Building Novel Network Storage System Based on Volume Holographic Device*

Qiang Cao Changsheng Xie
caoqiang@mail.hust.edu.cn cs_xie@mail.hust.edu.cn
Chinese State Storage System Lab
Computer School of Huazhong University of Science and Technology
Wuhan, Hubei, P.R.China 430074

Abstract: In the past few years considerable demand has developed for oceanic data storage system that are able to store Petabytes data. Volume holographic recording has the potential to offer high density, fast data readout rate, and associative content addressable storage as compared with other conventional mass data storage technology. However, its total cost and size impedes to go into market. If the volume holographic device could join network and be shared by more and more customers, the average cost for each customer should be reduced. On the other hand, the growing network storage technologies shake off the traditional storage architecture (DAS) limit including physical topologies and access mode, and provide high scalability, availability and flexible for storage system, moreover are widely applied to many fields. Obviously, it is valuable for building network storage device based on volume holographic. In this paper, we design the architecture and relative software of volume holographic device to enable common storage network, which can support the SAN based on Fibre channel and iSCSI. Our experiment shows the HSD (Holographic Storage Device) has excellent performance for network storage.
Keyword: network storage, volume holographic, SCSI target

1. Introduce

In the past few years considerable demand has developed for oceanic data storage system that are able to store Petabytes of data, such as science computing, weather forecast, business, etc. However, disk access times have not kept pace with the evolution of disk capacities, CPU speeds and main memory sizes. They have only improved by a factor of 3 to 4 in the last 25 years whereas other system components have almost doubled their performance every other year. As a result, disk latency has an increasingly negative impact on the overall performance of many computer applications. With disk array, such as RAID0, 1, the performance can be improved remarkably. However, the more parallelism maybe be limited by disk synchronization and topology. For example, the RAID 0 usually concludes less 6 disks.

The growing network storage technologies shake off the traditional storage architecture (DAS) limit including physical topologies and access mode, and provide high scalability, availability and flexible for storage system, moreover are widely applied to many fields[1]. The network storage system has been the common architecture for data intensive applications.

Apparently the new storage technique with high performance and capacity is expected. Volume holographic recording has the potential to offer high density, fast data readout rate, and associative content addressable storage as compared with other conventional mass data storage technology[2]. It's meaningful to design novel architecture of the storage system based on volume holographic technology.

Although high-speed CCD detectors have been developed, the prospect for low cost, high performance devices is questionable. Since the laser diode array allows us to switch between multiplexed data pages with negligible delay (on the order of nanoseconds), the random access time and the readout rate become limited by the required integration time of the detector. The CCD is bottleneck of the system. Although the better performance CCD can be used in holographic storage, the price of whole system makes the storage system lost competition from magnetic disk. However, if more and more customers can share the same expensive, high capacity and performance device, the average cost for each customer should reduce. On the other hand, with traditional DAS (directed attached storage) mode, the excellent performance provided by holographic device is weakened by attached host that has limited external width and process ability. The network storage topologies could void single point failure and performance bottleneck caused by server.

* This paper is supported by both National Natural Science Foundation of P R China under the Grant NO 60173043 and 60273073

In this paper, we present the architecture and relative support software of holographic device for network storage. There are two ways to be able to implement, one is Fibre channel and the other is iSCSI, which are fashionable interface in network storage and support the SAN based on Fibre channel and iSCSI. Our experiment shows the HSD (Holographic Storage Device) has excellent performance for network storage.

2. Network Storage System

At first, we must decide to use which topologies and protocol connect volume holographic storage drivers with host. Volume holographic storage typically provides a simple, untyped, fixed-size (page), memory-like interface (such as read page, write page) for manipulating nonvolatile magnetic media. As data block always is the elementary unit for file allocating and data organizing at high application level, Holographic storage must provide block-level interface, such as SCSI, moreover be compatible to traditional file system and higher applications.

