Implementation and Performance Evaluation of an Object-based Storage Device

Shuibing He, Dan Feng

Key Laboratory of Data Storage Systems, Ministry of Education of China
School of Computer, Huazhong University of Science and Technology, Wuhan, China
E-mail: hesbingxq@163.com, dfeng@hust.edu.cn

Abstract

Object-based Storage System (OBSS) has led to a new wave in network storage area .Object-based Storage Device (OSD) which is the cornerstone of OBSS plays a decisive role in the performance of the whole OBSS. This paper describes how we implemented an object-based storage device which adopts a unique hardware architecture based on switching fabric supporting parallel data transfer in multiple I/O channels, and a new object-based storage device file system (HustOSDFS) that can reduce the software overloads to achieve high performance. The experimental evaluation results show that the OSD performs well for system performance. Furthermore, the OSD provides characteristic of large capacity and low cost.

1. Introduction

Storage system has become the bottleneck which heavily influences the further development of today's computer systems. Various problems existing in traditional storage architecture make it difficult to largely improve the performance, reliability and security for storage system. However, the idea of the Object-based Storage System (OBSS) [1] [2] eases the pressure.

Traditional storage system provides the client either with block-based interface (i.e SAN) or with file-based interface (i.e NAS). However, both of the NAS and the SAN have some shortcomings. NAS provides good cross-platform file sharing, but the performance is limited by the file sever. SAN has high throughput by providing direct access to the storage device, but it is not suitable for cross-platform data sharing. Furthermore, SAN has large cost for its security. Contrastively, the OBSS effectively represents a convergence of the NAS and SAN architectures with an object-based interface. The OBSS overcomes the deficiencies in NAS and SAN and makes a perfect solution for network data storage by providing high performance, scalable capacity and throughput, secure object sharing for heterogeneous Operating System.

Object, composed of application data and attributes, is the base logical unit for data access in OBSS. Object is of variable size and can be used to store every type of data such as files, database tables, medical images, or multimedia. Object attribute is used to describe characteristics of object data. For example, a quality of service attribute may describe latency and throughput requirements for a multimedia object. Hence, object attribute is useful to provide the storage device with an awareness of the storage application and enable more intelligence in the device. In OBSS, the object-based storage device (OSD) which is a device stores the objects will represent the next generation of disk drives for network storage[3]. Among the OBSS components, OSD is the most important one and plays a decisive role in the performance of the whole OBSS. Since the petabyte-scale OBSS has thousands of self-contained OSDs working together to provide storage service [8][9], a little improvement of single OSD's performance will result in tremendous performance increase of the large-scale storage system.

This paper presents an implementation of OSD. The goal is to build a high performance OSD meanwhile considering the cost and the capacity. To achieve it, we make efforts in both hardware architecture and software design which is different from many related works. We use special hardware architecture with switching fabric as the platform of the OSD. Based on the platform, a new object-based file system which can reduce the process overheads is implemented. As the switching fabric provides two independent PCI-X buses, it is easy to expand the storage capacity by adding more disks. Meanwhile, the cost is low due to the embedded chips is used as the hardware platform.

The rest of this paper is organized as follow: in Section 2, we give an overview of the Object-Based Storage System and show further details on the functions of the OSD. Section 3 describes the related works on the implementation of the OSD. In Section 4, we discuss the hardware, object-base file system for our OSD. The evaluation results in Section 5 show that the OSD is

0-7695-3097-4/08 $25.00 © 2008 IEEE
DOI 10.1109/SNAPI.2007.11

IEEE computer society

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.

suitable for OBSS and provides better performance as compared to NFS file system and other OSD implementation. We conclude in Section 6 with some discussions on further works.

2. OBSS Architecture Overview and OSD

The OBSS architecture is shown in Figure 1.The OBSS has three main components: the Metadata Server (MDS), the OSDs and clients. The Metadata Server (MDS) provides objects mapping information in OSDs and authentication for clients' data access. When a client accesses the data in an OSD, it first contacts with the MDS and gets the mapping information about the objects. Then the client interacts with the OSD directly. Unlike request to a block device, the request here contains object ID, an offset within the object, attribute values and so on. Finally, the OSD receives the object-based request and performs corresponding operations.

