Design of an Object-based Storage Device Based on I/O Processor

Shuibing He
Wuhan National Laboratory for Optoelectronics
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074, China
E-mail: hesbingxq@163.com

Dan Feng
Wuhan National Laboratory for Optoelectronics
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074, China
E-mail: dfeng@hust.edu.cn

Abstract

Object-based Storage Device (OSD) is the foundation of the Object-based Storage System (OBSS).As the petabyte-scale OBSS includes thousands of OSDs, the performance, cost and power of single OSD must be considered together to build such a huge storage system The existing OSD based on server and general-purposed PC platform cannot have an excellent tradeoff among the three factor as they are not designed specifically for storage applications. An original OSD architecture based on the Intel IOP315 I/O processor chipset is presented in this paper. The I/O processor makes it powerful for the OSD to process the network communication protocol and the unique switch fabric of the chipset can further improve the I/O performance through parallel data transfer in multiple I/O channels. The experimental results show that the OSD performs well for system performance. Moreover, it provides characteristic of low cost and power.

Categories and Subject Descriptors

B.4.2 [**input/Output and Data Communication**]: Input/Output Devices-*channels and controllers*; C.3 [**Special-Purpose and Application-Based Systems**]:Real-time and embedded systems

General Terms

Design, performance

Keywords

Object-based storage device, I/O processor, switch fabric, performance, power

1. Introduction

Object-based Storage System (OBSS) has led to a new wave in network storage field [1] [2]. The OBSS captures the benefits of NAS and SAN; it can provide storage with high performance, scalable capacity and bandwidth, secure data 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. Object attribute is used to describe characteristics of object data.

The object-based storage device (OSD) which stores the objects is the core of the OBSS. The OSD will represent the next generation of disk drives for network storage [3]. Since the petabyte-scale OBSS has thousands of self-contained OSDs working together to provide storage service [8][9], the performance, cost and power of single OSD can largely influence the overall system. As a result, the design of an OSD with excellent tradeoff among the three factors is critical to build such a large-scale storage system.

At present, the OSD or its prototype is usually based on Sever or PC platform [3, 4, 5, 6, 7]. Though these kinds of OSDs usually have good performance, they are not suitable for OSD design as they do not fully consider the characteristics of storage applications. First, as these platforms have powerful processors, plenty of memory and other functional modules, they cost a lot. Second, the existed resources are not configured with good balance for storage. On the one hand, the components unrelated to storage application are waste and the exceeded processing capability is not fully used. On the other hand, the storage related components seem to be lack or weak. For example, the bandwidth of the I/O bus in PC platform usually has limitation for further improve the disk and network I/O performance. Third, though these kinds of design have relative superiority in terms of computation speed, they usually bring forward an inherent drawback that the power of these OSDs is considerable. As the OSD usually provides 7x24 services, the overall power of such a huge system is a serious problem when the system runs over time.

This paper presents a novel design of OSD. The goal is to build a good performance OSD meanwhile considering the cost and the power. The Intel IOP315 I/O processor chipset with switch fabric is used as the platform of the OSD. Based on the platform, effective object-based software is implemented. Meanwhile, the cost and the power is low due to the embedded chips is used.

The rest of this paper is organized as follow: in Section 2, we give an overview of the Object-Based Storage System and the OSD. Section 3 describes the related works on the design of the OSD. In Section 4, we discuss the hardware platform and the object control software on our OSD. The test results in Section 5 show that the OSD performs well for system performance. Finally, section 6 concludes the paper.

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 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.

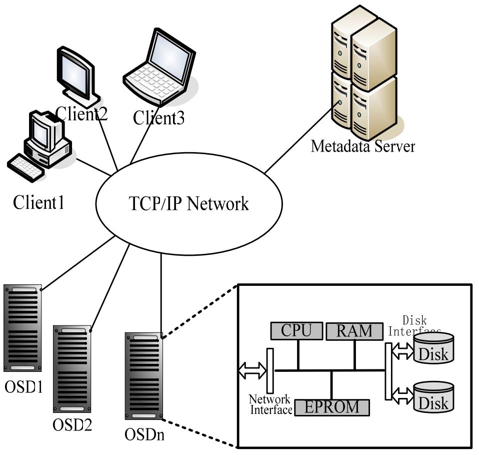


Figure 1. OBSS architecture

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. Generally speaking, the OSD provides three major functions:

Object management. The first is to reliably store and retrieve object data and object attributes from physical media. The second is to optimize the storage management by using its memory and processor.