The SCSI that is most popular protocol in storage field supports most modern storage formats, regardless of the connection or transport topology used. Because systems can easily package and transport SCSI commands, delivered in command descriptor blocks (CDBs), across many mediums, they can link multiple devices to various types of buses. SCSI's primary asset is the ability to address physical blocks of data on a hard drive without knowing the drive's exact geometry and type. So the holographic storage device can employ the SCSI interface for data accessing in block-level.

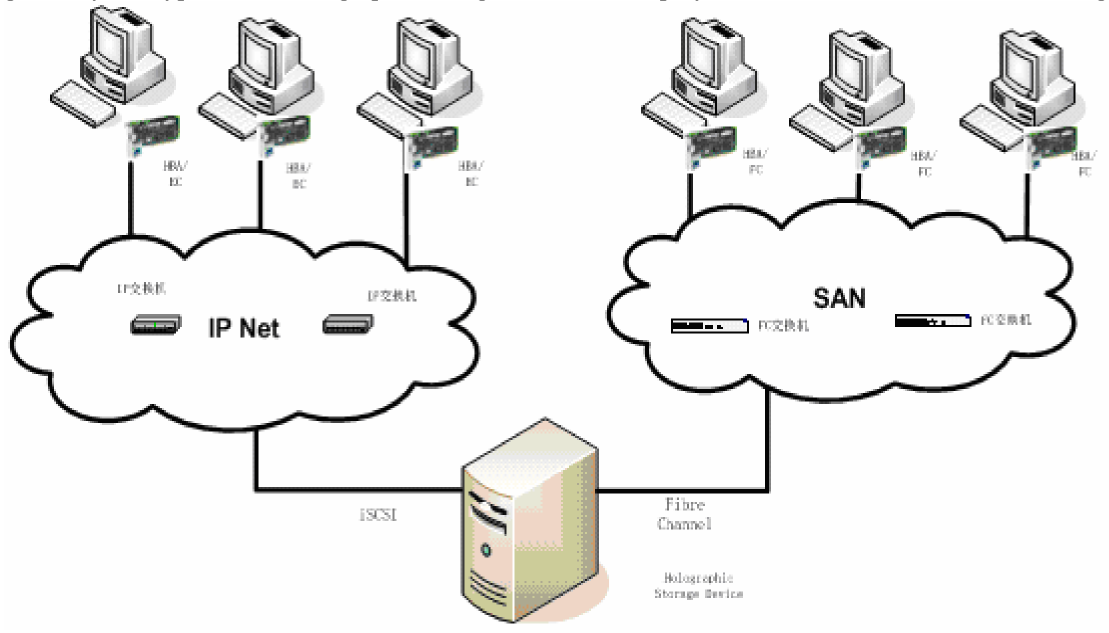

Figure 1 the topology of two network storage systems based on HSD

Parallel SCSI targets discover devices through bus arbitration, whereas devices in a Fibre Channel infrastructure discover through loop initialization or name services. Fibre Channel technology has become the standard for SAN deployments, moving beyond the original arbitrated loop (AL) definitions to switched fabrics that let multiple devices access a large number of port connections. Designed to replace direct attached storage (DAS), most Fibre Channel implementations that began to ship in the mid-1990s broke parallel SCSI's 15-device limitation and permitted much longer distances between host and devices. Serial Attached SCSI (SAS) can extremely extend connected distance and is used in Fibre Channel or iSCSI which are basic of network storage[3].

The main idea behind SAN development is to extend the traditional external bus between a host computer and its storage devices with a high-speed data switched network. In a SAN, the storage device is directly attached to that network, and interacts with multiple host computers using standard network protocols, rather than specialized bus protocols. Network storage opens up the prospect of geographically disperse enterprise-wide storage. With the advent of gigabit per second and higher network transfer rates, the effective transfer rates between hosts and storage increase while costs are decreased. To enable this paradigm, the approach that is being taken is to design new protocols that encapsulate SCSI commands and data for transport over the network. These are collectively referred to as "SCSI Transport Protocols". The first technology to utilize this idea was Fibre Channel, which uses a gigabit net link to carry

SCSI commands and data over distances up to 10 KM. This technology requires special hardware adapters on both the host computers and the target storage devices, and installation of new fiber-optic cabling. To avoid the special hardware requirements of Fibre Channel, several recent proposals have been made to leverage the huge existing network infrastructure, which is largely built on Ethernet at the host computers and which forms the basis for the global Internet. The new storage transport protocols are those that utilize the existing TCP/IP protocol stack. The main proposals in this category are the iSCSI Protocol [4], which becomes currently a Standard by the Internet Engineering Task Force (IETF).