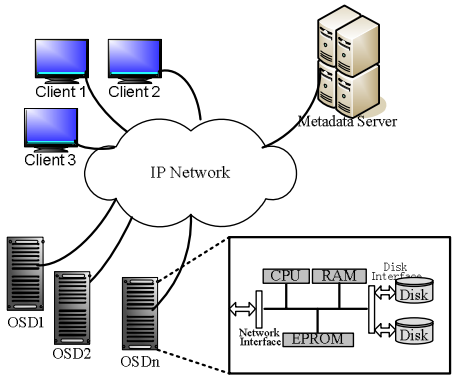

Figure 1. OBSS architecture

The OSD is the cornerstone of the OBSS. It is an intelligent storage device that contains CPU, memory, the storage media (disk), and the network interface which allow it to manage the local object store, and autonomously serve and store data from the network.

The OSD provides three major functions:

Object management. It includes two functions. The first is to reliably store and retrieve object data and object attributes from physical media which includes object meta data management, free space management, object space allocation and so on. The second is to optimize the storage management by using its memory and processor. OSD has the potential to actively learn important characteristics of the environments in which they operate and can understand some of the relationship among the blocks on the device. They can use this information to better organize the data such as intelligent data layout and prefetching .

Device security management. The OSD plays a new security control mechanism. Each request to the OSD must be accompanied with a capability which authorizes the client and its action. The capability is a secure, cryptographic token provided to the client, describing to the OSD which object that the client is allowed to access, with what privileges, and for what length of time. The OSD inspects each incoming transmission for the proper authorization capabilities and rejects any that are missing, invalid or expired[7].

Network communication. The OSD is attached to the TCP/IP network as a network storage device, the network interface is gigabit Ethernet instead of fiber channel and the protocol is iSCSI. The OSD must manage the network communication so that it can receive the iSCSI command from clients and the MDS and resolve the corresponding commands.

As the OSD takes charge of so many functions, it has heavy workload. The intelligent management and the resolution of complex storage protocols greatly increase the workload of the OSD. Therefore, the processing capacity, network interface speed and the disk interface speed must be increased. Otherwise, the OSD will be the bottleneck of the OBSS.

3. Related Works

The Object-based storage system is a hot research field today in network storage technology, a lot of researches has been done to improve the OSD performance.

Typically, as far as we know, IBM Haifa Labs, Luster Inc and Panasas Inc have implemented their own OSD or its prototype. IBM Haifa Labs implemented an OSD prototype: ObjectStone[16], it runs on a Linux server. This hardware architecture has high performance, but its price is expensive .The OST in Lustre project can be view as a OSD and it is implemented with general-purpose PC[4] [5].The OST exports object interface, it translates the object access into the file access which based on the general purpose file system (i.e ext2/3,resiserfs,xfs and jfs) through an internal OBD Filter. Indeed, this implementation has shortcomings both in hardware architecture and software design. In term of hardware platform, though it has relatively lower cost compared with server, but the PC platform has potential limitations in performance and capacity. For example, the disk controllers, network interface controller, and other devices usually are attached to the single I/O bus (PCI) and they share the I/O bus bandwidth. As a result, the PCI bus maybe becomes the bottleneck to further improve the performance of both the disk I/O and network I/O. Furthermore, as limited storage devices can be added to the PCI slot in the main board of PC, it suffers drawbacks of storage capacity. From the view of software design, as the OST is

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.

based on local file system rather than an independent object-base file system, the OST degrades the OSD performance. Similarly, the OSD in Panasas is StorageBlade [6][7],it uses 1.2GHz Intel Celeron CPU and 2 SATA disks as its hardware platform. This OSD has good performance ,but has poor scalability for capacity in one StorageBlade . Consequently, to build a large-scale OBSS, too many StorageBlade will be needed!

To have an excellent trade-off between the performance, capacity and cost, a new OSD architecture based on switch fabric is presented in this paper. Additionally, the object-based device file system also has crucial relationship with the OSD performance. As the traditional object-based file system is usually based on the VFS and general file system, which leads to additional software overheads in the object store, we design a completely independent object-base file system which mapping the object accesses directly to the block device without passing the requests to the VFS. Hence, the OSD performance will be improved.

