CAWRM:A Remote Mirroring System Based on AoDI Volume

Zhenhai Zhao, Tingting Qin, Fangliang Xu, Rui Cao, Xiaoguang Liu*, Gang Wang*
Nankai-Baidu Joint Lab
College of I.T., Nankai University, Tianjin, China, 300071
Email:{zhaozhenhai1985, paulaqin, xuflnk}@gmail.com, {caorui12001, liuxg74, wgzwp}@yahoo.com.cn

***Abstract*—Nowadays, data reliability and availability are big challenges that the designers of large data centers have to face. Remote mirroring technology is an effective approach for data protection. It maintains a complete copy of primary site at geographically distant locations. Data consistency is the key point of the replication process in the remote mirroring system. This paper improves the consistency strategy of traditional remote mirroring architectures from three aspects: initial synchronization , real-time replication logic and resumption from network failure with the help of a novel logical volume, AoDI. AoDI is an *Allocation-On-Demand Incremental* volume. It uses an appending storage structure to provide the ability of space allocation on demand and fast snapshot. This paper presents two remote mirroring architectures based on AoDI volume. One architecture incorporates the remote mirroring logic with the AoDI structure loosely. It provides a way to construct block-level heterogeneous remote mirroring system and improves random write performance remarkably. Another architecture combines the remote mirroring logic with the AoDI structure tightly. Besides the advantages of the loosely coupled architecture, the tightly coupled architecture simplifies remote mirroring logic significantly. The remote backup process separates from normal write process completely. The remote mirroring module simply copy data sequentially from the local volume (an AoDI volume) to backup site repeatedly, so we call this architecture CAWRM (Copy After Write Remote Mirroring).**

***Keywords*-remote mirroring; AoDI volume; data consistency; loose coupled; tight coupled;**

I. Introduction

Nowadays, the demand for massive data storage increases rapidly. The reliability and availability of data is particularly important. The irreversible loss of data caused by various disasters will lead to immeasurable loss, even bankruptcies and bank failures. So data protection has aroused more and more attention. Remote mirroring technology [1]–[4] is a typical data protection technology. Remote mirroring ensures that all data written to a primary site is also written to a remote backup site as to support disaster recoverability. As the primary site and backup site usually disperse in geographically distant locations, how to transmit the data correctly and fast to the backup site is the key issue of remote mirroring. This paper mainly focuses on the data consistency of remote mirroring [5], [6], which ensure the integrity and accuracy of data.

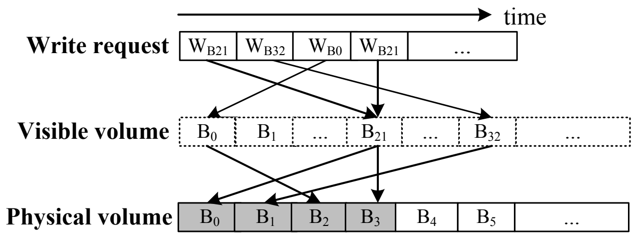


Figure 1. The schematic diagram of AoDI Volume.

Seneca [2] uses a *log barrier* technology to ensure data consistency, however it needs a complex protocol. Abandoning the complex and pursuing the simplicity are the design goal of computer system. So in this paper, we design a simple remote mirroring architecture based on AoDI [7] volume. It guarantees the data consistency naturally. Figure 1 shows the basic design idea of AoDI. It uses an appending rather than overwriting strategy to deal with the write requests. Therefore, it can translate random requests into sequential requests which can speed up random access obviously. Moreover, this log-like structure provides a natural way to implement efficient multi-version snapshots of logical volumes. The motivation of this paper is just to improve traditional remote mirroring architectures using these advantages of AoDI.

We first design a loosely coupled architecture: the remote mirroring module is simply built above AoDI volume, the two layers are independent. Although just a shallow combination, this architecture also provides some distinct advantages with the help of AoDI. First, block-level heterogeneous remote mirroring system can be constructed based on this architecture. That is, we can deploy storage devices of different sizes at the primary site and the backup site, and replicate data between them at block level. Second, AoDI's appending structure improve random access performance significantly.

