Design and Implementation of a SAN Agent for Windows NT Architecture*

Ran Meng, Jiwu Shu, and Wei Xue

Department of Computer Science and Technology,
Tsinghua University, 100084 Beijing, China
mengran@tsinghua.org.cn
http://www.cs.tsinghua.edu.cn

Abstract. In an out-of-band SAN virtualization system, the virtualization appliance maintains metadata, and the agents inside the kernel of servers use that data to supply virtual storage devices and to perform the mapping of I/O address. A design of an out-of-band SAN virtualization system based on Windows NT volume manager driver, and its underlining technologies were presented in this paper. It shows that, in general our system is able to supply large volume and high bandwidth virtual storage devices for applications, and it can be used as a basic environment to manage the SAN centrally. The system performance was investigated in comparison with a plain SAN under FAT32 and NTFS, using different data block sizes and access patterns. The results reveal that the overhead induced by our approach is much low. Under FAT32, the performance characteristics of the 3-striped virtual volume follow a typical strip distribution strategy and the bandwidth is 1.20 3.71 times greater than general volume. Furthermore, under NTFS, the bandwidth of the 3-striped virtual volume is an average of 4.10 (max 4.82) times greater than general volume with the random read access test. Hence it can be concluded that our virtualization approach could make use of the storage resources in SAN more effectively.

1 Introduction

According to the definition given by the Storage Network Industry Association (SNIA) [11], storage virtualization is "an abstraction of storage that separates the host view from the storage system implementation." Another more detailed definition of virtualization from Robert Frances Group is "Those architectures and products designed to emulate a physical device where the characteristics of the emulated device are mapped over another physical device" [1]. Virtualization provides many benefits to the SAN system, including:

* The work described in this paper was supported by the National Natural Science Foundation of China under Grant No.60473101, the National Key Basic Research and Development 973 Program of China under Grant No. 2004AA111120 and the National High-Tech Research and Development 863 Plan of China under Grant No. 2004AA111120.

- The manpower needed to administer the SAN system is significantly reduced [1],
- The capacity of the virtual storage device is not limited by a single disk or a single RAID system [2],
- The virtualization can dramatically improve the utilization of storage resources [2].

The virtualization techniques inside a SAN system can be classified as two types in terms of their control path: in-band and out-of-band [3] [11]. In an in-band system the virtualization appliance is in the data path and virtualization functions such as address mapping are accomplished by the same component that performs reading and writing. In an out-of-band system the implementation related to the virtualization is not in the data path. Nowadays, there are already some SAN virtualization solutions, such as HP OpenView Storage Operations Manager [8]. In these solutions the storage management tasks ship with an integrated HBA (Host Bus Adapter). Therefore they are low-level implementations and some special HBAs and corresponding drivers and/or firmware are required.

This paper presents a design of an out-of-band SAN virtualization system based on the Windows NT volume manager. In this design we implement an agent to perform virtualization tasks at the Windows NT volume manager level; thus only standard HBAs are required. Additionally in our system there is a virtualization server centrally managing the storage resources. In section 3 the design of the virtualization agent based on the Windows NT volume manager are presented, and the key techniques we used are described. Furthermore we investigate the performance of our system compared with a plain SAN under FAT32 and NTFS with different data block sizes and access patterns and focus on the analysis of the virtual volume using stripe distribution strategy.

2 The Virtualization Server

In our design the virtualization server consists of five cooperative components: management module, interface module, communication module, device-monitor module and device-agent module. Here it is incumbent on the management module to maintain the virtualization metadata, to prompt and harmonize other components and the virtualization agents running on the application servers, and then to achieve the virtualization of the storage resources of the entire SAN system. The virtualization metadata is stored in a specific location in multiple devices in the storage pool. Each device has a complete copy and backs up the metadata for others.

3 Design and Implementation of the Virtualization Agent on Windows Platform

The virtualization agent running on Windows application servers is comprised of the kernel module and the communication module. The former performs vir-

tualization functions, for example, creating virtual volumes, translating the I/O address from virtual volumes to physical devices, etc. The latter communicates with the virtualization server and assists the kernel module to accomplish the virtualization tasks. In this section, we first introduce the driver model of storage devices in the Windows NT architecture. Then the design and implementation of the Windows virtualization agent will be discussed.