4. The Object-based Storage Device Design

4.1 The Hardware Design of an OSD Based on Switching Fabric

As mentioned earlier, the OSD has heavy workloads, with the increase of the clients in OBSS, this characteristic will be more notable. To implement an OSD with high performance, the ideal hardware architecture must provide powerful processor, high speed network interface and disk interface. Certainly, the OSD performance will further improved if the hardware platform supports short I/O data path and parallel data transfer in multiple I/O channel.

Since the main task of OSD is processing the object-based I/O requests, we chose the special I/O processor Intel 80200 to support the high process capability. At the same time, we use the Intel I/O companion chip 80314 with switch fabric to transfer data simultaneously in multiple data paths. Rather than relying on the traditional 33MHz 32-bit PCI bus, we chose the 133MHz 64-bit PCI-X bus to transfer data so that the higher I/O throughput can be obtained.

The new hardware architecture is shown in Figure 2.The Intel 80314 is the interconnection core of the OSD，with two Intel 80200 processors attached to its Core Interface Unit (CIU). The 80200 is the chief processing unit to perform the major tasks of OSD, such as network protocol resolve，basic system operations and advanced task management. Besides these main chips, some peripheral components are employed. A DIMM SDRAM is connected to the 80314 as system main memory, while a flash memory is used to keep the necessary data needed to boot the system attached to the Peripheral Bus Interface (PBI) of 80314. Two Gigabit Ethernet PHY Transceivers, Marvel 88E1020, connect to the 80314's two integrated MAC ports. Connected to the 80314 by PCI-X bus, the four Intel 31244 serial ATA controllers realize the communication between host and disk storage, and bring large capacity for OSD since they each has four serial ATA disk interfaces. A LCD display attached to the PBI is used to display the system status. Also, a JTAG port is provided for advanced hardware debugging.

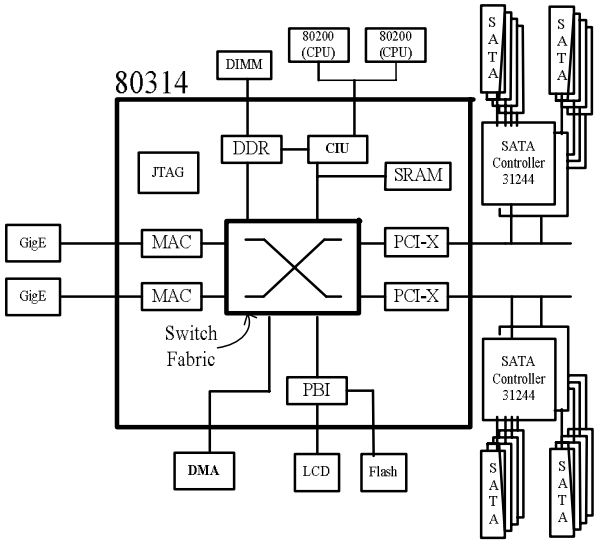

Figure 2. The OSD architecture based on switch fabric

The Intel 80200 processor based on Intel XScale architecture supports frequency from 200 MHz to 733 MHz[10]. What is the much surprising is that even at 600 MHz the 80200 processor dissipates less than a watt. This will make the OSD provide high performance and low power. Indeed, the iSCSI protocol is a heavy weight protocol, the resolution of iSCSI protocol and the disk processing may result in a high utilization of one CPU. Consequently, the OSD can only provide goodish performance. As the OSD has two 80200 chips, one of them can resolve the iSCSI protocol, the other can take charge of the disk operations and run the OS to have an excellent performance.

The Intel 80314 I/O companion chip includes the following main parts: two 80200 bus interfaces, integrated DDR SDRAM controller, two integrated Gigabit Ethernet MAC controllers, two integrated PCI-X interfaces, and the peripheral bus interface [11]. The dual-ported SDRAM memory controller interface runs at 200 MHz, and offers two-port concurrent access to memory with programmable arbitration for each port. The two integrated Gigabit Ethernet controllers provide high network bandwidth for the OSD. The Intel 31244

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.

serial ATA controller is a single-chip solution for a PCI-X-to-Serial ATA Host Bus Adapter (HBA) [12]. It supports serial ATA speed of 150 MB/s of raw data.