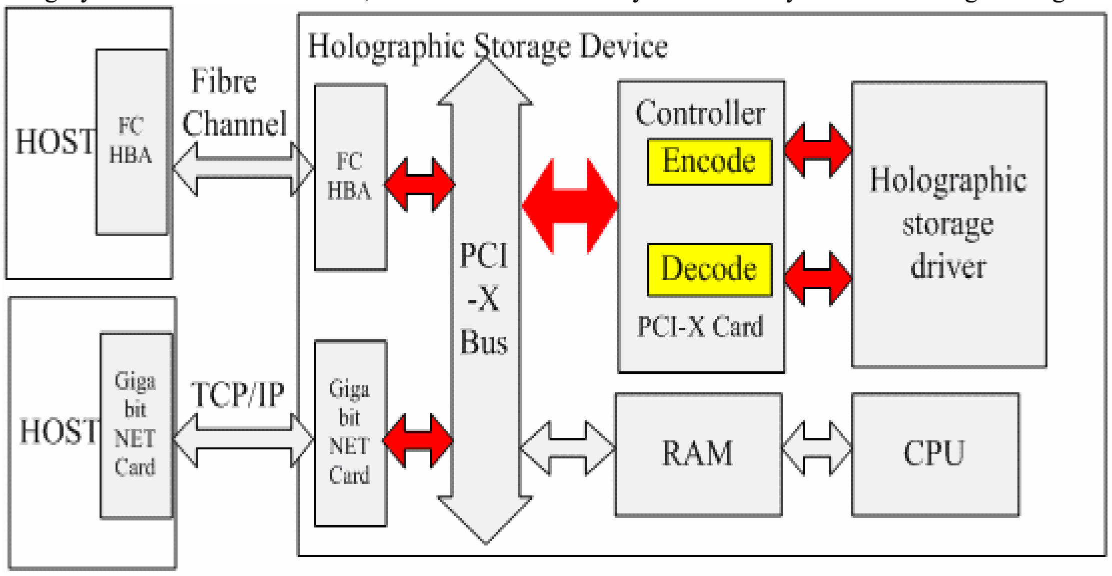

Figure 2 the block diagram of HSD

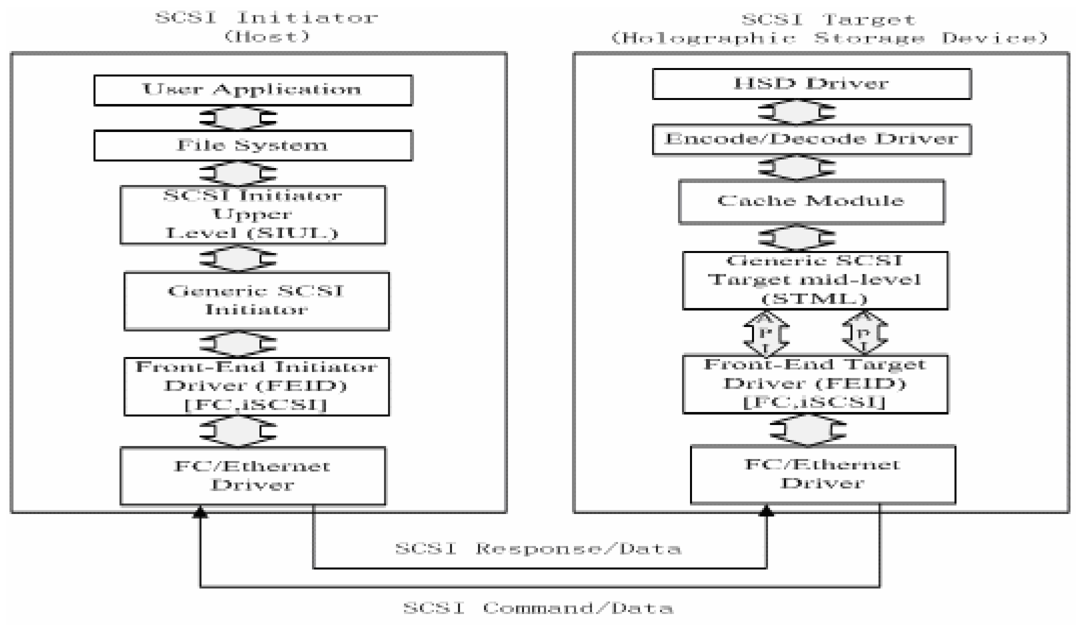
SCSI Initiator (Host)
SCSI Target (Holographic Storage Device)
User Application
File System
SCSI Initiator Upper Level (SIUL)
Generic SCSI Initiator
Front-End Initiator Driver (FEID) [FC,iSCSI]
FC/Ethernet Driver
HSD Driver
Encode/Decode Driver
Cache Module
Generic SCSI Target mid-level (STML)
P I
P I
Front-End Target Driver (FEID) [FC,iSCSI]
FC/Ethernet Driver
SCSI Response/Data
SCSI Command/Data

Figure 3 the emulator software architecture of SCSI Target and Initiator in network storage system

Figure 1 shows the two topologies of network storage system based on HSD. The block diagram of hardware structure in HSD also is shown as figure 2. Two storage interfaces based on FC and iSCSI can be maintained by HSD.

So it is necessary for Fibre channel to use FC HBA and iSCSI to employ Gigabit network card in HSD. For ensuring the entire performance, the PCI-X bus which width can reach 1033MB/s is used to connect these HBAs and other components such as CPU, encode, decode, and so on. Perhaps, the system bus will replace PCI-X bus in future production. The data from host must be encoded before writing to holographic storage driver or decoded for reading process, which ensures the error code rate is less than 10^{-12} that is common in storage system. On the other hand, the controller of Holograph can convert the logical block address in SCSI into physical address in the holographic storage media described by several parameters, such as angle of reference beam.

3. Software Architecture of SCSI Target Emulator

To be able to realize the characteristics of the FC and iSCSI connections, we have developed a prototype FC and iSCSI target mode drivers. The target mode driver makes the HSD running it look like a SCSI disk for the host. Having a linux server emulating a HSD enables us to conduct performance experiments with greater control.