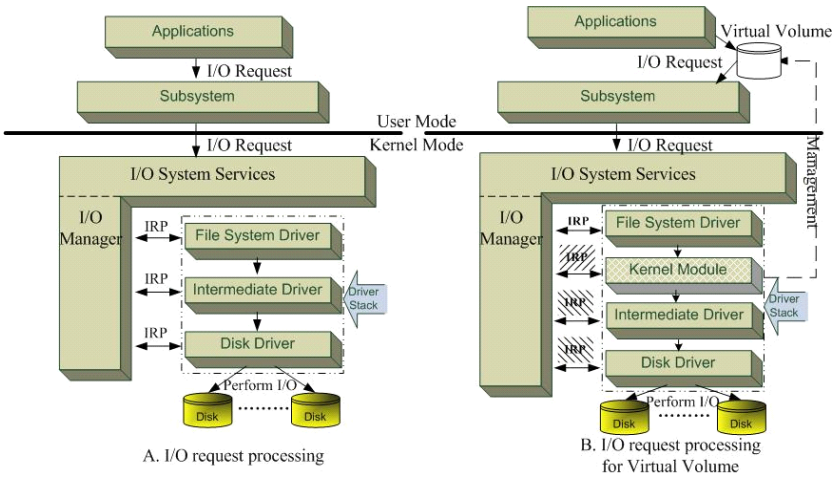

Fig. 1. Windows NT I/O driver stack model. The (*I/O Manager*) of the Windows NT operating system handles the flow of data to and from peripheral devices. It exports an (*I/O system services*) whose user-mode protected subsystem supplies a programming. (*Applications*) use that interface to manipulate the devices such as send I/O requests to them, so that the (*I/O Manager*) can intercept all those requests

3.1 The Windows NT Disk Driver Model

As shown is Figure 1, The I/O Manager provides a consistent interface for all kernel-mode drivers (including the lowest drivers, intermediate drivers and file system drivers (FSD)). All I/O requests sent to these drivers are represented as I/O request packets (IRPs). These drivers are called as driver stack in terms of the processing of IRPs. The driver (usually the FSD) that is situated on the top of the driver stack will process IRPs first, and then passes them to the next driver via the I/O Manager. [10] [12]

3.2 The Virtualization Agent

The communication module of our agent is a user-mode network program that uses TCP/IP over the Ethernet to talk with the virtualization server. It receives

the metadata and instructions from the virtualization server, orders the kernel module to perform virtualization tasks and sends the results or other information back. The kernel module is represented as a volume manager driver inside the

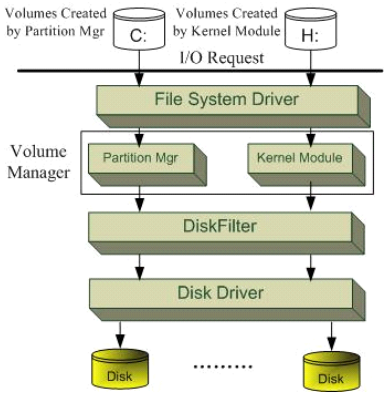

Fig. 2. The relation between the virtual volumes and the Windows system volumes and the relation between the (*kernel module*) and the (*Volume Manager*) are illustrated. The (*Disk Filter*) is an upper filter driver that hides the physical storage devices from accesses other than from the kernel module

Windows kernel. Its functions include:

- Providing the virtual volumes for the applications in user mode, and maintaining and managing them,
- Distributing the data of virtual volumes between physical devices and mapping the I/O requests to them,
- Recording some statistics of virtual volumes and sending them to the virtualization server for further analysis.

Now we discuss the design of the communication module and the kernel module.

IRP Processing in Kernel Module. As viewed from the perspective of applications and file systems, the virtual volumes are the same as other volumes that are created by the Windows Disk Manager, so that they can access these volumes in the same way. As an example, the IRP representing a read request from an application to the virtual volume is viewed as a request to read a specific range of data on the virtual volume when it has been processed by the FSD. So the kernel module suspends the original IRP first, and then creates one or more IRPs to read the corresponding data from physical devices according to the distribution strategy. When those IRPs are completed by the disk driver and popped to the kernel module, the original IRP will be filled with the data they read and will be completed by the kernel module.

