MagicStore: A New Out-of-Band Virtualization System in SAN Environments*

Guangyan Zhang, Jiwu Shu, Wei Xue, and Weimin Zheng

Department of Computer Science and Technology,
Tsinghua University, 100084 Beijing, China
zhang-gy04@mails.tsinghua.edu.cn
http://www.cs.tsinghua.edu.cn

Abstract. In this paper, MagicStore, a new out-of-band virtualization system designed for SAN environments is proposed. Online multiplication of the components in a striped volume can help enhance both the I/O performance and storage capacity of a system, but it requires online redistribution of the data on the volume. MagicStore employs a new mapping management solution based on a sliding window to support the online data redistribution without loss of scalability. Furthermore, some virtualization transactions, such as online resizing, require modification of the virtualization metadata, which results in the challenge of keeping the persistent consistency of metadata. MagicStore, by using a combination of ordered writes, REDO logging and log integrity checking, can survive across panics and power failures robustly. In order to support log integrity checking effectively, MagicStore also uses a new log format.

1 Introduction

Storage virtualization [1] can enhance the overall quality of service in storage area networks because it enables the competence of a logical volume to go beyond the limit of single physical storage devices. For example, the online resizing and reconfiguration of logical volumes ensure business continuity. The disk utilization rate can also be increased from only 50% up to 80% through the centralized and more flexible administration of virtualization software [2].

However, an issue facing storage virtualization is the likelihood that it will put an inordinate strain on existing hosts. This concern has led to two schemes for offloading some of the work associated with virtualization: in-band and out-of-band virtualization. The in-band device, which is placed inside the data stream, could itself become a performance bottleneck. Conversely, out-of-band virtualization may provide better scalability because its main function device resides outside the data stream and does not touch the actual data.

In this paper, we propose a new out-of-band virtualization system working in SAN environments called MagicStore. It employs a new mapping management

* This research was supported by the National High-Tech Research and Development Plan of China under Grant No. 2004AA111120 and the National Grand Fundamental Research 973 Program of China under Grant No. 2004CB318205.

solution based on a sliding window. When the data redistribution is not needed, our solution is equal to the mapping function. A sliding window is introduced when the data needs to be redistributed. The solution not only supports online data redistribution but also occupies a small amount of memory space.

Moreover, MagicStore uses a combination of ordered writes, REDO logging and log integrity checking to obtain high persistency. Ordered writes keep the sliding window and physical Extents consistent. REDO logging ensures that the multiple writes to metadata blocks in single virtualization transactions are atomic. And a new log format enables MagicStore to detect whether writing to the log is complete.

The remainder of this paper is organized as follows. Section 2 gives an overview of the MagicStore system. In Section 3, we propose a new mapping management solution based on a sliding window. The strategies for persistent consistency are presented in Section 4. In Section 5, we evaluate the I/O performance of MagicStore through the representative experiments. We conclude with related works and a summary.

2 Overview of the MagicStore System

MagicStore is made up of the manager and the agent software on each host (Figure 1). The manager knows the states of physical devices and manages logical volumes. Instructed by the manager, the agent virtualizes logical volume devices and does the address mapping from the logical address space to the physical address space. Each agent is connected to the manager via TCP/IP.

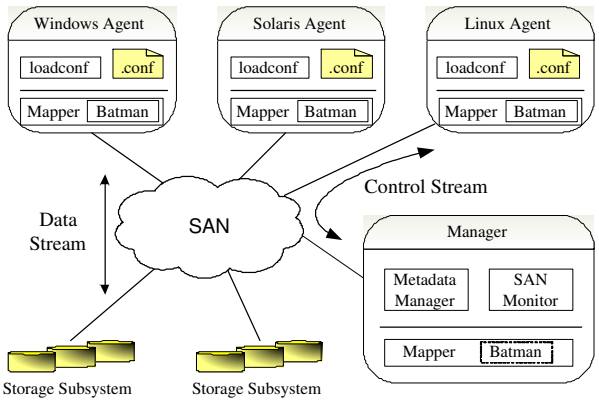

Fig. 1. Architecture of the MagicStore system

The manager consists of two cooperative modules: the metadata manager and the SAN monitor. The metadata manager organizes virtualization metadata using a simple 3-layered model separating physical volumes, volume groups

and logical volumes [3]. Logical volumes may be allocated to hosts with access permissions. Information about the state of the SAN is collected by the SAN Monitor. In addition, the manager enables the applications on itself to access any logical volume by loading the agent on the corresponding platform.