The STML is organized as two threads[6]: the SCSI Target Processing Thread (STPT), which is the main vehicle for processing SCSI commands, and the SCSI Signal Processing Thread (SSPT), which is used to deal with the asynchronous arrival of SIGIO signals, which simply catches the SIGIO and enqueues a command for the STPT, thereby avoiding reentrancy problems in the STPT. An important part of the design of the Target Emulator was specification of an "API" between the STML and the FETD. This API consists of a set of data structures and two sets of functions – one for use by the FETD to hand off incoming SCSI commands and data to the STML, and the other for use by the STML to hand back the generated SCSI responses and data to the FETD.

Table 1 the API functions available for use by an FETD in order to pass SCSI commands and data to the STML

Register_target_template(struct Scsi Target Template *)
Deregister_target_template(struct Scsi Target Template *)
Register_target_front_end(struct Scsi Target Template *)
Deregister_target_front_end(struct Scsi Target Device *)
Rx_cmnd(struct Scsi Target Device *, u64, char *, int)
Scsi_rx_data(struct Target Scsi Cmnd *)
Scsi_target_done(struct Target Scsi Cmnd *)
Scsi_release(struct Target Scsi Cmnd *)
Rx_task_mgmt_fn(struct Scsi Target Device *, int, void *)

Table 1 lists the nine API functions available for use by an FETD in order to pass SCSI commands and data to the STML. The six API functions required by the STML in order to "call back" the FETD are provided in the SCSI Target Template structure. There are three main data structures used by these functions, as SCSI_Target_Device, SCSI_Target_Template and SCSI_Target_Message, which relation are shown in figure 4.

