Journal of Southeast University (English Edition) Vol.20 No.1 Mar. 2004 ISSN 1003—7985

Design and implementation of a new special storage server

Han Dezhi Xie Changsheng Fu Xianglin Liu Chun

(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: To improve I/O speed and system performance for network storage devices, we have designed a new special storage server that is an iSCSI-based network-attached storage server (iSCSI-based network-attached storge server, for short iNAS). Firstly, the iNAS can provide both the file I/O and the block I/O services by an iSCSI module, and it converges with the NAS and the SAN (storage area network). Secondly, the iNAS greatly improves the I/O speed by the direct data access (zero copy) between the RAID (redundant array of inexpensive disks) controller and the user-level memory. Thirdly, the iNAS integrates the multi-RAID for a single storage pool by a multi-stage stripping device driver, and it implements the storage virtualization. In the experiments, the iNAS has ultra-high-throughput for both the file I/O requests and the block I/O requests.

Key words: network-attached storage; storage area network; iSCSI; zero copy; multi-stage stripping; Linux file system

With the digital information network increasing at a tremendous speed, more and more efforts have been made to improve the traditional network-attached storage (NAS). *The Storage Networking Industry Association (SNIA) Technical Dictionary* defines NAS as: a term used to refer to storage elements that connect to a network and provide file access services to computer systems. In common usage, an NAS system is a special-purpose device designed to serve files to clients over a LAN. There are many advantages in the architecture, such as sharing files under a hetero-architecture, making use of the existing LAN architecture, easy installation, operation and management, PNP, good connection compatibility and network adaptation, low costs and so on[1].

Nevertheless, in applications, there are some disadvantages to the NAS device. ① The NAS device does not support the block I/O protocol except for the network file system (NFS) and the common Internet file system (CIFS)[1], and the access for the block I/O user is limited. ② In accessing speed, when the NAS reads some data, firstly, the data is transferred between various I/O devices and the kernel-level buffer cache by the DMA (direct memory access),then the kernel-level buffer cache is used as a temporary buffer cache that is always mapped into the physical memory. The data transfer between the kernel-level buffer cache and the user-level memory are executed by the CPU's memory-to-memory copy. When any data are written, they go through a reverse process. In this case, the I/O performance is limited by the speed of the memory-to-memory copy[2]. ③ Regarding data backup, the data transfer in the NAS system would engross so much of the LAN bandwidth that precious network resources may be wasted. Sometimes this status influences users' applications. ④ With respect to resources integration and the NAS system management, the NAS can only integrate the disk resources in a single NAS device, and cannot span different NAS devices. So it is difficult to converge several different NAS devices to a large NAS device; the system has to manage those devices individually.

To improve the situation, this paper describes an NAS server system based on the iSCSI protocol (iSCSI-based NAS server, for short iNAS).

1 Design Mechanism

This paper proposes three mechanisms to solve the problems of the common NAS. One is a zero copy mechanism to realize direct data transfer between the RAID controller and the user-level memory; the second provides the block I/O service and the file I/O service at the same time by an iSCSI module; the third realizes storage virtualization by an MSS (multi-stage stripping) device driver.

1.1 Design strategy

The design strategy in our system is described as follows.

1) The iNAS can offer block I/O services by the iSCSI modular including a client iSCSI module and a server iSCSI module[3].

2) The direct data transfer between the RAID

Received 2003-07-10.

Foundation items: The National Natural Science Foundation of China (No.60173043) and the State Key Basic Research and Plan Program (973 Program) (No.G1999033006).

Biographies: Han Dezhi (1966—), male, graduate; Xie Changsheng (corresponding author), male, doctor, professor, ccsxie@263.net.

controller and the user memory should be implemented, instead of the buffer cache. It improves the file I/O speed of the iNAS system with the zero copy technique[2].

3) By developing a multi-stage stripping device driver, the iNAS can be mounted to the multi-RAID simultaneously, and the multi-RAID can be virtualized at high-speed large RAID[4,5].

