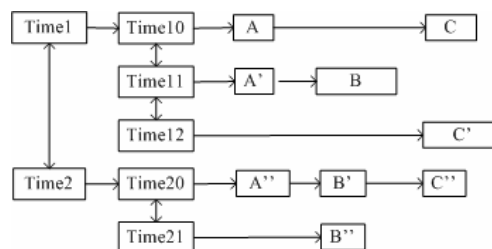


Figure 3 The metadata structure of CDP disks

As the metadata is linked in the list, we can store the metadata at any place of the disk. The head of the list should be placed in a fixed place, so the CDP module can correctly find all of the data. We placed the head of the list in the first block of the disk. In order to accelerate the process of reading the metadata, we stored many metadata items in one metadata block. When the previous metadata block is full, another block is allocated at the fixed size. All protected data of the metadata in one metadata block is sequentially written at the tail of the metadata block. So after the CDP module analyzes the metadata in one metadata block and needs to recover the protected data, it can quickly read data from the disk.

The write anywhere characteristic of the metadata provides good scalability for the CDP system. When the current CDP disks become full, new disks can be added and marked as the CDP disks, and the metadata can be linked together.

3.4 The Metadata Consistency

and Non-Volatile RAM

Software failure or human mistakes are inevitable in practice. The protection of the metadata is very important. The main reasons of system failures are power failure and system crashes including software crash and hardware crash. We used Non-Volatile RAM to protect the metadata in the memory from power failure. So the main cause of system failure is system crashes. We write the metadata to the disk to protect the system from system crashes.

When the write operation is performed frequently, the metadata of the CDP system will be updated quickly. If the CDP system writes every update operation to the disk, the metadata block in the memory will be blocked frequently and the bandwidth for writing data will be affected. But if the CDP system does not update the metadata, the loss of the metadata will result in the loss of data. In order to efficiently protect the metadata and have little effect on the bandwidth for writing data, we used a kernel thread, called the committer, to write the metadata to the disk every ten seconds. During the process of writing the metadata to the disk, the committer thread will lock the metadata and block new commands. As the disk write operation is long, many commands will be blocked. To solve this problem, the committer thread first locks the metadata, makes a copy of the metadata in RAM, and then unlocks the metadata and writes the copy to the disk. Because the copy in memory is quick, it will not block new commands.

The update of the metadata is called the committer transaction. The process of the committer transaction is

as shown below:

Step 1: Write the copy of the metadata block into the journal of the disk.

Step 2: Make a mark in the journal after the protected data recorded by one metadata is written into the disk. Repeat step 2 until all protected data recorded by the copy of the metadata block is written to the disk.

Step 3: Update the metadata block to the disk.

If the system crashes in step 1, the CDP system only recalls the metadata block recorded in the journal after the system reboots. If the system crashes in step 2, the CDP system does recovery according to the metadata block and the marks in the journal. If the system crashes in step 3, the CDP system recovers the metadata from the journal.

3.5 Recovery and Backup

The CDP system needs to perform data recovery in two conditions:

(1) When incorrect data has been written and the user needs to recover data to some previous time.

(2) When one or more local disks or the whole local site fails and the remote mirroring disks or site provide service.

In the first case the remote mirroring site rolls back all the modified data recorded by the CDP module from the current time to the previous time and copies data back to the local disks. The recovery kernel thread first finds the latest time in the list which the user wants to recover to and recovers the protected data at this time. Then the recovery thread traverses through the list, analyzes the metadata, and performs the necessary recovery. For example, a user needs to recover data from the Time21

to the Time11 as shown in Figure 4. The recovery kernel thread will first recover the stored data at the Time11 and record the data address and size into the recovery table. Then the recovery thread analyzes the metadata of the Time12, finds protected data C', recovers data C' and updates the recovery table. At the Time20 data A'' partly overlaps data A' at the Time11, so the recovery thread only recovers additional data and updates the recovery table. Data B' and C'' do not have new data to recover. The recovery thread continues to analyze the metadata of the Time21. Data B'' at the Time21 has no new data, so the recovery thread does nothing. As all the protected data of one metadata block is sequentially written into the disk just after the metadata block, the disk seek time is reduced.

In the second case, the remote mirroring disks have the latest data, and the remote site can provide service. When the local errors are fixed, the user can synchronize the data from the remote mirroring site.

Our CDP system provides a more flexible method for data backup. As all modified data is stored on the CDP disks, the incremental backup can be performed well. The amount of backup data is greatly reduced and the backup time is shortened. The user can also choose different backup modes for different applications. For some special applications, the user can back up all the modified data. And for some common applications, the user can only back up the modified data at some time and abandon the modified data at another time. The backup operation is performed when the system is not busy, for example at midnight. So the backup

operation will not have much effect on the performance of the CDP system.