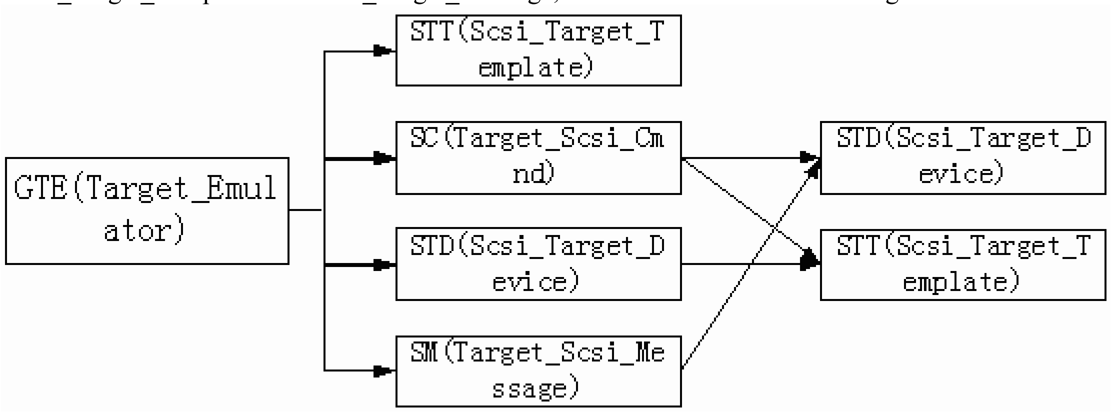

Figure 4 the main relation of three main data structures

A explanation of how the FETD's interact with the STML using the API functions follows. When an FETD module is dynamically loaded into the Linux kernel using the insmod command, the FETD calls register_target_template() to register itself with the STML. Later, when this module is removed, the FETD must call deregister_target_template(). As part of processing the register_target_template() function, the STML will call back the FETD's detect() function in order to detect all “devices” being handled by this FETD. The FETD’s detect() function in turn registers each device with the STML by calling register_target_front_end().

Once registered, an FETD operates asynchronously with respect to the STPT. An FETD device interrupt handler calls the STML’s rx_cmnd() function whenever it receives a new SCSI command from the network. This command will be enqueued for processing by the STPT thread. If the command is a READ-type command, where data is being sent from the Target to the Initiator, the STPT thread allocates buffers and passes the request to the SCSI mid-level subsystem on the Target using the SCSI generic interface. When this subsystem informs the STPT that the data is in the buffers, the STPT calls back the xmit_response() function of the FETD, which is responsible for encapsulating the data and asynchronously sending it over the transport network to the Initiator. Once this transmission is complete, the FETD notifies the STPT by calling the scsi_target_done() function of the STML. This sequence of events is illustrated schematically in Figure 3. Processing a WRITE-type command is somewhat more complicated, because the data from the Initiator will asynchronously follow the SCSI command itself. The necessary sequence of events is illustrated in Figure 4. As in the case of a READ-type command, the STPT thread processes a WRITE by first allocating buffers, but it obviously cannot fill these buffers locally as was done for a READ. Instead, it passes these buffers back to the FETD by calling the rdy_to_xfer() function. This function uses flow control mechanisms unique to the particular SCSI transport protocol to initiate the transfer of data from the Initiator over the transport network. When this data arrives, the FETD passes it to the STML by calling the scsi_rx_data() function, which in turn will pass this data to the SCSI mid-level subsystem on the Target using the SCSI generic interface. When this subsystem informs the STPT that the data transfer to the local SCSI device has been completed, the STPT calls back the xmit_response() function of the FETD, which must send the SCSI status response back to the Initiator. Once this transmission is complete, the FETD notifies the STPT by calling the scsi_target_done() function of the STML.