4) To improve the read/write performance of the disk, a disk fragment clear-up program has been developed. When the system isn't working, this program can clear up the whole disk and try to store the blocks of each file in a contiguous disk block[6].

5) We have revised the Linux system and kept its network and storage functions and some hardware concerned. So the iNAS is a black casket from its outside.

1.2 Operational system choice

Because source codes for the Linux are open to all users, we may modify them according to our requirements. To reduce the system run overhead, on the one hand, we have simplified the file system, the network protocol, and the system services, etc; on the other hand, we have appended our product with a high-speed cache, an intelligent pre-fetch, a load balance, and a distributing file system, etc., so the performance of the system is greatly improved and a storage-oriented and embedded system integrating many new techniques is achieved.

The Linux uses the kernel-level buffer cache mechanism as the PC-Unix[5], which mediates the data transfer between the disk I/O system and the user's process. The kernel-level buffer cache is always used, because the Linux has two process modes, i.e., the kernel mode and the user mode. In data transfer between various I/O devices and the kernel-level buffer cache by the DMA, the kernel-level buffer cache is used as a temporary buffer that is always mapped into the physical memory. The data transfer between the kernel-level buffer cache and the user-level memory is executed by the CPU's memory-to-memory copy. The reason for this transfer sequence is that the user's memory might be in the "exchange" state. Although the data transfer between the kernel-level buffer cache and the I/O devices is by the DMA, the DMA is transferred between the physical memory areas. And the data transfer between the kernel-level buffer cache and the user-level memory is executed as a software process, which is a data transfer between virtual memories. The memory-to-memory copy would limit the I/O performance of the system. To solve this problem, we have introduced the zero copy mechanism[2].

1.3 iSCSI protocol

The iSCSI protocol defines a mapping from the SCSI to the TCP/IP; namely it packs the host SCSI commands to the IP packets and transports them over the IP network[3]. When the packets arrive at the destination node, they revert to the SCSI commands, consequently, a direct and transparent transport process for the SCSI command over the IP network is realized. It integrates the SCSI and the TCP/IP protocols, and realizes a non-slot connection with the storage system and the network.

Nowadays there are three ways to realize iSCSI[7]. The iNAS implements the iSCSI function by the pure software. The iSCSI software modular includes the initiator modular, which is located at the client, and the target modular, which is located at the server. Both of them are loaded to the OS as kernel state drivers. The initiator is responsible for intercepting and capturing the I/O requirements handed down by the file system, and transforms them into the iSCSI data units, then sends them via the network interface card. The target is responsible for resuming the SCSI commands and delivers them to the SCSI device according to the information of the iSCSI protocol data units. After loading the initiator modular at the client, there will appear a device named/dev/sd*, which can be directly mounted to the system.

1.4 Zero copy mechanism

The conventional and proposed processing stacks are shown in Fig. 1, and the traditional read information transfer flows are shown in Fig. 1 (a). When data and commands are transported between the RAID controller memory and the user memory, the kernel-level buffer cache acts as a temporary buffer cache. Reading proceeds in two steps: the DMA transfer from the RAID controller to the kernel-level buffer cache, and the memory-to-memory copy from the kernel-level buffer cache to the user-level memory.

As for the proposed processing stack, see Fig. 1 (b). To avoid using the buffer cache, a new data flow and a command flow are established in both the Linux and the device driver. In this way, the LFS (Linux file system) can directly access the device driver via the zero copy entry, and send read/write commands, including the initial sector number, the sector count and the virtual memory addresses (user memory addresses). We call the command request and the data transfer via this flow the zero copy mechanism.