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 OSD inspects each incoming transmission for the proper authorization capabilities and rejects any that are missing, invalid or expired.

Network communication. The network interface is gigabit Ethernet instead of fiber channel. The OSD must manage the network communication so as to receive the iSCSI command from the clients and the MDS.

The OSD will have heavy workload when it faces with the data intensive application. Therefore, the processing capacity, network interface speed and the disk interface speed must be increased. Otherwise, it will be the bottleneck of the OBSS.

3. Related Works

The Object-based storage is a hot research field today in network storage technology. A lot of researches have been done to design the OSD and the PC platform is a popular choice for their convenience and the short time to develop the system.

IBM Haifa Labs implemented an OSD prototype: ObjectStone[13], it runs on a Linux server. It has high performance but its price is expensive. The OST in Lustre project can be view as the control software of an OSD and it usually is configured to run on 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 through an internal OBD Filter. Though it has relatively lower cost, but the PC platform has potential limitations in performance. For example, the disk controllers, network interface controller, and other devices usually are attached to the single I/O bus (PCI). 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. Further more, as the OST is based on local file system rather than an independent object-base file system, the OST degrades the OSD performance. 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 bandwidth and capacity in one StorageBlade.

4. The Object-based Storage Device Design

4.1 I/O processor overview

The Intel IOP315 I/O processor chipset with Intel XScale technology is the product in Intel's fourth-generation of I/O processors. It contains two devices: the Intel 80200 processor based on Intel XScale microarchitecture [10] and the Intel 80314 I/O companion chip [11]. The chipset provide a rich peripheral set designed for storage and network applications. When used as an OSD, the chipset provides a high-performance cost-effective platform.

4.2 The Architecture of an OSD Based on Switch Fabric

The ideal design of OSD should consider the requirements of the storage applications such as proper performance of processing capability, high-speed I/O bandwidth for network and disk interface, specified hardware units to accelerate the calculating when the technique RAID is used and the low power.

In view of the perfect solution for all these aspects, the Intel IOP 315 embedded system is used as the hardware platform of OSD. As shown in the 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 Intel 80200 performs 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 Intel 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). Two Gigabit Ethernet PHY Transceivers, Marvel 88E1020 which provide high speed network bandwidth are connect to the 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.

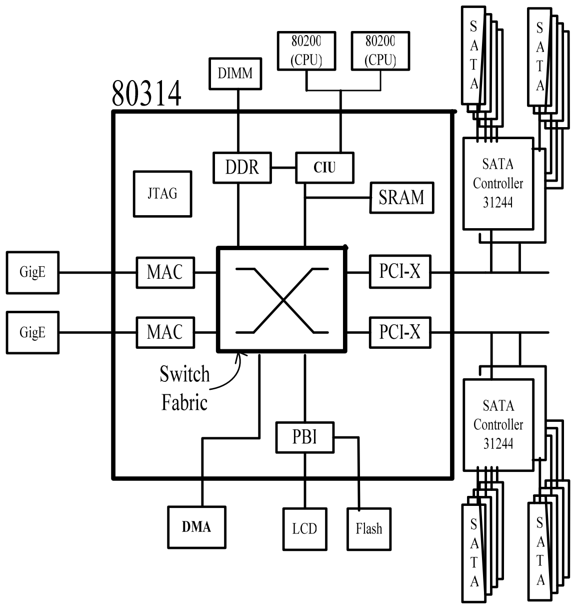


Figure 2. The OSD architecture based on switch fabric

The Intel 80200 processor supports maximum frequency 733 MHz [10]. What is the much surprising is that even at 600 MHz the 80200 processor dissipates less than a watt. Comparing with the power 89 watt which a typical Intel P4 3.0GHz processor dissipates, the embedded processor Intel 80200 has salient superiority in 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 processor when it face with data intensive applications. As the OSD has two processors, 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 significant characteristic of the OSD is that the Intel 80314 is designed as a fabric-centric, any-port-to-any-port bridge. It uses an internal switch 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, two methods are used to optimize the I/O performance with the implements by the software.

1) Data is transferred in parallel by using two independent high-speed PCI-X buses attached to the switch fabric. Here, the technique RAID is used improve the OSD disk I/O performance.