We also design a tightly coupled architecture. The remote mirroring module comprehends the structure of AoDI, and consciously exploits its characteristics to achieve data consistency in a very clear and efficient way. With the help of the log-like structure of AoDI, the remote mirroring module simply copy data sequentially from the local AoDI volume to the backup site. It no longer concerns with how to intercepts write requests in the normal request processing path, and how to clone the requests, buffer them and send them. Moreover, this architecture unifies the algorithms for initial synchronization, real-time replication and resumption from network failure - just copy. So we call this architecture CAWRM (Copy After Write Remote Mirroring).

II. Related Work

EMC Symmetrix Remote Data Facility (SRDF) [8] is a disk array level remote mirroring technique. Although EMC literature encourages the use of synchronous mode, semi-synchronous mode is available, in which a subsequent write request will be delayed until the completion of the preceding remote write command to avoid inconsistency. [9] proposed a special semi-synchronous remote mirroring system. By using a log mechanism, the system allows a limited number of write I/O operations to proceed before waiting for acknowledgment of receipt from the remote site, which significantly reduces write latency. Meanwhile the consistency is guaranteed.

Seneca [2], an asynchronous remote mirroring protocol, which uses logs at both primary site and secondary backup site. By its transaction mechanism and two-phase commit protocol, Seneca supports write coalescing and in-order delivery, and provides resilience to many kinds and sequences of failures.

Veritas Volume Replicator (VVR) [10] is a volume level remote replication technique like our system. It performs asynchronous replication using a log and transaction mechanism to ensure consistency. SnapMirror [3], another asynchronous and volume level mirroring technology, leverages file system snapshots to provide a consistent copy in a remote site and optimize data transfer.

III. Consistency of Traditional Remote Mirroring

Figure 2 is a traditional architecture of remote mirroring system. In primary site, write request is intercepted and copied by the remote mirroring module. The original write request is delivered to the local disk and the replica is entered to the remote queue. The queue is a FIFO queue, which guarantees data consistency. Remote volume is a virtual map of the physical volume at the backup site constructed by NBD (the Network Block Device) [11] module. NBD sends the request to the backup site via network. The backup site commits the new data to its local disk. We implement our prototype in LVM (Logical Volume Manager) layer [12] of Linux because it is a appropriate place to embed remote mirroring logic.

A. Initial Synchronization

If we create a remote mirror for a volume immediately after its creation, since no useful data has been written to it, we can simply empty both the original and the mirror

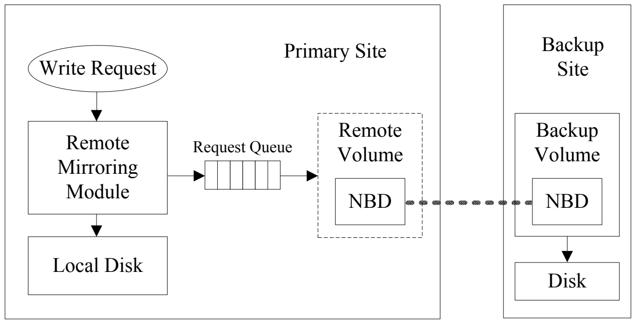


Figure 2. The architecture of traditional remote mirroring system.

volumes. However, if we create a remote mirror for an existing volume, we must copy all valid data from the original volume to the remote mirror. We call this necessary step *initial synchronization*.

The initial synchronization starts exactly after the creation of the remote mirror. In a traditional remote mirroring system, since there is no idea which blocks of the original volume contain useful data, a synchronization thread must be created to copy entire original volume to its remote mirror block by block in background. Considering that modern storage systems become larger and larger, initial synchronization is obviously a extremely time-consuming process. Moreover, the original volume must accept the normal read and write requests from user applications simultaneously because even a brief downtime will cause big losses. Therefore, the remote mirroring module must carefully coordinate synchronization I/Os and normal business I/Os, which leads to complex remote mirroring logic. With the help of AoDI, our new architecture addresses these problems effectively.

