Performance Study of iSCSI-Based Storage Subsystems

Yingping Lu and David H. C. Du, University of Minnesota

ABSTRACT

iSCSI is emerging as an end-to-end protocol for transporting storage I/O block data over IP networks. By exploiting the ubiquitous Internet infrastructure, iSCSI greatly facilitates remote storage, remote backup, and data mirroring. This article evaluates the performance of two typical iSCSI storage subsystems by measuring and analyzing block-level I/O access performance and file-level access performance. In the file-level performance study, we compare file access performance in an NAS scheme with that in an iSCSI-based SAN scheme. Our test results show that Gigabit Ethernet-based iSCSI can reach very high bandwidth, close to that of a direct FC disk access in block I/O access. However, when the iSCSI traverses through longer distance, throughput relies heavily on the available bandwidth between the initiator and the target. On the other hand, the file-level performance shows that iSCSI-based file access (SAN scheme) provides higher performance than using NFS protocol in Linux and SMB protocol in Windows (NAS scheme). However, the advantage of using iSCSI-based file accesses decreases as the file size increases. The obtained experimental results shed some light on the performance of applications based on iSCSI storage.

INTRODUCTION

After the 9/11 terrorist attack, it was observed that even though a server machine may be very robust and highly fault-tolerant, it is still vulnerable to an unexpected catastrophe such as a natural disaster or planned attack. All of the stored data and computing equipment could be wiped out instantly. In order to make mission-critical data (e.g., business data, banking data) operational even after a devastating strike, it is highly imperative to have remote data backup and remote disaster recovery enabled.

Currently, the most widely used storage access protocol is SCSI [1]. The SCSI protocol supports several underlying I/O interconnects. The parallel SCSI bus is the most widely used interconnect between SCSI devices and a host. However, parallel SCSI has several limitations. The fundamental limitations are distance and the number of storage devices that can be attached. A parallel SCSI bus can only stretch to several meters and attach at most 16 disk devices. These limitations restrict the scalability of a storage system.

Fibre Channel has evolved and played a major role in storage area networks (SANs) [2, 3] due to its high performance and low overhead. It also considerably increases the number of attached disks, and the distance between hosts and disks. In fact, one Fibre Channel-Arbitrated Loop (FC-AL) can attach 126 nodes (disks and/or hosts). By connecting multiple FC-ALs into Fibre Channel switching fabric, a Fibre Channel network can attach a large number of disks and hosts. In addition, Fibre Channel also facilitates storage and data sharing since disk devices are not dedicated to a single host. Multiple hosts can share data in a Fibre Channel storage subsystem. Nonetheless, the distance between a host and a Fibre Channel disk device is still restricted to a local area.

With prevalent deployment and accessibility, IP networks can be exploited to serve as storage data carriers to extend the storage system. Emerging as an end-to-end protocol for transporting storage I/O block data over IP networks, iSCSI encapsulates disk access requests (in the form of SCSI CDB commands) into TCP packets, and transmits the SCSI commands and block-level data over IP networks. Thus, iSCSI encompasses two major protocols: for storage access, SCSI, and for networking transport, TCP. It extends the SAN network to a remote area and enables new applications like data mirroring, remote backup, and remote management. It also unifies the storage and data networks, thus greatly reducing management cost.

However, TCP/IP is inherently slow due to several factors such as checksum generation, protocol processing, memory copy, and context switching. In addition, since the IP network is not a secure network, it is crucial to have the iSCSI commands and data traversing an IP network to be encrypted. This adds more overhead and further exacerbates the performance of iSCSI [4]. Thus, it is very important to understand the performance characteristics of the iSCSI storage subsystem and its impact on applications.

This work is partially supported by a gift from Intel and Digital Technology Center Intelligent Storage Consortium.