4. The Process of Accessing Data in HSD

The HSD can not only receive and responses a request from host, but also transfer data back host or write data to media as soon as quickly. From a view of host end, both read and write process can be divided to three obvious phases shown in figure 5 and 6, which are command phase, data phase and response phase.

Host is an initiator, and HSD is target in a SCSI link. The read process can further separate into five stages. At first, Host sends a read command to HSD with host FC adapter. In second step, FC layer in HSD analyzes the data frame and FETD analyzes the SCSI command. Then, after accepting a read command, SMTL allocates the data buffer. Moreover the relative functions of FC-HSD can be called. In the third step, FC-HSD converts LBA into physical address and reads data from holographic media throughout decoding. Moreover FC-HSD fills data into buffer defined by STML. In the fourth step, SMTL begins DMA transfer mode, moreover, sends command and the data back initiator. At the fifth step, after completing data transmitting, the host acknowledges the target success and HSD releases the data buffer. The overall read process is over.

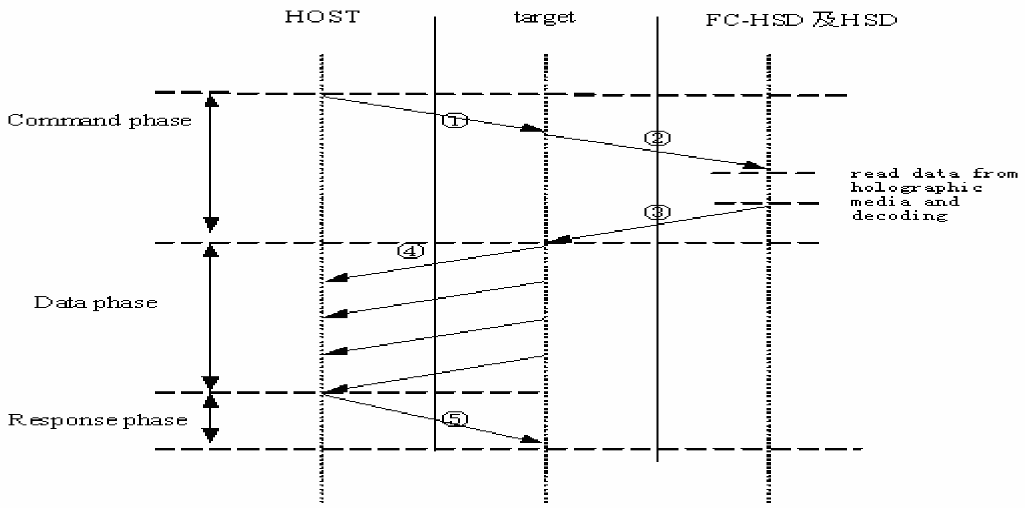

Figure 5 read data process

The process of write is more complicated than read, which includes seven steps. The first step same to read. At the second step, STML allocates the buffer and acknowledges to initiator. In the third step, the host starts to transmit data into the buffer and notifies target. At the fourth step, target send write command to HSD and FC-HSD converts LBA into physical address. At the fifth step, after encoding, data is stored into holographic media and send OK command to SMTL. At the sixth step, success message can pass above three layers orderly. Moreover, SMTL releases the buffer. At the end, the host accepts the success message.