The significant characteristic of the OSD is that the 80314 is designed as a fabric-centric, any-port-to-any-port bridge. It uses an internal switching fabric and supports concurrent transactions from any interface to any other interface. Such a unique characteristic which can't be obtained from other popular hardware architecture brings a lot of benefits to improve the OSD performance. In the OSD, the above characteristic is used to optimize the I/O performance through three methods:

1. Data is transferred in parallel by using two independent high-speed PCI-X buses attached to the switch fabric, the OSD disk I/O performance can be improved.
2. Two network interfaces can work concurrently to increase the throughput and reduce the network latency.
3. The direct data transfer between the disk and the network interface is used, so the I/O data path is shorter and additional memory copy is avoided.

The complete design of the OSD includes two important issues: the hardware design and the porting of bootloader and embedded OS for the hardware. In the hardware design of the OSD, a fundamental hardware platform is rapidly developed using the Intel IQ80315 processor evaluation platform. Redboot is choosed as the bootloader and it is successfully ported on the hardware platform. Above redboot, a commercial embedded linux OS -Timesys linux is then ported to manage the hardware resources and perform the functions of TCP/IP protocol stacks.

4.2 The Object-based File System

In order to provide the Object-based data store functions, storage software is implemented based on the new hardware architecture in the OSD. Figure 3 shows the software layers. The iSCSI module waits for the iSCSI packages from the clients and puts them into a system queue. The iOM module is implemented as a kernel module. It gets package from the system queue and resolves the iSCSI commands, once a SCSI/OSD command is received, it performs the data access through Object-based File System (OBFS) and the corresponding device driver. According to the information in the SCSI commands, the iOM module implements the operation of Primary Command such as INQUIRY, REPORT LUNS, MODE SELECT and OSD Command, such as OSD READ, OSD WRITE [2].

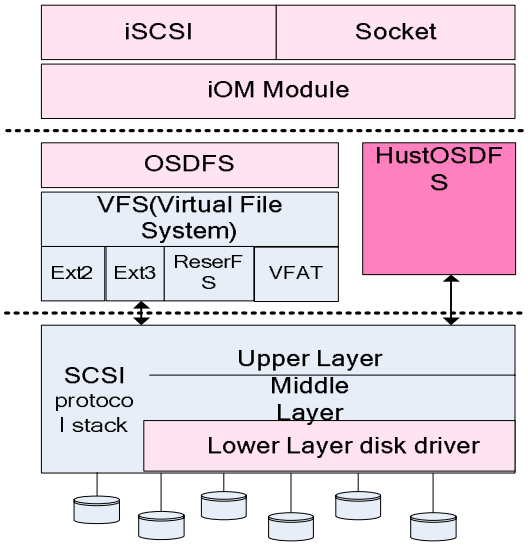

Figure 3.The software architecture in OSD

Table 1. The interface functions in OBFS

The functions related Object operation	Description
int osdfs_create_partition (uint64_t lun, uint64_t * pid)	Create partition object
int osdfs_remove_partition (uint64_t lun, uint64_t pid)	Delete partition object
int osdfs_create_object (uint64_t lun, uint64_t pid, uint64_t* uid)	Create object
int osdfs_remove_object (uint64_t lun, uint64_t pid, uint64_t uid)	Remove object
int osdfs_read_object (uint64_t lun, uint64_t pid, uint64_t uid,char* buf, uint64_t offset, uint64_t count)	Read object
int osdfs_write_object (uint64_t lun, uint64_t pid, uint64_t uid,char *buf, uint64_t offset, uint64_t count)	Write object
Int osdfs_flush_object (uint64_t lun, uint64_t pid, uint64_t uid,)	Flush object to disk
int osdfs_read_attr (uint64_t lun, uint64_t pid, uint64_t uid,uint32_t page, uint32_t index, char * buf, uint32_t maxlen)	Read attribute
int osdfs_write_attr (uint64_t lun, uint64_t pid, uint64_t uid,uint32_t page, uint32_t index, uint32_t len, char * buf)	Write attribute

The OBFS implements the specific object-based operations derived from the iOM module. It maps the object level requests to the block level requests through a certain mechanism, and then accomplishes the disk read/write operations by calling the underlay disk device driver. Several functions are provided for iOM in OBFS and they are listed in table 1.