The agent consists of the mapper in the kernel space and the loadconf utility and a configuration file in the user space. The mapper is a light-weight driver residing between the file system driver and the disk driver. When the mapper is loaded, it creates the batman, a kernel thread which receives virtualization instructions from the manager and executes them. The mapper maps the I/O requests sent to logical volumes to the corresponding physical volumes. The loadconf utility is used to ask the mapper to reload the configuration information from the configuration file.

The mapping mode for each logical volume can be alternated between the buffer mode and the non-buffer mode. The former can eliminate the overhead of the network communications for sending frequent mapping requests to the manager. The latter is convenient for online updating of the mapping information.

3 Mapping Management Based on a Sliding Window

To enhance the I/O performance and storage capacity of a system, users often have a reasonable need for increasing the number of components in a striped volume online. It is necessary for the data on the striped volume to be redistributed across the old and new volume components.

The address mapping can be expressed through the mapping function [3,4,5] and the mapping table [1,6] traditionally. The mapping table makes it possible to handle the data redistribution and normal I/O operations at the same time because it can keep track of the movement of data. However, the mapping table occupies a very large space. The transfer and storage of a large amount of mapping information puts tremendous pressure on both the network and the memory, and further impairs the scalability of the whole system.

In contrast to the mapping table, the mapping function which only stores its own function eliminates the transfer and storage of a large amount of mapping information. In this technique, unfortunately, the I/O operation occurring during the data redistribution can not find the correct location of relevant data because the data can exist on the original or new location.

We propose a new solution for managing mapping information. The key idea behind the solution is to introduce the concept of a sliding window into the mapping function. When the data redistribution is not needed, our solution is equal to the mapping function. A sliding window is introduced when the data needs to be redistributed.

Figure 2 illustrates how the metadata is updated when the components in a striped volume are multiplied from 2 to 3. The sliding window is a quite small mapping table which describes the mapping information of a continuous segment of the striped volume. At any time, only data within the range of the sliding window is redistributed. The normal I/O requests to the logical address before

the sliding window are mapped through the original function; those sent to the address after the sliding window are mapped through the new function, and those to the address in the range of the sliding window are mapped through the sliding window. After all the data in the sliding window are moved, the window slides ahead by one window width. The data redistribution of the whole volume is completed when the sliding window reaches the end of the original striped volume. From then on, the address mapping of the whole volume is done through the new mapping function.

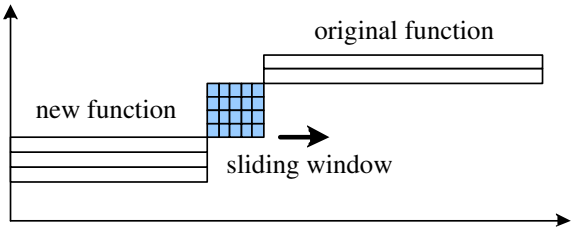

Fig. 2. The components in a striped volume are multiplied from 2 to 3

Introducing the concept of a sliding window enables online redistribution of the data on logical volumes. Additionally, the fact that the size of the sliding window is small and independent of the size of the logical volume contributes to the high performance and scalability of the whole system.

4 Strategies for Persistent Consistency

To enable the out-of-band virtualization system to survive across panics and power failures, virtualization metadata has to be both available and consistent when the system reboots. When the mapping information of a logical volume is modified, the manager asks the mapper to switch the mapping mode of the logical volume to the non-buffer mode. Thus, only the metadata consistency on the manager side has to be ensured.