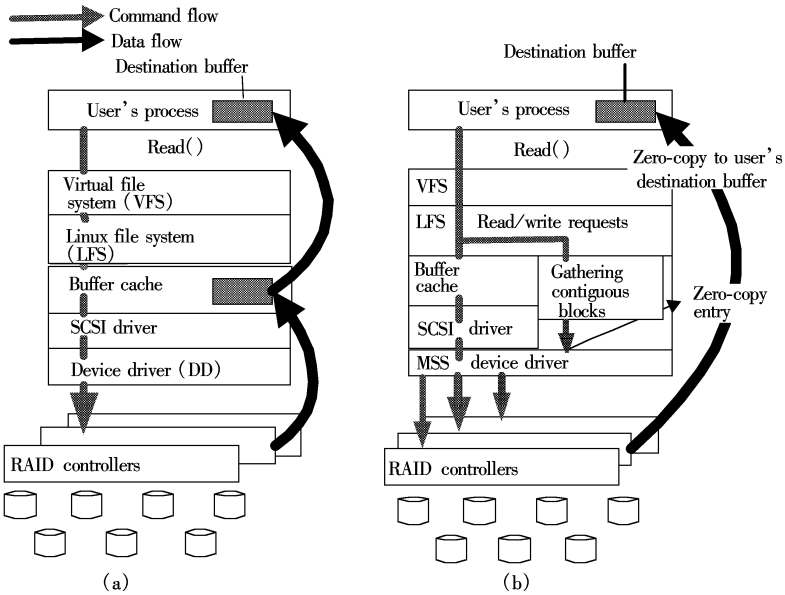

Fig.1 Traditional file server stack and newly-designed special storage server stack. (a) Traditional file server stack; (b) Newly-designed special storage server stack

In the iNAS, when the LFS receives a read/write request, in case it is a file request, the server will find out the distributed blocks and combine all adjacent blocks together as a large block, then transport it directly between the user memory and the RAID I/O. If the LFS receives an iSCSI block request, because the iSCSI protocol does not support the zero copy, the data are transmitted by the conventional flow via the buffer cache. For other file operations, such as creating a directory or accessing the disk information, **the conventional flow via the buffer cache is used.**

1.5 Multi-stage stripping device driver

In the proposed system, the Linux device driver for the RAID controller PCI cards has two special functions. One is the storage virtualization function, offering an access to two or more RAID controllers, and constructs a large MSS RAID device with high-speed performance. The other is the zero copy mechanism.

In the zero copy mechanism, the LFS directly sends the read/write command to the MSS device driver. The command includes the buffer cache address, in addition to the starting sector number and the number of sectors. The device driver should convert the address from virtual form to physical one, confirming that the page tables are contiguous or not and scattering contiguous pages and command request to the RAID controllers with a chaining DMA function to reduce the overhead of the scattering pages.

Although the iSCSI protocol doesn't support the zero copy and the block I/O client can only use the MSS functions, the iSCSI is better than the traditional file transfer protocol in terms of flow and congestion control, discovery mechanism, timeout and retranslation, etc., and it can respond to the block I/O **requests of clients rapidly.**

1.6 Disk fragment clear-up program

To enhance the iNAS performance, reading or writing a file would run best when it is in contiguous disk blocks. So we develop a program to clear up disk fragment on schedule to assure that the blocks of a **file can be stored at contiguous sectors.**

2 Prototype System

2.1 Hardware

The specifications for the iNAS prototype are shown in Tab. 1. It has a 1 GB memory, 3 RAID controllers, each of which has two hard disks, a Pentium 4 (1 500 MHz) CPU, Redhat version 7.1 used as the Linux. In the Elstorage-2000, there is an RAID control card with two SCSI hard disk interfaces, which is developed by Elstorage Corporation and our **laboratory together.**

Tab.1 iNAS hardware configuration

Processor	Pentium 4, 1 500 MHz
Host memory	1 GB interleaved EDO-DRAM
PCI buses	2 PCI buses with 3 slots each
RAID controllers	Elstorage-2000
RAID level	RAID0, RAID1 and RAID5 supported
#Controllers	3
Hard drive	IBM 60 GB
#Drives	6 (2 drives at each controller)
OS	Linux 7.1