The OBFS is a crucial layer to the OSD performance. In the OSD, two kinds of OBFS: OSDFS and HustOSDFS are implemented. The OSDFS is a former version and it is based on the general-purpose file sys-

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.

tem as shown in Figure 3. The OSDFS is based on the OBFS in [8] and the OSD reference implementation of Intel [13].As the object processing is passed to the VFS and ext2/3 file system, much additional overhead is generated. To improve the performance, the new OBFS-HustOSDFS is designed, which maps the object-based access to lower block-based access directly. It manages the object store in the OSD independently without interacting with the VFS and the file system such as ext2, ext3 and so on. The structure of the HustOSDFS is shown in Figure 4.

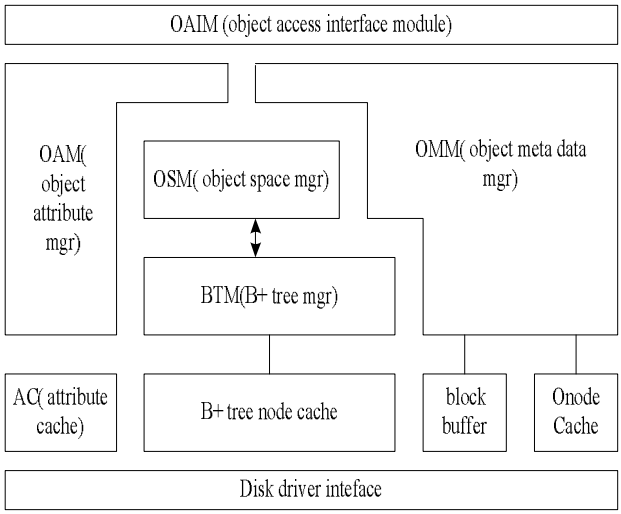

Figure 4.The structure of HustOSDFS

HustOSDFS has several modules working together to provide object-based data store with high efficiency. The BTM takes charge of all kinds of operations in B+ tree such as insert, delete and select. In HustOSDFS, B+ tree is a widely used structure to effectively organize the disk space and other information .The OSM is the core of the HustOSDFS, it manages all the free blocks in system by calling the BTM and it stores the B+ tree node structure in cache. The OMM manages the metadata of the objects and accomplishes the mapping between the logical block to the physical block in an object. Analogous to the inode structure in the VFS, the object metadata exists in the onode. The OMM allocates and retrieves the disk space for the object in the OSD. Besides object data related modules, the OAM accomplishes the store and retrieval of the object attributes in the disk with Extensible Hashing.

5. Experimental Evaluation

The OSD Prototype has been implemented. Presently, only one CPU is running on the OSD because the Timesys linux doesn't support two CPUs now; modifying the kernel to support both CPUs is the future work. Nevertheless, the OSD still exhibits excellent performance in our experiments. In this section, we present experimental setup and numerical results.

5.1 Experimental Setup

For the purpose of performance evaluation, an object-based client file system mounted on the client machine is also developed. Furthermore, the MDS is implemented on a PC. The experiment platform is show in table 2.

Table 2. The Configure information of the MDS and the Client

	CPU	Main board	Memory	Disk	Network
MDS	Intel Xeon 3.0GHz	Super-micro X6DHE-XB	DDR ECC RG 512M	Maxtor Diamond Max10/ SATA150 200GB/ 7200RPM/ 8MB buffer	Bcm5700 Gigabit Ethernet
Client					
Switch	Cisco Catalyst 3750 series Gigabit Ethernet Switch				
OS	Redhat 9, Kernel Version 2.4.20				