Whenever online multiplication of the components in a striped volume occurs, we have to keep the sliding window and physical Extents consistent. This consistency can be achieved by the method of ordered writes. The physical Extent is first copied to the new location and then the map block is written to the disk. Even if the power fails in between, just an extent copy is wasted and the consistency is not destroyed. The opposite order is problematic.

Another issue of persistent consistency is that some virtualization transactions write multiple metadata blocks. MagicStore, by using REDO logging, ensures that the multiple writes to metadata blocks in single virtualization transactions are atomic. In this case, intentions are logged first and the metadata updates can be done. In case of a crash, when the manager comes up, it scans through and replays the log. Thus the metadata remains consistent.

A new issue that REDO logging brings is that, in case of a power crash while writing to the log, we must be able to detect that writing to the log is not complete. We propose a new log format, with which MagicStore can detect whether writing to the log is complete by checking the log integrity.

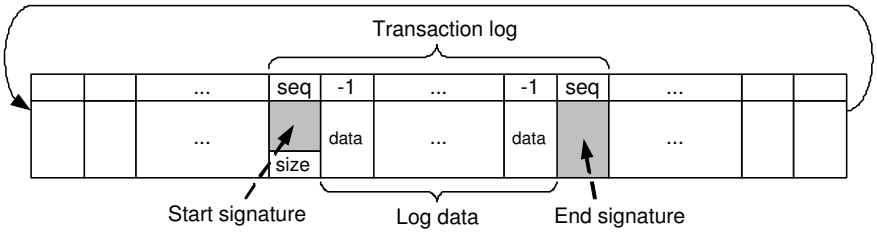

Fig. 3. The new design of the log format

Our new design of the log format is shown in Figure 3. It uses two special signatures to label the beginning and end of the log respectively. The sequence number fields of the start and end signatures store the sequence number of the transaction log, while that of the log data is set to the invalid sequence number value -1. The size of the whole log is recorded at the end of the start signature. This design eliminates the need for scanning through the whole transaction log to find the end signature because the size of the log has been introduced. In addition, there is no possibility of mistaking the old metadata or end signature for the current end signature since the values in their sequence number fields are different.

5 Experiments

The manager was implemented in the user space on the Linux platform. The agent software were implemented on the Windows, Solaris and Linux platforms. In this section, we compare the performance of the linear, striped and mirrored volumes managed by MagicStore with that of the plain volumes managed by the original operating systems.

5.1 Experimental Setup

The Solaris agent was installed on a two-way 300 MHz UltraSPARC-IIi machine with 256 MB of memory and an Emulex LP9802 HBA card running SunOS Release 5.10 Version. Each other subsystem of MagicStore was installed on a two-way 2.4 GHz Intel Xeon machine with 1 GB of memory and an Emulex LP982 HBA card running Linux kernel v.2.4.16 of RedHat 9 distribution or Windows Server 2003. The file systems used were NTFS, UFS and EXT2 respectively. Via a Brocade Silk Worm 3800 fibre channel switch, these machines were connected with an FC disk array controlling five 146 GB Seagate Cheetah 10K disks.

We configured IOmeter[7] to generate the representative workloads, and all of them consisted of 20% writes and 80% reads since Vogels found that 79% of accesses to files were read only [8]. All workloads used random addresses with transfer request size doubled from 8 KB to 4096 KB.