0163-6804/03/$17.00 © 2003 IEEE

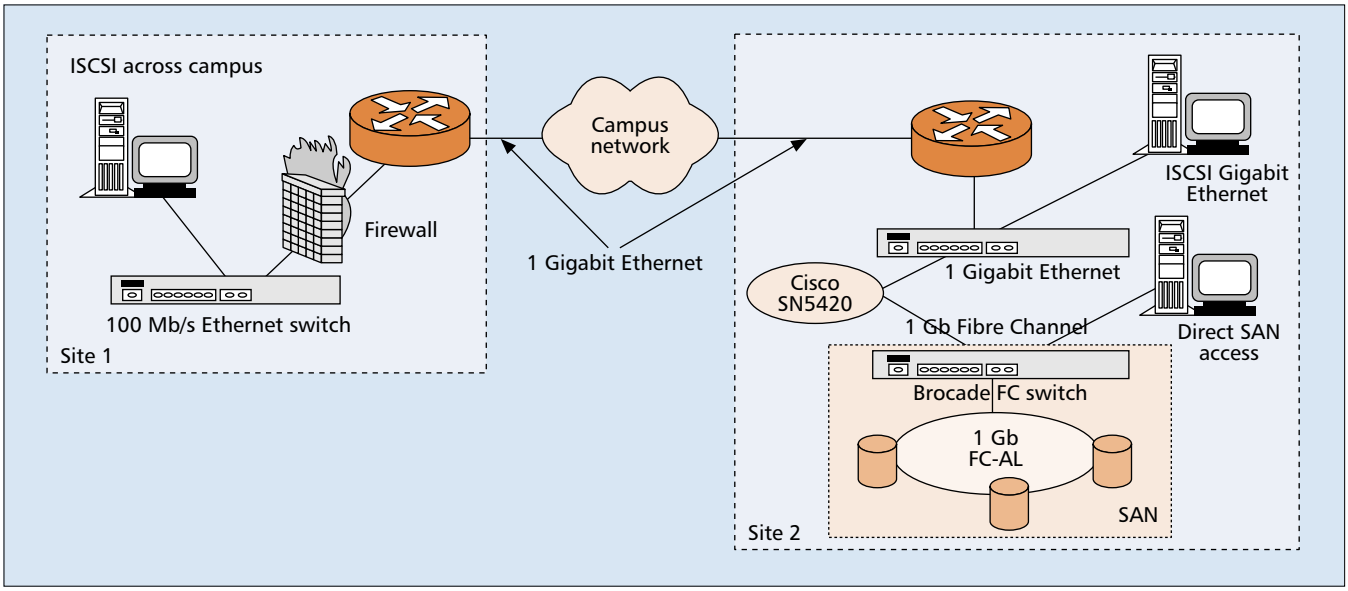

Figure 1. *The testbed environment.*

In this article we are interested in the real performance of typical iSCSI storage subsystems, the factors that affect the iSCSI storage performance, and where the performance bottleneck lies. To look into these, we set up a testbed and conduct real performance measurements for two typical iSCSI subsystem configurations. We examine both block-level raw I/O performance and file-level I/O performance.

Testbed Setting

In this section we describe the testbed environment, and the measurement methods and performance metrics used in the experiments.

The Initial Setup Process

An iSCSI client (initiator) needs to complete the setup process with the storage device (target) before iSCSI commands and data can be transferred. This is normally performed by the iSCSI driver in the client machine during the system boot process. We use an Ethereal network analyzer [5] to capture all the commands and data exchanged between the initiator and target in the initial setup process. We then use the captured information to analyze the setup process. The details are discussed later.