The MDS, Client and the OSD are connected to the Switch. In the experiment, only one Gigabit Ethernet interface in the OSD is used and the other will be used in the future. The OSD directly provides 8 Serial ATA disks in every PCI-X bus and more disks can be provided to expand the capacity by adding a Serial ATA control card into the PCI-X slot. In the experiment, only one disk is tested, indeed, more disks can be configured as a MD with RAID technique to improve the disk I/O performance. To access the data in OSD, a test directory is created in the Client machine, and the client object-based file system is mounted on the directory, and then all kinds of operations can be done in the directory as in a general file system. The client object-based file system first requests the MDS with a file-based interface, then the MDS maps the file information into the object information and returns the Client the object mapping list which indicates that the OSD address is the OSD IP and the object ID in the OSD. After that, the Client accesses the data with an object-based interface through the Gigabit Ethernet network. Iozone is a popular benchmark that has been used extensively for basic evaluation of file systems [14], and iozone is choosed as the benchmark for the experiment.

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.

5.2 Results

In the first experiment, the read and write performances are measured and it is compared with the NFS configuration. The motive of this experiment is to measure the performance of the OSDFS, HustOSDFS and NFS. For the NFS configuration, the NFS daemon is set up on the OSD and exported a directory and the directory is mounted from the Client. In the experiment, an 8MB file is tested with multiple transfer sizes. Figure 5, Figure 6 shows the results of this experiment.

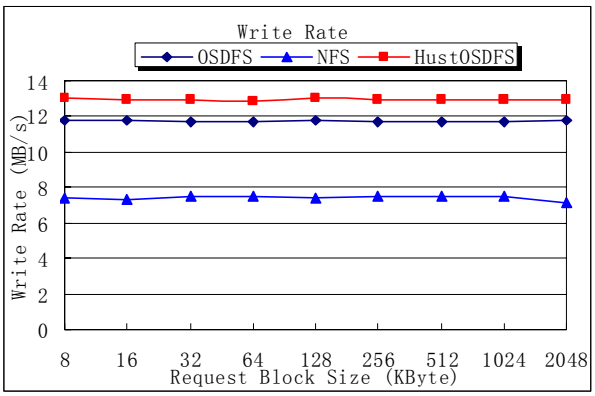

Figure 5. The write performance comparison

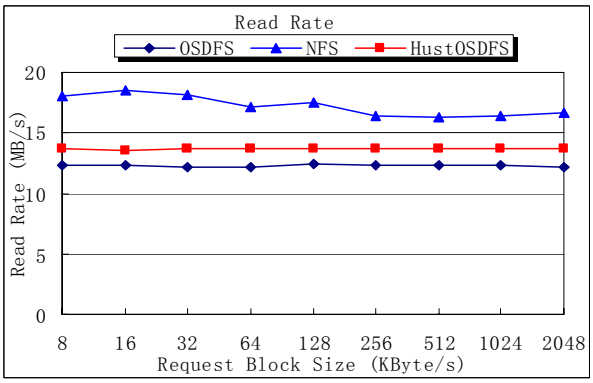

Figure 6. The read performance comparison

The HustOSDFS has better write and read performance than OSDFS due to the reduced VFS and general file system overloads in HustOSDFS. It can also be noticed that both the OSDFS and HustOSDFS have better write performance than NFS. This can be explained as follow. First, when the OSD platform is configured with the NFS, it manages all the metadata of the file includes user component and storage component, but OSDFS and HustOSDFS only manage the storage component. In addition, an important issue to the performance of write is the disk space management and disk free space allocation algorithm. In the OSDFS and HustOSDFS, disk space management and disk free space allocation are implement with B+ tree which can lead a high efficiency. In term of read performance, NFS is excellent than OSDFS and HustOSDFS. As the OSDFS and HustOSDFS provide only cache for data has been accessed without prefetching the data will be accessed in the future, they can not provide excellent read performance.

Meanwhile, the performance of HustOSDFS is compared with the result in [15].It can be found that the OSD has a better write performance and a comparable read performance. Still, the OSD has potential to further improve the performance in Gigabit Ethernet network. As iSCSI is a heavy weight protocol, the CPU utility and network throughput is studied in the second experiment. The CPU utility reaches from 73 to 82 when the write and read performance is above 13MB/s. As a result, decreasing the CPU utility is a future work.