B. Real-time Remote Replication

After the completion of initial synchronization, the system starts the real-time remote replication. The write requests from user applications were intercepted, cloned, and then put into the remote queue. Since the requests are sent to the backup site in the same order as they enter the queue, data is guaranteed to be written to the mirror volume in the same order as they are written to the original volume. In other words, this traditional real-time remote replication logic ensures consistency between the primary site and the backup site.

However, to boost transfer throughput, many remote mirroring systems accumulate write requests at the primary site, and then send them to the backup site batch by batch rather than one by one. The backup site must commit each batch atomically to guarantee consistency. A general strategy is to log the batch first, and then write data to the backup volume. If the backup site crashes during batch committing, the log can be used to re-commit.

By utilizing the appending structure of AoDI, our new real-time replication logic is much more simpler than the traditional one.

C. Resumption From Network Failure

Resumption from network failure is the problem a remote mirroring system have to face. The key issue is how to resynchronize the primary site and the backup site. A process like initial synchronization is too inefficient because of a great number of unnecessary block copy operations. A common strategy is to mark the blocks actually need to be resynchronized by an incremental bitmap.

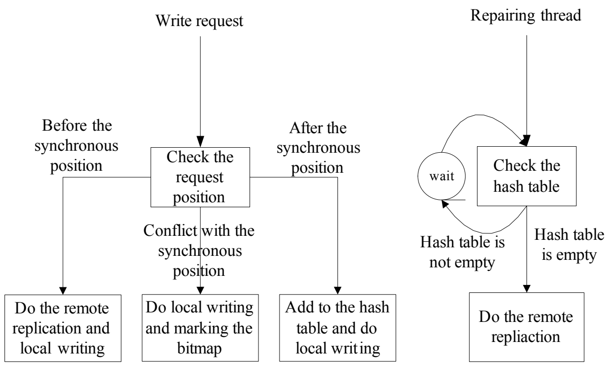
1) Incremental Bitmap: An *incremental bitmap* indicates which blocks were modified during network failure. Each bit corresponds to a unique disk block. A set bit means a modified block, and a zero bit means a unchanged block. Therefore, after the network failure is repaired, we can only copy modified data blocks from the original volume to the mirror volume with the help of the incremental bitmap.

2) Re-synchronization: After the network recovered, the resumption thread should be started to re-synchronize data. It is very similar to initial synchronization except it only copies data blocks corresponding to set bits in the incremental bitmap.

Like initial synchronization, the resumption thread should coordinate between synchronization I/Os and normal application I/Os. More specifically, the application write requests are divided into three categories according to synchronous position, and be dealt with using different strategies respectively.

i) The write request is before the synchronous position. That is, it is in the area has been synchronized. The system can simply perform normal replication and local write.
ii) The write request conflicts with the synchronous position. The resumption thread should suspend until the write request is written to the original volume and the incremental bitmap is set. We can use a hash table to test this kind of conflicts efficiently.
iii) The write request is after the synchronous position. It is inserted into the hash table, and will be removed after it is written to the original volume. Therefore, the resumption thread can test conflicts by looking up the hash table.

Figure 3 shows the logics of the write request processing and the resumption thread.



Write request
Repairing thread
Before the synchronous position
Check the request position
After the synchronous position
wait
Check the hash table
Conflict with the synchronous position
Hash table is not empty
Hash table is empty
Do the remote replication and local writing
Do local writing and marking the bitmap
Add to the hash table and do local writing
Do the remote repliaction

Figure 3. The maintenance of data consistency.

IV. AoDI Volume Based Remote Mirroring System Architecture

A. Design of AoDI volume

As shown in Figure 1, AoDI volume has a double volume structure - visible volume exposed to users and physical volume actually storing data. AoDI adopts appending rather overwriting strategy to deal with write requests. That is, each newly arrived data chunk is stored just behind the last stored data chunk, instead of overwriting its old version (if has). This strategy brings several advantages: easily obtain accurate storage utilization, therefore dynamically allocate storage space according to users' real-time requirements; convert random user write requests into sequential disk write operations. However, since dynamic mapping from user address space to physical address space is used, mapping table must be maintained.