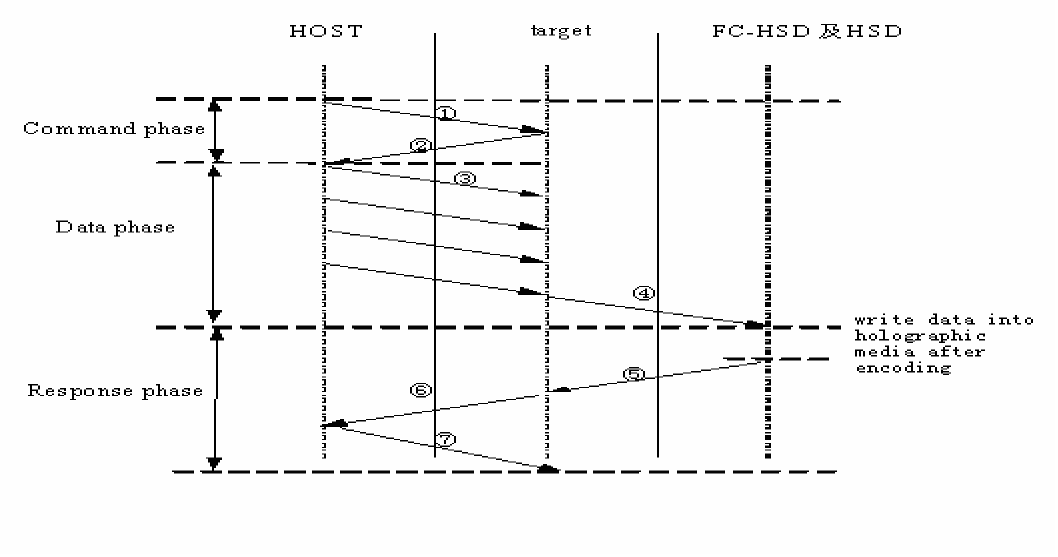

Figure 6 write data process

5. Experiment

In our experiment, a page of CCD is 1024*1024 pixels and the output rate of CCD is 2000 frames/s. Since a bit is defined by two pixels, the bit rate is 64KB/page. The fact size of data of a page is only 55KB because bit-error rate (BER) must be guaranteed less than 10-12 and some bits are used to CRC such as RS code with which 256 bits data contains 32 parity bits. So the maximum sustained sequence readout rate is almost 110MBytes/s for holographic driver. For random access, to angle reference light can locate data address. But a small I/O request which is less than 55KBytes, an entire page must be accessed. For improving performance, the 256MB cache is used because the main purpose of this experiment is test the function of accessing data throughout storage network.

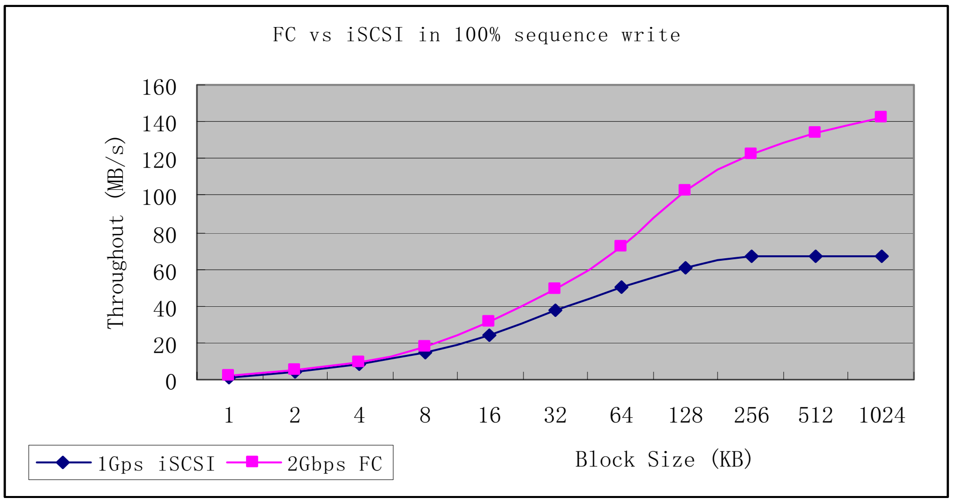

Figure 7 FC vs iSCSI in 100% sequence write

The network storage testbed used in our experiments consists of a HSD and a client connected over an isolated Gigabit Ethernet LAN or 2Gbps Fibre Channel. Our client is a dual processor machine with two 2.4GHz Xeon Intel processors, 512KB L1 cache, 512MB of main DDR memory and a D-Link 550TX Gigabit Ethernet card. The client contains an Adaptec ServerRAID adapter card that is connected to a SCSI disk that is a 15,000 RPM Ultra-320 SCSI drive with 30GB storage capacity. For the purpose of our experiments, we configure the HSD emulated by server, which is a 2.6GHZ Athlon MP processor, 512MB of main DDR memory and a D-Link 550TX Gigabit Ethernet card.