Block-Level Raw I/O Performance Measurement

Since the iSCSI protocol is a block-level data transfer protocol, our first measurement targets block-level raw I/O performance. In this study, we access the target disk by sending block-level SCSI commands for read/write operations. Two configurations for iSCSI storage are examined. One configuration is iSCSI storage with Gigabit Ethernet connection. This scenario reflects a best case, where an iSCSI initiator is close to the target and the communication speed is very fast, up to 1 Gb/s (referred to as *GigE iSCSI disks*). A typical case of this scenario can be found in a data center environment. The other configuration reflects a more common case, in which iSCSI storage resides remotely (referred to as *campus iSCSI disks*). In this scenario, an initiator connects to the target devices through the campus IP network. To meet the security requirement of iSCSI, this scenario assumes that the underlying security measures have been provided through a virtual private network (VPN). To protect the internal network where the initiator resides from malicious attack, a firewall is also deployed for the protection of the initiator. Furthermore, in order to compare the performance of non-iSCSI storage, we also examine another configuration, a directly attached FC SAN. We compare the performance with its iSCSI peers.

Figure 1 shows the testbed. This testbed spans two sites connected through the campus network. The speed in the campus backbone network is 1 Gb/s. In site 2, the FC-AL disks are connected to an FC-AL loop, which again connects to a Brocade FC switch. Both the Fibre Channel and Ethernet have the speed of 1 Gb/s. However, the test machines in Site 1 only have 100 Mb/s bandwidth. They connect to the campus backbone network through a firewall which has a pre-established VPN with site 2.

To provide iSCSI access from the IP network to the FC-AL disks in the FC SAN, we use a Cisco SN5420 storage gateway in site 2 to connect these two disparate networks. Cisco SN5420 provides a bridge between the iSCSI protocol and the FCP protocol. It also provides target mapping and access control over the targets. A target is the basic unit with assigned permission in the storage gateway, with which an iSCSI initiator with proper access can discover and communicate. A target contains one or more physical disks[1] behind the gateway. The access list in the storage gateway is used for access control.

Two metrics are used in this performance study:

- Throughput: the sustained data transfer rate between an initiator and the storage devices.
- Response time: the time spent for an initia-

[1] *Logical disks are also possible; for example, a disk can be a RAID disk containing a group of physical disks.*

The heavy load performance study can help us understand the maximal throughput the storage subsystem can offer and locate the potential system bottleneck. In addition, we also conduct performance measurements based on the database trace data that we obtained from IBM.

Parameters	Value
Model	Seagate 39102FC
Interface	Fibre Channel
Capacity	9.1 Gbytes
Cache size	1 Mbyte
Rotational speed	10,025 RPM
Avg. rotational time	3.0 ms
Seek time	Read 5.4 ms, write 6.2 ms
Internal transfer rate	19.0–28.4 MB/s

Table 1. *Disk parameters.*

tor to issue an iSCSI command till the request has been served by the target. This time is also called latency. We use these two terms interchangeably.

In this raw I/O performance study, we conduct both light load performance and heavy load performance tests. In the light load test, we only issue one SCSI command at each time. The light load performance study can help us understand the basic characteristics of the iSCSI protocol. On the other hand, in the heavy load test, we use 64 threads to imitate multiple clients generating heavy load requests to the target. The heavy load performance study can help us understand the maximal throughput the storage subsystem can offer and locate the potential system bottleneck. In addition, we also conduct performance measurements based on the database trace data that we obtained from IBM. This helps us to understand how iSCSI storage performs under a real database access pattern.

File Access Performance Measurement

In the file-level performance study, we compare performance between two file access schemes: remote shared file system vs. iSCSI storage.

These two schemes represent two storage access paradigms. The first scheme represents one paradigm, network attached storage [6] (NAS), where the server provides file-level access to its clients. The iSCSI scheme represents another paradigm: SAN, or SAN over long distance, where a client accesses a target through the block-level SCSI protocol. There are two popular file systems in NAS: Microsoft Common Internet File System (CIFS) and Sun Network File System (NFS). The former uses Server Message Block (SMB) protocol to communicate between a client and a server in Windows platform, while the latter uses NFS protocol in UNIX platform. We compare the file access performance in both the CIFS and NFS environments with that in the iSCSI scheme.