2.2 Software architecture

The iNAS client software architecture is shown in Fig.2. When the iNAS offers the block I/O services, it uses the iSCSI technology, as shown in Fig.2 (client 1). Concrete data read/write flow is: ① The block I/O commands (SCSI commands) sent by the application in client 1 are encapsulated to the IP data packets via the iSCSI device driver and the TCP/IP protocol stack, then transferred over the IP network; ② When the encapsulated packets arrive at the iNAS, they are restored to the original SCSI commands via an unpacking process, then the iNAS sends these SCSI commands to the storage device as the block read/write requests; ③ Those needed data blocks return to the iNAS from the RAID; ④ Data blocks are encapsulated by the TCP/IP protocol stack in the iNAS, and transmitted to the client 1 over the IP network, then the data blocks are unpacked and assembled for a file in the client 1 file system.

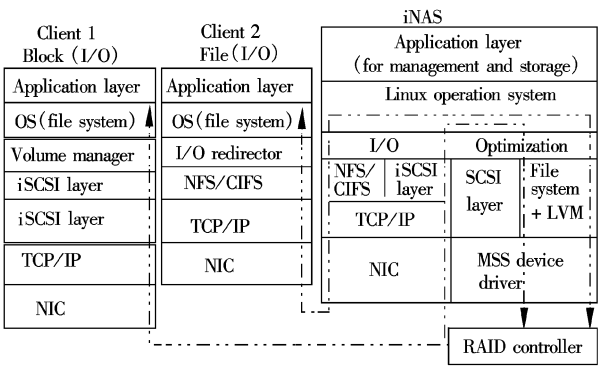

Fig.2 Communications software architecture of the iNAS and the client

When the iNAS offers the file I/O services, as shown in Fig.2 (client 2), the read/write process is as follows: ① The client 2 sends the file read/write requests to the iNAS (its working manner is the same way as that of the traditional NAS); ② The file read/write requests are converted to the block I/O requests by the iNAS file system, then transmitted to the RAID controller by an MSS driver; ③ The RAID controller directly sends corresponding data blocks to the user memory by the zero copy; ④ The data blocks are assembled to files by the iNAS file system, then the iNAS transfers the file to the client 2 via the network. In this case, the iNAS is connected with the network in the NAS way, working the same way as the traditional NAS, thus it has such advantages as those of the traditional NAS and can allow users to access the hetero-architecture files and share them. Via the IP network, it can make full use of the existing network and has good connectivity, adapting to a complicated network environment. On the other hand, the iNAS supports the iSCSI block I/O protocol. So it has high scalability (we can put many iNAS devices in a local network) and high availability by RAID, and it can centralize uniform storage resources and manage them, i.e., merge the multi-iNAS devices to a large uniform iNAS device.

3 Experimental Results

For storage networks, the most important performance parameters are average response time and network throughput. In this paper we use them to evaluate the iNAS. Average response time is the average time elapsed since the I/O requests are sent by the initial device until the I/O operations are finished by the target device and the finishing message is sent back to the initial device by the target device. Throughput is the maximum amount of I/O request processed by the storage subsystem per time unit. The average response time and throughput can be measured from different views, for example, the user, OS, and disk controller and so on. For simplicity, in our experiments, the two parameters are measured in terms of the OS of the server.

3.1 Methodology

To test the iNAS performance, we have carefully designed two groups of experimental environments. Fig.3 gives a highly simplified structure of the experiments; two groups of experimental configurations are shown in Tab.2. Host 1 is a block I/O request client, host 2 is a file I/O request client. The iNAS is a special storage server that we designed.