5.2 Results

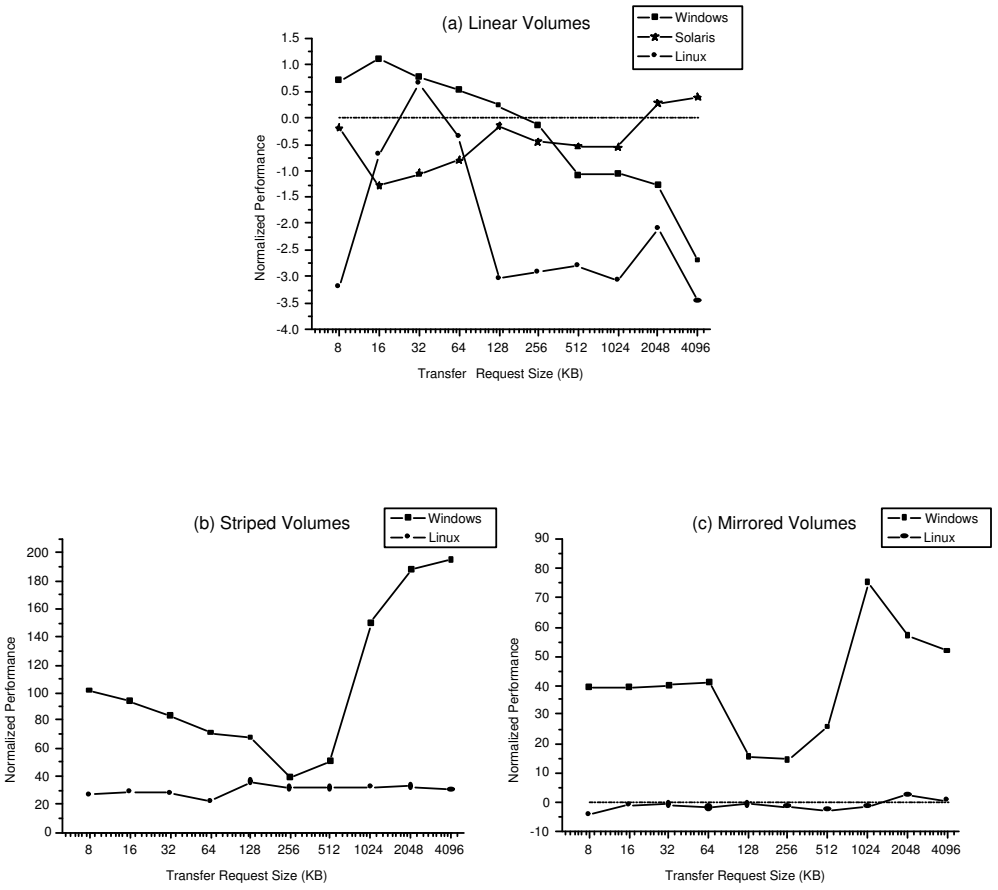

Fig. 4. The Performance of the MagicStore system

We first measured the I/O throughput (MB/s) of each logical volume x managed by MagicStore with different transfer request sizes. According to the following equation, we got its corresponding normalized performance, where Thrput(plain) denotes the I/O throughput of the plain volume, which resides on the same platform with x, with the same transfer request size.

$$Norm\_Perf_x = \frac{Thrput_x - Thrput_{plain}}{Thrput_{plain}} * 100. \quad (1)$$

Figure 4 shows a plot of the normalized performance of logical volumes versus the transfer request size. The minimum, maximum and average normalized performances of linear volumes were respectively, -2.70, 1.10 and -0.29 on the Windows platform, -1.27, 0.39 and -0.43 on the Solaris platform, -3.46, 0.64 and -2.10 on the Linux platform.

All the striped volumes in the experiments were constructed of four FC disks with a stripe width of 64 KB. The minimum, maximum and average normalized performances of striped volumes were respectively 39.16, 194.81 and 103.89 on the Windows platform, 21.94, 36.21 and 30.03 on the Linux platform.

In the experiments, all the mirrored volumes were constructed of two components. Writes to mirrored volumes were initiated concurrently and reads were alternated between the copies. The minimum, maximum and average normalized performances of mirrored volumes were respectively, 14.47, 75.25 and 40.11 on the Windows platform, -4.16, 2.46 and -0.98 on the Linux platform.