Distribution Strategy and Metadata Management. At the present time the kernel mode supports linear and strip distribution strategy. In linear strategy, data is continuously distributed on one or more physical devices. The strip strategy repeatedly places data among multiple devices according to a specific chunk size. In this way the reads and writes are done in parallel on the devices and the performance will improve.

In our design the metadata is centrally managed and distributively applied. While creating the virtual volumes the metadata containing distribution information is sent to the kernel module by the management module on the virtualization server. The kernel module stores this metadata in a kernel memory area that is associated with the volume device object. This metadata is a shared portion of all virtualization information and assists the kernel module in performing address mapping. If some settings are changed, updated metadata will be sent to the kernel module, which may perform some tasks such as data migration.

Table 1. The configuration of the application server

Processor	Intel®Xeon 2.4GHz × 2
Memory	1G
OS	Windows 2003 Server
FC HBA	Emulex LP982(2Gb/s)

4 Results and Analysis

In this section we will give the experimental performance results of the virtual volumes on Windows servers in our out-of-band SAN system. The test system consisted of an application server for which the OS was Windows 2003 Server Enterprise Edition, a specialized virtualization server and a Fiber Channel (FC) disk array. These components were connected through an FC network. The configuration of the application server and the FC disks are listed in Table 1 and 2. We investigated the performance under FAT32 and NTFS separately. The test tool we used was the IOMeter [14].

4.1 Results under FAT32

We compared the performance of our virtualization implementation to a plain SAN system. For this our test cases include:

- LV0: A virtual volume with 3 stripes and a chunk size of 64KB,
- LV1: A virtual volume using the continuous distribution strategy,
- PV0: A Windows system volume.

Table 2. The parameter of the FC disks

Series	Seagate Cheetah 10K
Capacity	147G
Speed	10000 RPM
Cache Size	8M
Max Bandwidth	105MB/S

The file system was FAT32 and the capacity was 10G for all the volumes. We tested their I/O bandwidth with 4 access patterns: sequential read, sequential write, random read and random write. The results showed that the contrastive characteristics of the three volumes were familiar for these four patterns. We can safely draw some conclusions from the results.

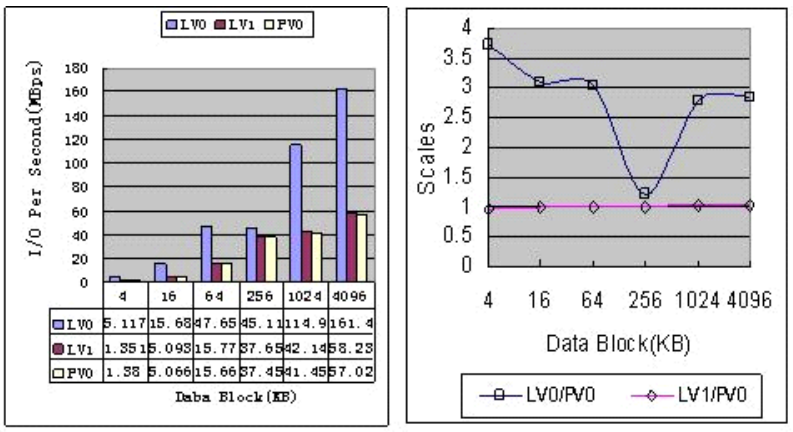

Fig. 3. Comparisons of the random read under FAT32

- The bandwidth of LV1 was almost the same as that of PV0. This indicates that the overhead induced by the kernel module is quite low in the linear strategy,
- LV0 showed performance characteristics typical of strip data distribution. It gained an average 3.09 (max 3.71) times bandwidth greater than PV0 when the data block size was considerably less or greater than the product of the chunk size and the number of stripes (64KB * 3 in this case). When the data block size was comparable to 64KB * 3, LV0 showed a bandwidth similar to PV0,
- The ratio of LV0 and PV0 decreased in the endmost part of the curve. This may have resulted from the excessive amount of concurrent IRPs.