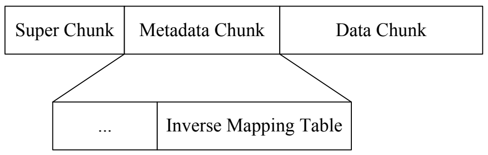


Figure 4. The disk layout of AoDI.

As Figure 4 shows, the disk layout for AoDI consists of three parts: the super chunk, the metadata chunks and the data chunks. In our system, the chunk size is 4KB. To support remote mirroring, we add a inverse mapping table to AoDI's metadata area. It stores mapping from physical address space to user address space. With this inverse mapping table, we can reconstruct the original write request of each data chunk in $O(1)$, which will be sent to the backup site by remote mirroring module. Next we will introduce how to combination remote mirroring with AoDI to provide distinct features and higher performance.

B. Loose Coupled Architecture

A simple way is to treat remote mirroring and AoDI as two independent layers, and build the former simply above the latter in the data path. The remote mirroring module does not concern about the internal mechanism of the AoDI, but consider it as a plain LVM volume. Although this loose coupled architecture is too simple, it still provide some important advantages:

- Improve random write performance. This is directly inherited from AoDI.
- Provide the ability to construct heterogeneous remote mirroring system at block level. In traditional remote mirroring system, the physical size of the remote mirror volume should be greater than or equal to that of its

original volume (generally equal to for cost reason). Otherwise, some write requests to the end zone of the original volume can not find their locations in the backup site. However, since uses appending strategy and has the ability of allocation-on-demand, AoDI generally occupies smaller physical space than its logical size. Therefore, we can use storage devices of different size at the primary and backup sites to construct a AoDI based remote mirroring system. At the backup site, you can build the mirror volumes using plain LVM volume or AoDI. If the latter is used, real-time physical space occupation is also saved in the backup site, especially for multi mirrors case.

- Improve the write performance in the backup site. As mentioned above, in traditional remote mirroring system, a log mechanism is used to ensure atomic batch committing. However, AoDI volume is substantially log-like. Therefore, log recording is avoid, which improves write performance greatly.

C. Tight Coupled Architecture

The loose coupled architecture does not take full advantage of AoDI's structure properties. We design a tight coupled architecture. It is illustrated in Figure 5. Although in this architecture we separate the remote mirroring module from AoDI (request processing module) completely, the two modules are really combined tightly. The remote mirroring module knows AoDI's structure clearly, and simplifies itself greatly using this structure. Since the remote mirroring module is not in the normal data path, the user write requests are not be intercepted and copied to the backup site when they are being processed in the data path. Therefore, we call this architecture CAWRM (Copy After Write Remote Mirroring architecture), distinguished with traditional Copy On Write method.

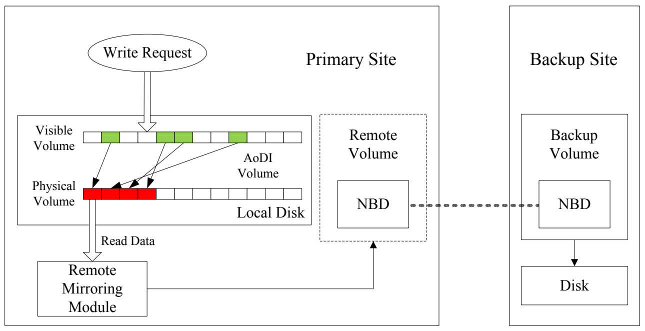


Figure 5. The architecture of tight coupled architecture.

1) CAWRM Algorithm: The core algorithm is shown in Algorithm 1.