2) Two network interfaces can work concurrently to increase the throughput and reduce the network latency. It is implemented with the network binding technique.

4.3 The address map mechanisms

The Intel 80314 has several address spaces: 32-bit CPU/CIU space, 64-bit fabric space(SFN address space) and two 64-bit PCI/X spaces. Each functional block attached to the switch fabric contains registers to set up addressing properties such as base address, window size, address translation, and routing across the fabric. Different functional blocks can have slightly different configuration mechanisms. The GigE, PBI, DMA/XOR, and 64-bit SDRAM fabric ports all operate off of the SFN address space and thus have simpler decode mechanisms comprised of an address register to set the SFN start address for the unit and a mask register to set properties such as the decode window size for the unit.Table 1 describes the OSD address mapping.

Table 1. The memory map of the OSD

Address	Size (MB)	Description
0x00000000	1 GB	SDRAM
0x40000000	256	FLASH/PBI
0x50000000	1	SRAM
0x50100000	64 KB	Control Registers
0x80000000	495	PCI1 MEM32
0x9EFF0000	1	PCI1 I/O
0x9F000000	16	PCI1 CFG
0xA0000000	256	PCI1 PFM1
0xB0000000	256	PCI1 PFM2
0xC0000000	495	PCI2 MEM32
0xDEFF0000	1	PCI2 I/O
0xDF000000	16	PCI2 CFG
0xE0000000	256	PCI2 PFM1
0xF0000000	256	PCI2 PFM2

4.4 The Software Architecture

In order to achieve the Object-based data storage functions, we have implemented the storage software based on the hardware architecture. Figure 3 shows that the software system is made up of two main components: Control Module and real time embedded Linux OS. The Control Module runs on the real time OS and processes the iSCSI and OSD commands, while the real time OS based on the hardware platform accomplishes the data read /write operations through Object-base File System (OBFS) and the corresponding device driver. The OBFS and the Control Module are implemented in the form of a module in kernel space.

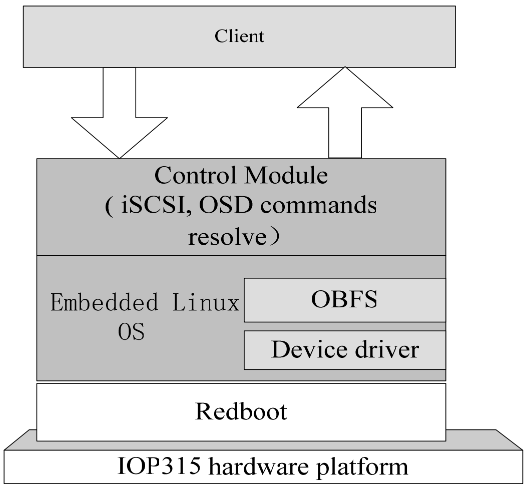


Figure 3. The OSD software architecture

The Control Module waits for the iSCSI commands come from the OBS clients。Once a command is detected, it resolves the SCSI commands carried in the iSCSI package. According to the information in the SCSI commands, the Control Module implements the operation of Primary Command such as INQUIRY, REPORT LUNS, MODE SELECT and OSD Command, such as OSD READ, OSD WRITE [2].The detailed processing is seen in figure 4.

The OBFS implements the specific object-based operations derived from the Control Module, such as hustosdfs_write_object, hustosdfs_read_object, etc. It maps the object level request to the block level request through a certain mechanism, and then accomplishes the disk read or write operations by calling the disk device driver in the lower layer.