4.2 Results under NTFS

The NTFS performance has been optimized in many aspects [10] [13], with the result that most Windows application servers format their volumes with NTFS. We repeated those tests described above for NTFS. Other parameters of LV0, LV1 and PV0 were the same but their file systems were changed to NTFS. The results showed that:

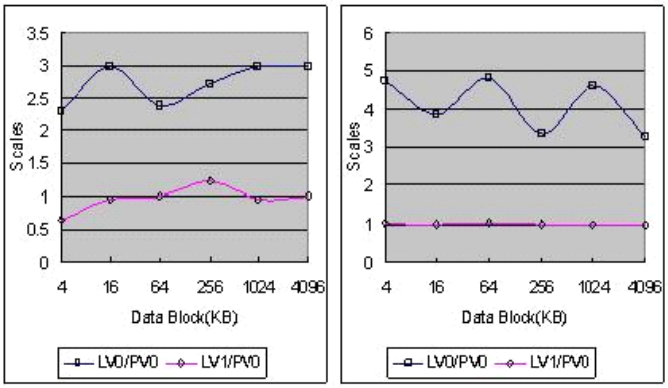

Fig. 4. Comparisons of the sequential read (*left one*) and random read (*right one*) under NTFS

- The performance of LV1 and PV0 was quite similar; it showed that the overhead involved by the kernel module was fairly low,
- LV0 showed an analogical strip performance characteristics in the sequential and random write patterns,
- LV0 showed a much better performance in the sequential and random read patterns. LV0 gained an average 2.73 (max 2.99) times bandwidth compared to PV0 in the sequential read pattern, and an average 4.10 (max 4.82) times in the random read pattern.

Under the NTFS the virtual volume gained much higher bandwidth than the general volumes in read accesses irrespective of the data block size. This shows that our virtualization approach utilizes the SAN storage resource more effectively, especially with the NTFS.

5 Conclusions

A design of an out-of-band SAN virtualization system based on Windows NT volume manager driver, and its underlining technologies were presented in this paper. Virtual volumes can be provided to the applications and end users by the virtualization agent implemented as a Windows volume manager driver with a

virtualization server. The agent maps the I/O accesses to the virtual volumes to physical devices and separates the physical devices from the application servers. Because the virtualizations are done at the volume manager level, only standard HBAs and corresponding drivers are required.

Generally our system is able to supply large volume and high bandwidth virtual storage devices for applications, and it can be used as a basic environment to manage SAN.

The system performance was investigated in comparison with a plain SAN under NTFS with different data block sizes and access patterns. The results showed that the utilization of the SAN storage resource could be increased significantly with our virtualization approach, particularly with the strip distribution strategy and the NTFS.

References

1. Charles Milligan, Sid Selkirk. Online Storage Virtualization: The key to managing the data explosion, Proceedings of the 35th Hawaii International Conference on System Sciences 2002.
2. Andre Brinkmann, Michael Heidebuer. V:Drive-Costs and Benefits of an Out-of-Band Storage Virtualization System, In Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST), College Park, Maryland, USA, 13 - 16 April 2004.
3. J.S.Glider, C.F.Fuente, W.J.Scales.The Software Architecture of a SAN Storage Control System. IBM SYSTEMS Journal VOL 42, NO 2, 2003.
4. Andre Brinkmann, Kay Salzwedel, Christian Scheideler. Compact, Adaptive Placement Schemes for NonUniform Distribution Requirements, Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures, Winnipeg, Manitoba, Canada, 2002.
5. Ismail Ari, Melanie Gottwals, Dick Henze, SANBoost: Automated SAN-Level Caching in Storage Area Networks, 13th IEEE International Conference on Autonomic Computing (ICAC'04).
6. Han Deok Lee, Young Jin Nam. Regulating I/O Performance of Shared Storage with a Control Theoretical Approach. Proceedings of the 21st IEEE Mass Storage Systems Symposium/12th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST2004), April 2004.
7. Robert Gramacy, Manfred Warmuth, Scott Brandt and Ismail Ari, Adaptive Caching By Refetching, 2002 Neural Information Processing Systems (NIPS'02).
8. http://h18006.www1.hp.com/products/storage/software/som/index.html, HP Open View Storage Operations Manager.
9. M. Farley. Building storage area networks McGraw-Hill, 2000
10. Microsoft Development Network, http://msdn.microsoft.com/.
11. Storage Networking Industry Association, http://www.snia.org/.
12. William J. Bolosky, Scott Corbin, David Goebel, and John R. Douceur, Microsoft Research Abstract, Single Instance Storage in Windows 2000, August 2000.
13. L. Chung, Windows 2000 Disk IO Performance, MS-TR-2000-55, June
14. IoMeter Project, http://sourceforge.net/projects/iometer/