Unlike traditional remote mirroring system and the loose coupled architecture, after being created, a CAWRM system starts a sync thread responsible for copying data from the original volume to the remote mirroring volume, instead

Algorithm 1 CAWRM algorithm

```
1: loop
2:   if Network is OK then
3:     if $sync\_ptr = end\_ptr$ then
4:       sleep for a time interval T
5:     else
6:       copy chunks $[sync\_ptr, sync\_ptr + k - 1]$ from
         the AoDI volume to the remote mirror volume
7:       $sync\_ptr += k$
8:     end if
9:   else
10:    sleep for a time interval $T$
11:  end if
12: end loop
```

of embedding remote mirroring logic into normal request processing logic. The thread uses two pointers: $sync\_ptr$ points to the first un-synchronized of chunk, and end_ptr points to the last written data chunk in the AoDI volume. If $sync\_ptr$ exceeds end_ptr, which implies that all the valid data chunks in the original volume have been copied to the backup site, CAWRM simply sleeps a moment. Otherwise, it copies an un-synchronized data chunk to the backup site. The actually remote copy operation is performed by Linux kernel function $kcopyd$. The chunks are copied to the NBD device, and then be sent to the backup site by NBD module. If a network failure occurs, CAWRM just waits until the network connection is repaired, and then goes on copying chunks..

2) Optimization: Separating the remote mirroring logic from the normal request processing logic simplifies the remote mirroring logic greatly. However, competition between normal user requests and read requests is caused by sync thread in the primary site.

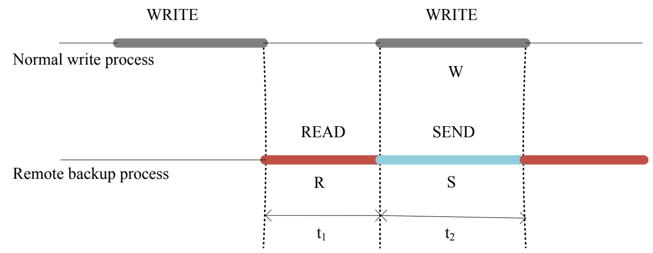


Figure 6. I/O competition in the primary site.

Figure 6 shows that the competition between user write requests and synchronization read requests in the primary site. The notations W, R and S represent the writing speed of the the user application, the reading speed of AoDI volume and the network transmission speed respectively. Suppose that $S \leq W$.

Suppose that the sync thread read B bytes each time. Then the average writing speed of the user application in the primary site can be approximated by W' :

$$W' = \frac{W * t_2}{t_1 + t_2} = \frac{W * \frac{B}{S}}{\frac{B}{R} + \frac{B}{S}} = \frac{W}{\frac{S}{R} + 1} \qquad (1)$$

It can be seen from the (1) that the local writing speed in the primary site are the ratio of network transmission speed to the AoDI read speed. The faster data is read from the AoDI volume and the slower data is transmitted on the network, the faster data is written in the primary site. Otherwise, data is written more slowly. In practice, network speed is usually fixed, so the performance can be optimized by improving the reading speed of the AoDI volume. According to the characteristics of the hard disk, the reading speed would be improved if data was read sequentially and in bulk. Therefore, the sync thread read a large number of continuous chunks from the AoDI volume. That is, we use a relatively large k rather than 1.

3) Advantages of CAWRM: Complete fusion of remote mirroring and AoDI brings several advantages:

i) The remote mirroring logic is simplified greatly. The remote mirroring module (the sync thread) needs not to intercept, clone and send write requests. What it only needs to do is data copy.

ii) Unified consistency processing. Note that when we discuss consistency in traditional remote mirroring system in the last section, we discuss three cases: initial synchronization, real-time remote replication and resumption from network failure. However, CAWRM unifies these three cases. No matter in which state a CAWRM system is, it just copies. The log-like structure of AoDI naturally guarantees the consistency. Moreover, conflicts between user requests and synchronization I/Os will never occur, because the sync thread always reads the area behind the area of new data will be written to.

iii) Reduce the amount of data need to be copied during initial synchronization. Since AoDI knows how many data chunks it stores and where they are stores (from the beginning of the volume to *end_ptr*), it only copies these chunks during initial synchronization. They may be only a small portion of the entire volume.

iv) Certainly, the advantages of loosely coupled architecture are still.

V. Experiments

We implemented both the traditional remote mirroring system and the CAWRM system. Experiments were carried out for the two systems.