6. Conclusions and Future Work

Object-based Storage Device (OSD) which is the foundation of OBSS plays a decisive role in the performance of the whole OBSS. How to build an OSD with low cost, large capacity and high performance for large scale storage system is a challenge. This paper describes how to implement an OSD with two unique characteristics which is different from other OSD design. First, the OSD adopts a unique hardware architecture based on switching fabric which can supports parallel data transfer in multiple I/O channels so as to improve the performance. Meanwhile, it supports large capacity for its two independent high speed PCI-X buses and low cost as it's an embedded platform. Second, based on the hardware platform, a new object-based storage device file system (HustOSDFS) is running on the OSD. Rather than depending on the VFS and general file system such as ext2/3, the HustOSDFS manages its own object-based access which can reduce the software overload.

In the future, a lot of works will be done on software layer to make full use of the characteristic of the OSD hardware architecture. For example, the OS kernel will be modified to support both CPUs in the OSD. Furthermore, the storage software will be optimized and an adaptive prefetching algorithm will be designed for the OSD to further improve the OSD performance.

Acknowledgements

This work was supported by the National Basic Research Program of China (973 Program) under Grant No.2004CB318201, the Program for New Century Excellent Talents in University NCET-04-0693, Wu-

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.

han Project 20061002031&200750730307, and the National Science Foundation of China No.60603048.

References

[1] M.Mesnier,G.R.Ganger,E.Riedel. Object-based Storage .IEEE Communications Magazine,2003，Vol.41, Issue 8:84-90

[2] J.Satran. Object-based Storage Device Commands.http://www.t10.org/draft/osd. Oct.2004

[3] PANASAS WHITE PAPER: Object Storage Architecture: Defining a new generation of storage systems built on distributed, intelligent storage devices. 2003

[4] Peter J Braam. The Lustre Storage Architecture. Cluster File Systems, Inc. Whiter Paper. http://www.clusterfs.com. 2004

[5] Yu Weikuan, R. Noronha, Liang Shuang, et al. Benefits of high speed interconnects to cluster file systems: a case study with Lustre. In: The 20th International Parallel and Distributed Processing Symposium(IPDPS 2006). 2006. 8~15

[6] Tang Hong, A.Gulbeden, Zhou Jingyu, et al. The Panasas ActiveScale Storage Cluster - Delivering Scalable High Bandwidth Storage. In: Proceedings of the ACM/IEEE SC2004 Conference on Supercomputing. 2004. 53~62

[7] Panasas Inc. Object Storage Architecture. White Paper. http://www.panasas.com/ objectbased_mgnt.html

[8] Wang Feng, Brandt Scott A., Miller Ethan L., et al. OBFS: A File System for Object-based Storage Devices. In: Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST2004). 2004. 101~118

[9] Andy Hospodor, Ethan L. Miller, Interconnection Architectures for Petabyte-Scale high-performane storage system, Proceedings of the 21st IEEE /12th NASA Goddard Conference on Mass Storage Systems and Technologies(MSST2004). 2004 .

[10] Intel Corp: Intel 80200 Processor based on Intel XScale Microarchitecture Datasheet,2003

[11] Intel Corp: Intel GW80314 I/O companion Chip Datasheet,2004

[12] Intel Corp: Intel 31244 PCI-X to Serial ATA Controller Datasheet ,2004

[13] R. Intel Corporation. Intel iSCSI Reference Implementation. Details available at http://www.intel.com/ technology/computing/storage/iscsi/index.htm.

[14] Iozone filesystem benchmark. http://www.iozone.org.

[15] David Du, Dingshan He, Changjin Hong, et al. Experiences Building an Object-Based Storage System based on the OSD T-10 Standard. In: Proceedings of the 23st IEEE /14th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST2006). 2006.

[16] M. M. Factor, K. Meth, D. Naor, et al. Object Storage: The Future Building Block for Storage Systems. In: Proceedings of the 2nd International IEEE Symposium on Mass Storage Systems and Technologies. 2005. 119~123

Authorized licensed use limited to: Zhejiang University. Downloaded on May 11,2023 at 11:36:19 UTC from IEEE Xplore. Restrictions apply.