5. Performance Evaluation and power comparison

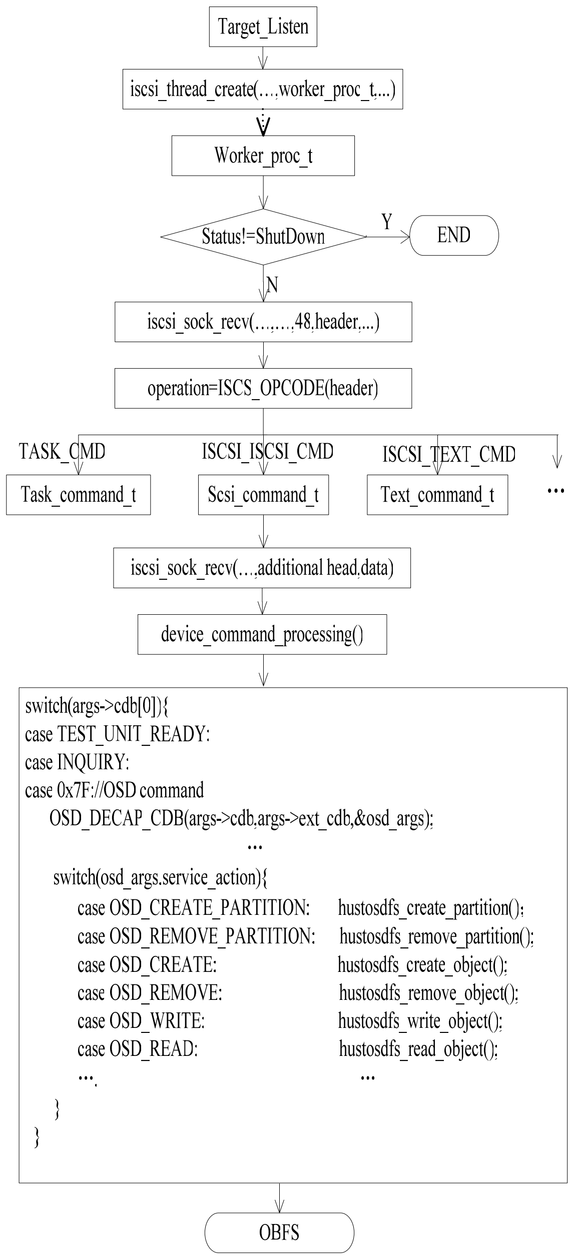
The OSD Prototype has been implemented. Its picture is shown in Figure 5.Presently, only one CPU is running on the OSD because the Embedded Linux doesn't support two CPUs now; modifying the kernel to support both CPUs is the future work. Nevertheless, the OSD still exhibits excellent performance in our 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.

The MDS, Client and the OSD are connected to the Switch. Iozone is a popular benchmark that has been used extensively for basic evaluation of file systems [12], and it is chosen as the benchmark for the experiment.

The motive of this experiment is to measure the performance of the OSD based-on IOP315 and OSD based on PC platform. The configuration of the OSD based on PC platform is the same as the client in the table 2. In the experiment, only one Gigabit Ethernet interface in the OSD based on IOP315 is used and the other will be used in the future. Moreover, 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.


Target_Listen
iscsi_thread_create(….worker_proc_t,...)
Worker_proc_t
Status!=ShutDown
Y
END
N
iscsi_sock_recv(…,…,48,header,...)
operation=ISCS_OPCODE(header)
TASK_CMD
ISCSI_ISCSI_CMD
ISCSI_TEXT_CMD
Task_command_t
Scsi_command_t
Text_command_t
…
iscsi_sock_recv(…,additional head,data)
device_command_processing()
OBFS

```
switch(args->cdb[0]){
case TEST_UNIT_READY:
case INQUIRY:
case 0x7F://OSD command
    OSD_DECAP_CDB(args->cdb,args->ext_cdb,&osd_args);
                    ...
    switch(osd_args.service_action){
        case OSD_CREATE_PARTITION:    hustosdfs_create_partition();
        case OSD_REMOVE_PARTITION:    hustosdfs_remove_partition();
        case OSD_CREATE:              hustosdfs_create_object();
        case OSD_REMOVE:              hustosdfs_remove_object();
        case OSD_WRITE:               hustosdfs_write_object();
        case OSD_READ:                hustosdfs_read_object();
        ….                              ...
    }
}
```

Figure 4. The details of the Control Module

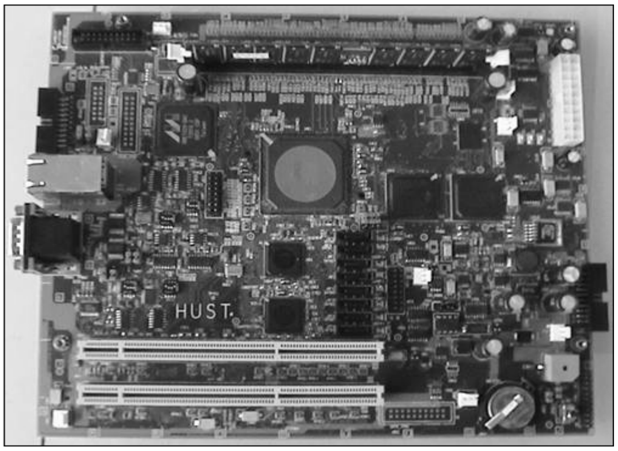


Figure 5. The OSD picture

Table 2. The Configuration 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	Bcm5700 Gigabit Ethernet
Client					
Switch	Cisco Catalyst 3750 Gigabit Ethernet witch				
OS	Redhat 9, Kernel Version 2.4.20				

5.2 Results

In the experiment, a 64MB file is tested with multiple transfer sizes. Figure 6, Figure 7 shows the results of this experiment.