A. Experimental Settings

We deployed both the primary site and the backup site on the server with one 2.66GHz Intel Xeon processor, 4GB RAM and a hardware RAID-0 composed of six 74GB SCSI disks. The underlying OS was RedHat AS server 5 (kernel version 2.6.18-128.el5). We employed WANemu [13] to emulate the WAN environment.

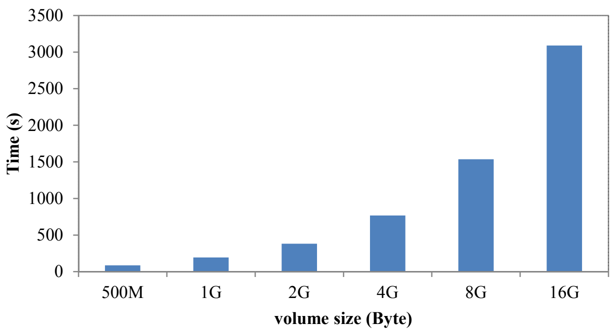


Figure 7. Initialization Time.

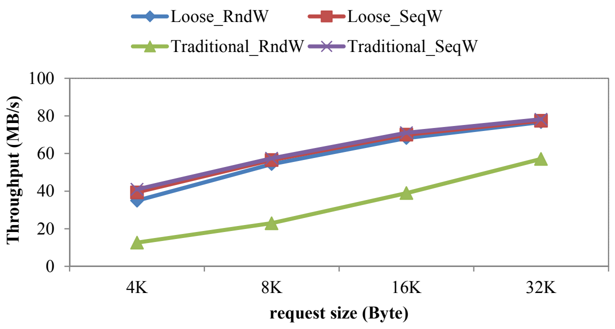


Figure 8. Loose coupled architecture performance.

All sites were connected by Gigabit Ethernet and WANemu v2.2 which executed on another gateway site.

B. Experimental Results

1) Initialization Time: Figure 7 shows the initialization time in the primary site of the traditional remote mirroring system. Different sizes of logical volume corresponding to different time. Because of the stable network transmission speed, it is the linear relationship between the transmission time and the size of the logical volume. However, CAWRM only copies already stored user data rather than the entire volume, which reduces disk I/O and network traffic significantly. It even copies nothing if the remote mirror is created on an empty AoDI volume.

2) Loose Couple architecture: Figure 8 shows the local throughput. We can see that, compared with the traditional remote mirroring system, the loose coupled architecture achieves better random write throughput, and has comparable throughput under other types of workload. This implies that the loose coupled architecture brings lower overhead to the primary site.

3) Tight coupled architecture: Figure 9 shows the local throughput. We can see that, although CAWRM move remote mirroring logic out of the normal request processing path, it achieves almost the same throughput as the loose coupled architecture under all kinds of workload, which

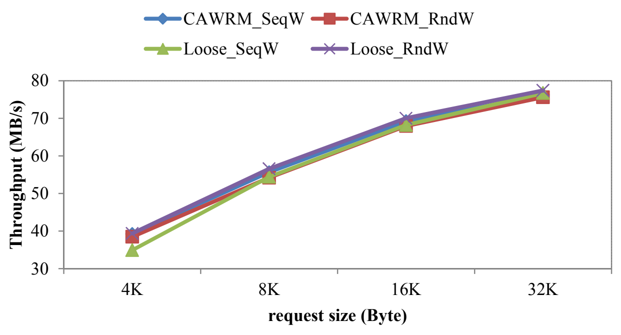


Figure 9. Local throughput.

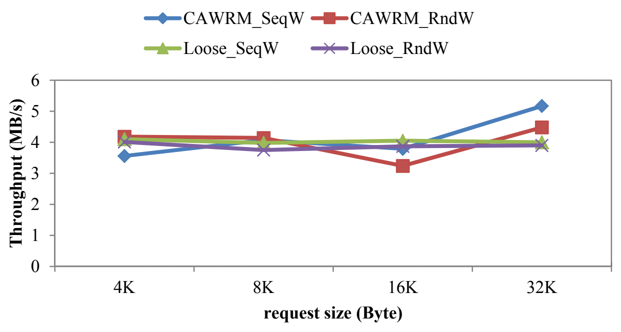


Figure 10. Backup throughput.

implies that the tight coupled architecture does not introduce heavy overhead.