Tab.2 Configuration of experimental machine

Device	CPU	Memory	OS	Hard disk	NIC/HBA
Host 1	Pentium 3, 730 MHz	256 MB	Linux 7.1	ST318437LW (SCSI)	AGE-1000SX (NIC)
Host 2	Pentium 3, 730 MHz	256 MB	Linux 7.1	Maxtor 91020D6 (IDE)	AGE-1000SX (NIC)
iNAS	Pentium 4, 1 500 MHz	1 GB	Linux 7.1	IBM 60 GB	AGE-1000SX (NIC)

The block I/O experiment is shown in Fig.3(a); the user can send the block I/O request from host 1 to the iNAS via the iSCSI modular. The file I/O experiment is shown in Fig.3(b); the user can send the file I/O request from host 2 to the iNAS via the NFS.

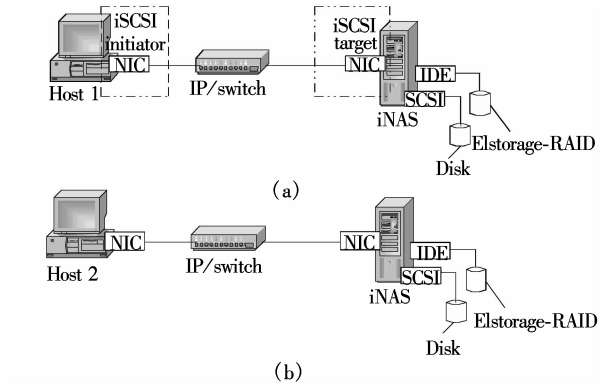

Fig. 3 Experimental configuration. (a) The block I/O experiment; (b) The file I/O experiment

In the experimental environments described in Fig.3, we have done three groups of experiments to measure the iNAS performance. The block I/O performance is tested via Fig.3(a) by the IO meter in the first experiment; the file I/O performance is tested via Fig.3(b) by the IO meter in the second; in the third, the performance of the iNAS file system is tested by a **benchmark program that we designed.**

3.2 Results and discussions

The testing results for the block I/O request and the file I/O request are shown in Tab.3. Fig.4 shows the effect of the file/block size on the throughput and the mean response time. As shown in Fig.4(a) and Fig. 4 (b), when the file/block size is increased, the throughput (MB/s) is increased, but the throughput (IO/s) is decreased. Most importantly, the throughput of the block I/O is bigger than that of the file I/O. As observed in Fig. 4 (c), when the file/block size is increased, the mean response time is increased. As shown in Fig. 4, there is no performance difference between the block I/O requests and the file I/O requests, when the block I/O requests are processed by the iSCSI module and the file requests are **processed by the optimized file system.**

Tab.3 Experimental results

File, block size/KB	Throughput/(IO · s^{-1})		Throughput/(MB · s^{-1})		Average response time/ms	
	File	Block	File	Block	File	Block
1	1 253.316 2	1 539.891	1.223 94	1.503 81	3.981 48	3.643 28
2	1 153.537 5	1 385.011	2.253 00	2.705 23	3.853 84	3.739 62
4	1 099.799	1 202.381	4.296 0	4.696 81	4.474 65	4.246 59
8	806.671 4	941.427 2	6.302 1	7.354 29	4.868 74	4.704 26
16	459.650 3	583.033 6	7.182 0	9.140 99	5.602 83	5.401 48
32	351.385 6	380.147 2	10.980 8	11.879 46	6.832 74	6.501 05
64	234.016 6	254.009 6	14.626 0	15.875 56	8.538 46	8.058 08
128	147.821 23	150.270 4	18.477 6	18.783 78	11.432 94	10.214 78
256	96.777 41	101.103 2	24.194 35	25.275 88	15.562 35	13.053 59
512	56.225 03	57.742 8	28.112 51	28.871 54	19.947 623	17.902 52
1 024	30.007 99	31.812 4	30.007 99	31.812 39	33.342 89	30.814 66

We evaluate the performance of the Linux with proposed mechanisms, zero copy and storage virtualization by a benchmark program design of ours. The purpose of this evaluation is to confirm the performance degradation in the file system by comparison with the non-optimization file system performance shown in Fig.5. The benchmark program is used to measure sustained sequential file read speeds. The time used for reading files at a total size of 400 MB was measured. For example, when the read speed of a 10 MB file was measured, forty 10 MB files were prepared, the read system for all entire files was issued, and the read speed was calculated from the moment the read began. In order to evaluate the performance under practical conditions, the clear-up program was not used. The sustained sequential file write speeds were measured by similar procedures.