We have implemented three drivers in target side, and have tested two drivers for storage transport protocols. The drivers we implemented, for SEP and iSCSI draft 20, utilize the existing Linux software TCP/IP stack to communicate over 1Gbps Ethernet network. The existing driver was for the Fibre Channel protocol. Both the client and HSD utilize Agilent HBA5221a Fibre Channel HBA communicating over 2Gbps Fibre Channel link.

The client machine runs windows 2003 and the HSD runs version 2.4.18 of the Linux kernel. The test tool is Iometer developed by Intel, which has various parameters to test I/O performance. We test the rates of sequence full write or read in 2Gbps FC and 1Gbps Ethernet respectively. The results are shown as figure 7, 8. Because the cache is used between STML and accessing media, the read rate is better than write. On the other hand, Fibre channel has more width than Ethernet. So the performance of FC represents better than iSCSI. However, the results show the Holographic storage device can work well in network storage environment.

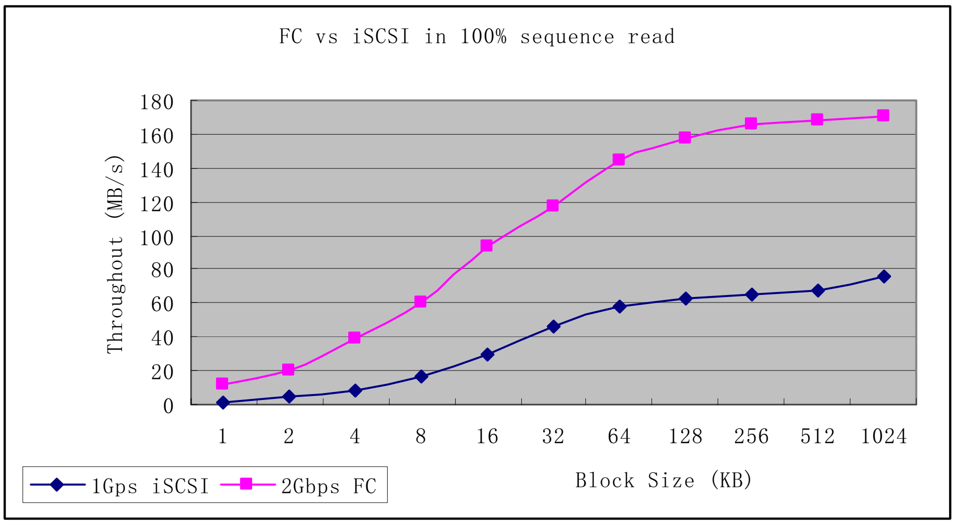

Figure 8 FC vs iSCSI in 100% sequence read

Reference

[1] Garth A. Gibson，Rodney Van Meter．Network Attached Storage Architecture．IEEE Internet Computing, 43(11), 2001：50 -58

[2] Demetri Psaltis, Geoffrey W. Burr, Holographic Data Storage, IEEE Computer, 1998, Feb:pp52-61

[3] Robert Griswold．Storage Topologies．IEEE Computer, 35(12)．Dec, 2002:56-63

[4] Stephen Aiken, Dirk Grunwald, Andy Pleszkun．A Performance Analysis of the iSCSI Protocol. IEEE/NASA MSST2003．Twentieth IEEE/Eleventh NASA Goddard Conference on Mass Storage Systems & Technologies．April, 2003, San Diego, California, USA:135-165

[5] Qiang Cao, Changsheng Xie. The Study of the I/O Request Response Time in Network Storage System, the Journal of Computer Research & Development, 40(8)，2003:1271

[6] Ashish Palekar, Narendran Ganapathy, Design and Implementation of a Linux Scsi Target for Storage Area Networks, Proceedings of the 5th Annual Linux Showcase & Conference, Oakland, California, USA, November 5–10, 2001