4. Performance Evaluation

Because the local storage server of the semi-synchronous remote mirroring system logs write commands, the effect of the CDP module on the system performance is not very clear. In order to test the impact more precisely, we set up the CDP system as shown in Figure 1 and tested its performance.

The host uses one 2.4 GHz Xeon processor, with 1 GB memory and one Emulex LP982 HBA (2Gb/s), and run the Linux kernel version 2.4.26. The storage server uses one 2.4 GHz Xeon processor, with 1GB memory and one Qlogic ISP 2300 (Target mode, 2Gb/s), and runs the Linux kernel version 2.4.26. Seagate ST373307LC 10,000 RPM SCSI disks in Dell PowerVault 220s are connected to the storage server via Adaptec 2200S Ultra320 SCSI HBA.

We used the commonly known IOMeter tool to test the performance. We mapped one disk to one host, and the host issued sequential write commands with different data block sizes to the disk to test the delays introduced for the original disk write by the CDP module. Each test runs 4 minutes. Figure 4 shows the results.

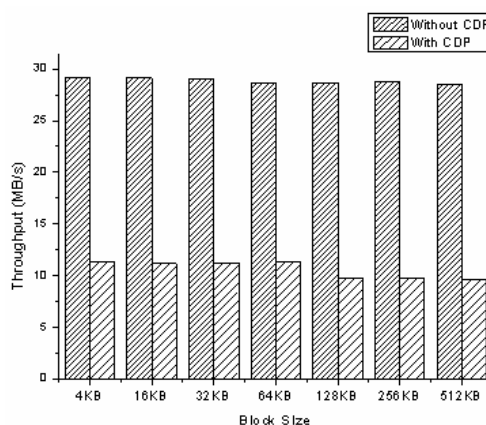


Figure 4 The write performance of CDP

Currently in our prototype we write the metadata to the system disk of the storage server instead of the CDP disks. For larger accesses such as 128KB, the CDP performance decreases from 11MB to 9.8MB, while the performance decreases only from 29MB to 28.6MB without CDP. As the CDP disks and protected disks are connected to storage server by the same SCSI HBA, the cache effect may be reduced for large access.

5. Conclusion

Traditional data protection methods, such as remote mirroring, backup and snapshot, have their own limitations. They can not recover data to any previous time. The CDP system can continuously protect enterprise data. But current CDP systems require additional software on the host, or are designed for specific applications. They increase the load of the host. The applicability of these CDP systems is limited.

This paper describes the design and implementation of a new CDP system which was integrated with the semi-synchronous remote mirroring system to provide full disaster recovery. The CDP system is transparent to the host. It also provides more flexible methods for data backup.

6. Acknowledgement

The work described in this paper was supported by the National Natural Science Foundation of China (No. 60473101) and the National Key Basic Research and Development (973) Program of China (Grant No. 2004CB318205).

References:

- [1] Shu Ji-wu, Li Bigang, Zheng Wei-min. Design and Implementation of a SAN System Based on the Fiber Channel Protocol, *IEEE Transactions on Computers*, 54(4), 2005: 439-448
- [2] Yan Rui, Shu Ji-wu, Wen Dongchang. Design and Implementation of an Efficient Semi-Synchronous Remote Mirroring for SANs. *The 3rd International Conference on Grid and Cooperative Computing (GCC' 2004) Workshop on Storage Grid and Technologies*, LNCS 3252, pp 229-237.
- [3] Ashish Palekar, Narendran Ganapathy. Design and Implementation of a Linux SCSI Target for Storage Area Networks. *Proceedings of the 5th Annual Linux Showcase & Conference*, 2001.
- [4] Hugo Patterson, Stephen Manley. SnapMirror®: File System Based Asynchronous Mirroring for Disaster Recovery. *Proceedings of the FAST 2002 Conference on File and Storage Technologies*.
- [5] Dave Hitz, James Lau, Michael Malcolm. File System Design for an NFS File Server Appliance. *Proceedings USENIX Winter 1994 Conference*.
- [6] Using EMC SnapView and MirrorView for Remote Backup. *Engineering White Paper*, EMC Corporation (April 2002).
- [7] Revivio: Time Addressable Storage: <http://www.revivio.com/>
- [8] Business and IT Requirements for Continuous Data Protection. *White Paper*, XOssoft Corporation (September 2004).
- [9] Mendocino Software: <http://www.mendocinosoft.com/>
- [10] Suresh B Siddha, K Gopinath. A Persistent Snapshot Device Driver for Linux. *The 5th Annual Linux Showcase and Conference on incorporation with USENIX*, Oakland, 2001.