In order to raise efficiency the Linux parameters of block size, fragment size and cylinder group size were 64, 64 and 1 024 KB, respectively. Fig. 5 (a) shows the measured performance of file read speeds in the proposed Linux for the iNAS. In the measure, the file size was changed from 64 KB to 100 MB. The performance of file read speeds in the conventional Linux is also plotted in Fig.5(a). In the conventional Linux, the iNAS performance is saturated at about 21 MB/s, because of the overhead of the memory-to-memory copy performance in the buffer cache. On the other hand, the optimized Linux obviously overcomes the bottleneck, and the sustained sequential file read speed exceeds 80 MB/s when the file size exceeds 100 MB.

Next, the performance of the sustained sequential file write speed for the iNAS is shown in Fig.5(b). It shows that the proposed mechanisms can reduce the bottleneck and speed up to 30 MB/s. On the other

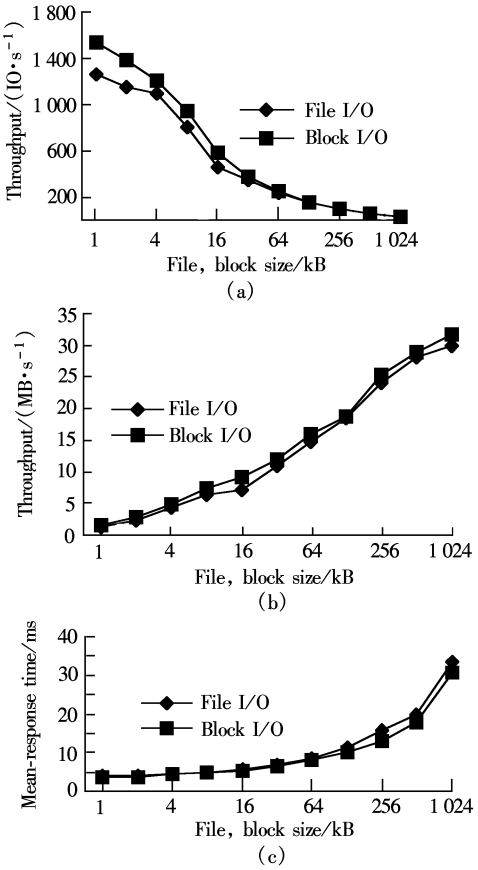

Fig.4 The curves of mean-response time and throughput vs. file/block size

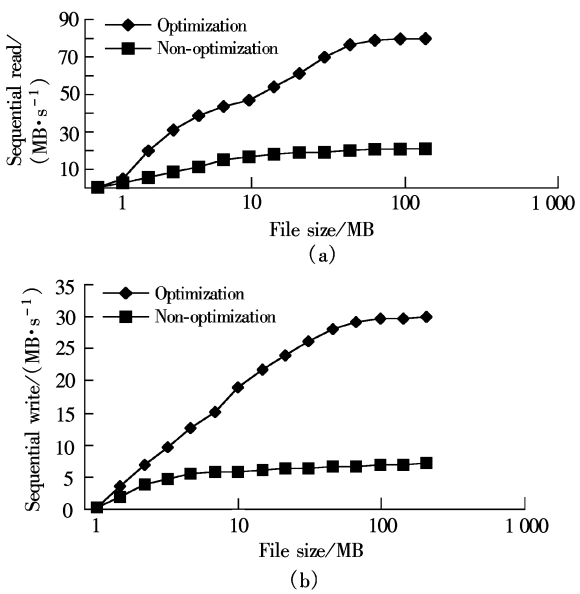

Fig.5 The curves of read/write performance test in server system. (a) iNAS read performance; (b) iNAS write performance

hand, the conventional Linux only provides saturated performance up to about 7 MB.