It is can be seen from the figures that the OSD based on PC platform has relative better read /write performance than that based on IOP315. It is mainly because the processing capability in PC platform farther exceeds that based on IOP315. To verify the cause of performance difference, the CPU utilizations in the two OSDs also are tested. We find the utilization is about 53-64% in PC platform and 82-91% in IOP315 platform. Though the CPU in PC platform is several times faster than that in the IOP 315 platform, the I/O performance benefit can not be obtained proportionally. This shows the resource in our new design is configured with better balance. As the read/write performance gap is not large, if the two processors work together in the OSD based on IOP315 in the future, we believe that the performance of the new design will exceed that in the OSD based on PC platform.

5.3 Power Comparison

The powers of the two kinds of OSD are tested. The IOP platform is less then 25 watt, but the PC platform is more than 160 watt when the disk power is not considered as it is the same. This can show the benefit of the OSD based on IOP315 in power due to the adaptations of the embedded chips.

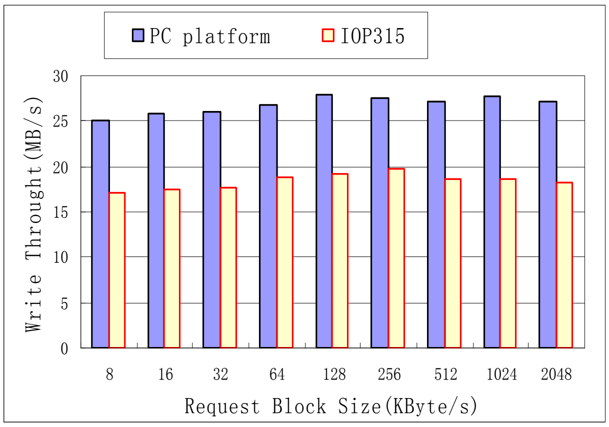
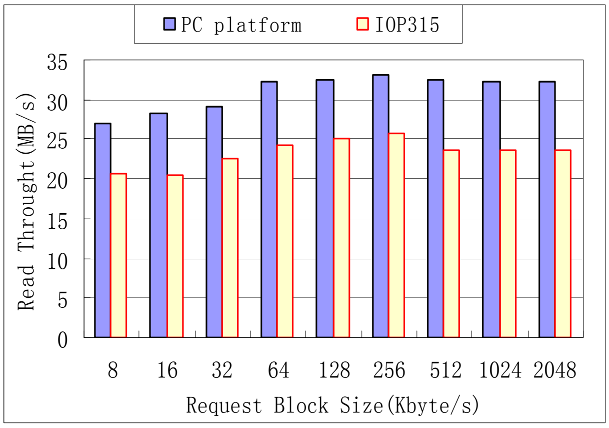


Figure 6. The write performance comparison

Figure 7. The read performance comparison



6. Conclusions

As the petabyte-scale OBSS includes thousands of OSDs, the performance, cost and power of single OSD must be considered together to build such a huge storage system The existing OSD based on server and general-purposed PC platform cannot have an excellent tradeoff among the three factor as they are not designed specifically for storage applications. An original OSD architecture based on the Intel IOP315 I/O processor chipset is presented in this paper. The I/O processor makes it powerful for the OSD to process the network communication protocol and the unique switch fabric of the chipset can further improve the I/O performance through parallel data transfer in multiple I/O channels. The experimental results show that the OSD performs well for system performance. Moreover, it provides characteristic of low cost and power.

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.

7. 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, Wuhan Project 20061002031&200750730307.

8. References

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

[2] T10 work group. SCSI object-based storage device commands (OSD). T10/1355-D working draft,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] Iozone filesystem benchmark. http://www.iozone.org.

[13] 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