6 Related Works

In recent years, considerable attention has been paid to the storage virtualization systems for SAN environments. Some of them, such as the Pool Driver [4], CLVM[5] and the SANtopia volume manager[1], employ symmetric architecture. This means that they only apply to clusters running a single operating system.

There are also some systems which use asymmetric architecture. However, they all have some limitations. For example, OpenView[9] only applies to the specified HBA card and driver because its agent is implemented on the HBA driver. When the SANfs-VM[6] or V:drive[2] is used, only Linux can be run on the hosts.

Among all the above systems, only the SANtopia volume manager supports online multiplication of the components in a striped volume. Unfortunately, the mapping management using a mapping table restricts its scalability and makes it inadequate for SAN environments with a large amount of storage. Jose and Toni proposed an algorithm for increasing the capacity of RAID5 [10], which has an easily controlled overhead. A similarity between the algorithm and our solution is that the new disks are gradually available to serve requests during the multiplication process.

Reference [11] presents a log format for detecting whether writing to the log is complete. However, it has no capability to tell log data blocks from the transaction epilogue block belonging to the same transaction log by their transaction ids and offsets. Furthermore, without introducing the size of the log, this solution makes it necessary to scan through the whole transaction log to find the transaction epilogue. If some data block of the transaction log exactly matches the current transaction epilogue, a checking mistake will appear.

7 Conclusions

MagicStore employs a new mapping management solution based on a sliding window. This solution enables it to support online multiplication of the components in a striped volume. Furthermore, it contributes to MagicStore's high scalability since it occupies a very small space. By employing a combination of ordered writes, REDO logging and log integrity checking, MagicStore can survive across panics and power failures robustly. Moreover, a new log format effectively supports log integrity checking. In the representative experiments, MagicStore demonstrated its ability to provide high performance.

References

1. Chang-Soo Kim, Gyoung-Bae Kim, Bum-Joo Shin. Volume Management in SAN Environ-ment. In: Proceedings of the 8th International Conference on Parallel and Distributed Sys-tems, ICPADS 2001. 2001. pages 500-505.
2. A. Brinkmann, M. Heidebuer, F. Meyer auf der Heide, et al. V:Drive - Costs and Benefits of an Out-of-Band Storage Virtualization System. In: Proceedings of the 12th NASA God-dard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST), pages 153-157, College Park, Maryland, USA, 13-16 Apr. 2004.
3. David Teigland, Heinz Mauelshagen. Volume Managers in Linux. In: Proceedings of the 2001 USENIX Annual Technical Conference, pages 185-198, June 2001.
4. David Teignald. The Pool Driver: A Volume Driver for SANs, In Partial of Fulfillment of the Requirements for the Degree of Master of Science, Oct 1999.
5. Heinz Mauelshagen. Linux Cluster Logical Volume Manager, In: Proceedings of the 11th International Linux System Technology Conference. Erlangen, Germany. Sept. 2004.
6. Seung-Ho Lim, Joo Young Hwang, Kyung Ho Kim, et al. Resource Volume Management for Shared File System in SAN Environment. In: Proceedings of the 16th International Con-ference on Parallel and Distributed Computing Systems (PDCS), 2003.
7. Intel Corporation, Iometer, July, 2004. http://www.iometer.org.
8. W. Vogels. File system usage in Windows NT 4.0. In Proceedings of the 17th ACM Sympo-sium on Operating Systems Principles, pages. 93-109, Dec. 1999.
9. Hewlett-Packard Development Company. HP OpenView Storage Operations Manager v1.2. Sept. 2004. http://h18006.www1.hp.com/products/quickspecs/11778_div/11778_div.html.
10. Jose Luis Gonzalez and Toni Cortes. Increasing the capacity of RAID5 by online gradual assimilation. International Workshop on Storage Network Architecture and Parallel I/Os. Antibes Juan-les-pins, France, September 30, 2004
11. Suresh B Siddha, K Gopinath. A Persistent Snapshot Device Driver for Linux. In: Proceed-ings of 5th Annual Linux Showcase & Conference, 2001.