4 Conclusion

The development of network technology requires bulky and high I/O speed storage devices. Although the traditional NAS has many advantages with respect to file sharing and low price, it only supports the file I/O protocol and has many problems in terms of accessing speed, merging and managing resources. The SAN has many virtues, such as high performance, high response speed, high availability, easy-to-merge resources and centralized management resources, etc., but it doesn't support the file I/O protocol; thus it is very difficult to construct, to maintain and to manage. So the SAN is only suitable for wealthy enterprises, not ordinary ones. First, with the iSCSI module, the iNAS can provide the file I/O service and the block I/O service simultaneously; the iNAS has merits of both traditional NAS and SAN. We can connect multiple iNAS with the IP protocol to build up a large storage network and it can be managed in a manner resembling that of the SAN. Therefore the iNAS converges the NAS and the SAN, and will prove to be very attractive to many medium-sized or small enterprises. Second, with an optimized system, the iNAS can adapt to high-speed networks. The experimental results show that high-speed sustained sequential read performance exceeding 80 MB/s has been achieved.

At present, the RAID in the iNAS sample is RAID0. In the future works, first, we will alter RAID0 with RAID1 and RAID5, respectively, and enumerate their performance analysis. Second, we will connect multiple iNAS to construct a large storage pool via the IP protocol, and evaluate the performance of the whole storage pool.

References

[1] Marc F. *Building storage networks* [M]. New York: McGraw-Hill Companies, 2000. 205－300.

[2] Thadani M, Yousef A K. An efficient zero-copy I/O, framework for UNIX [EB/OL]. http://www.sunmicrosystem.com.2001/2003-06.

[3] Satran J. iSCSI[EB/OL]. http://search.ietf.org/internet-drafts/draft-ietf-ips-iscsi-09.txt. 2001/2003-05.

[4] Kanada T, Onoda T, Yamashita I, et al. Netwarp: an ultra-high-throughput computer communications service [J]. *Network, IEEE Magazine,* **1997, 11**(1): 44－50.

[5] Tsujioka T, Onoda T. An ultra high-speed file server with 105 MB/s read performance based on a personal computer [J]. *Communication, IECICE Trans,* **1998, 81** (12): 2503－2508.

[6] Brustoloni J C, Steenkiste P. Effects of buffering semantics on I/O performance [A]. In: *Proc* 1996 *USENIX Technical Conference* [C]. San Diego, 1996. 277－291.

[7] SNIA. iSCSI for storage networking [EB/OL]. http://www.snia.org, 2001.

一种新的专用存储服务器的设计与实现

韩德志　谢长生　傅湘林　刘　春

(华中科技大学计算机学院, 武汉 430074)

摘要: 为了改进网络存储设备的 I/O 速度和系统性能, 设计了一种新的专用存储服务器, 该服务器是一种基于 iSCSI 的附网存储服务器 (iSCSI-based network-attached storage server, iNAS). 在 iNAS 中, 利用 iSCSI 软件提供的模块, 使得 iNAS 同时提供 file I/O 和 block I/O 服务, 实现了 NAS 和 SAN (storage area network)的融合; 通过在 RAID (redundant array of inexpensive disks)控制器和用户内存之间的直接数据传输(零拷贝), 极大地提高了 iNAS 的 I/O 响应速度; 通过一个多级分流的的设备驱动程序, 将多个 RAID 整合成单一的存储池, 从而实现了存储虚拟化.实验结果显示, iNAS 对文件 I/O 请求和块 I/O 请求都具有极高的响应速度.

关键词: 附网存储; 存储区域网; iSCSI; 零拷贝; 多级分流; Linux 文件系统

中图分类号: TP303