Figure 10 shows the backup throughput. The conclusion is similar as Figure 9.

VI. Conclusion and Future work

The contributions of this paper can be concluded as follows:

- We propose two remote mirroring architectures based on a new kind of volume, AoDI.
- With the help of AoDI's distinct characteristics, the loose coupled architecture improves random write performance and provides the ability construct heterogeneous remote mirroring system at the block level.
- Besides the advantages of the loose coupled architecture, the tight coupled architecture also has advantages: simple logic, unified consistency mechanism and better initialization performance.

Virtualization is a fundamental technology in cloud computing storage system, in which dynamic management of storage resource is indispensable. The tight coupled architecture (CAWRM) can both guarantee the reliability of the data center and adapt to the command of storage management in modern cloud computing system.

Optimizing the sync thread is an important future work. Combining remote mirroring and AoDI with other technique, such as continuous data protection is also planned.

Acknowledgment

This work was supported in part by the National High Technology Research and Development Program of China (2008AA01Z401), NSFC of China (60903028,61070014), Science and Technology Development Plan of Tianjin (08JCYBJC13000), and Key Projects in the Tianjin Science and Technology Pillar Program.

References

[1] H. Weatherspoon, L. Ganesh, T. Marian, M. Balakrishnan, and K. Birman, "Smoke and Mirrors: Reflecting Files at a Geographically Remote Location Without Loss of Performance," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies, FAST 2009*, San Francisco, California, USA, Feb 2009, pp. 211–224.

[2] M. Ji, A. C. Veitch, and J. Wilkes, "Seneca: remote mirroring done write," in *USENIX Annual Technical Conference, General Track*, San Antonio, Texas, USA, 2003, pp. 253–268.

[3] R. H. Patterson, S. Manley, M. Federwisch, D. Hitz, S. Kleiman, and S. Owara, "Snapmirror: File-system-based asynchronous mirroring for disaster recovery," in *USENIX Conference on File and Storage Technologies*, Monterey, California, USA, 2002, pp. 117–129.

[4] "Secure Data Protection With Dot Hills Batch Remote Replication," Dot Hill Corporation, Tech. Rep., Jul 2009.

[5] J. MacCormick, C. A. Thekkath, M. Jager, K. Roomp, L. Zhou, and R. Peterson, "Niobe: A practical replication protocol," *ACM Transactions on Storage*, vol. 3, no. 4, pp. 1–43, 2008.

[6] E. Gabber, J. Fellin, M. Flaster, F. Gu, B. Hillyer, W. T. Ng, B. Özden, and E. A. M. Shriver, "Starfish: highly-available block storage," in *USENIX Annual Technical Conference*, San Antonio, Texas, USA, 2003, pp. 151–163.

[7] R. Cao, C. Zhen, Y. Gao, G. Xu, X. Liu, G. Wang, and G. Xie, "Aodi: An allocation-on-demand incremental volume based on lvm," in *Proceedings of the 26th Symposium On Applied Computing*, TaiChung, Taiwan, 2011, pp. 132–137.

[8] "EMC SRDF - Zero Data Loss Solutions for Extended Distance Replication," EMC Corporation, Tech. Rep. P/N 300-006-714, Apr 2009.

[9] R. Yan, J. Shu, and D. chan Wen, "An implementation of semi-synchronous remote mirroring system for sans," in *Grid and Cooperative Computing Workshops*, Wuhan, China, 2004, pp. 229–237.

[10] "VERITAS Volume Replicator (tm) 3.5 Administrator's Guide (Solaris)," Symantec Corporation, Mountain View, CA, USA, Tech. Rep. 249505, Jun 2002.

[11] P. T. Breuer, A. M. Lopez, and A. G. Ares, "The Network Block Device," *Linux Journal*, vol. 2000, no. 73, 2000.

[12] "LVM," http://sources.redhat.com/lvm/.

[13] "WANemu," http://wanem.sourceforge.net.