Disk Model

In all the tests, we use the same group of disks to make the result comparable. The disks used in the tests are Seagate Fibre Channel disks (model 39102FC). They are placed into a Fibre Channel JBOD.[2] Table 1 shows the disk parameters for this disk model.

iSCSI Initial Setup Process

iSCSI is a session-based protocol. The communication between an initiator-target pair should occur within the context of a session. A session can encompass one or multiple TCP connections. There are four phases in a session, where the first phase, login, starts with the establishment of the first TCP connection. The four phases are:

1) Initial login phase: In the initial login phase, an initiator sends the name of the initiator and target, and specifies the authentication options. The target will respond with the authentication options the target selects.

2) Security authentication phase: To ensure that each party is actually talking to its intended party, this phase is used to exchange authentication information (ID, password, certificate, etc.) based on the agreed authentication methods. The authentication can occur both ways. That is, a target can authenticate an initiator, and an initiator can also request the authentication of the target.

3) Operational negotiating phase: This phase is used to exchange certain operational parameters such as protocol data unit (PDU) length and buffer size.

4) Full featured phase: This is the normal phase of an iSCSI session where iSCSI commands, and data messages are transferred between an initiator and a target.

Note that phases 2 and 3 are optional.

In addition to the session establishment, an initiator needs to discover the targets in the first place. There are three ways to find out the intended targets. The first way is a static configuration where an initiator is statically configured with the IP address of the target node. This is suitable for small networks (e.g., an intranet in a company with dozens of computers connected). The second discovery approach is to use the Service Location Protocol (SLP). This is appropriate for medium-sized networks (e.g., enterprise networks). For a large network like the Internet, the Internet Storage Naming Service (iSNS) can be applied to discover storage targets. In our test environment, we use the static configuration due to its small scale and simplicity.

We measure the setup time from the test machine in site 1, which traverses the campus network to reach the target. The test machine is a Dell Optiplex Gx150 with 1 GHz of Pentium III CPU running Windows 2000. The iSCSI driver is the Cisco iSCSI driver (v. 2.1.2). We force the iSCSI driver to relogin and capture all network traffic using the Ethereal network analyzer. The elapsed time for each step is obtained by subtracting the start time of the next step to the start time of the current step. We conduct this process five times and take the average value as the elapsed time of each step. The following are the phases and time spent in each phase in the actual initial session establishment process.

Initial discovery session (average: 10,003 ms, minimum: 10,000 ms, maximum: 10,005 ms[3]):

- TCP connection to gateway (listening on port 3260) (average: 211 ms, minimum: 79 ms, maximum: 478 ms).
- Initial login phase (average: 500 ms, minimum: 500 ms, maximum: 501 ms). The initiator sends its login name and negotiates

[2] *Acronym for"Just a Bunch Of Disks," here we indicate the cabinet for holding the FC-AL disks.*

[3] *Represents the measured average, minimum, and maximum time, respectively.*

	Read (ms)				Write (ms)			
Configuration	1 kB	4 kB	16 kB	64 kB	1 kB	4 kB	16 kB	64 kB
SAN	9.04	9.19	10.21	14.38	9.55	9.72	10.54	13.57
GigE LAN iSCSI	9.57	9.7	10.91	15.82	10.08	10.35	11.24	15.0
Campus LAN iSCSI	11.37	12.45	14.33	24.2	12.8	12.5	14.62	22.98

Table 2. *Latency under light load; kB: kilobytes.*

	1 kbyte (ms)			4 kbytes (ms)			16 kbytes (ms)			64 kbytes (ms)		
Configuration	Min	Max	Std	Min	Max	Std	Min	Max	Std	Min	Max	Std
SAN	1.7	26.8	2.6	2.1	28.6	2.6	4.3	30.3	2.7	8.0	34.8	2.7
GigE LAN iSCSI	2.1	37.8	2.8	2.5	94.9	3.1	4.9	84.0	3.2	9.8	50	3.1
Campus LAN iSCSI	4.4	58.3	2.7	5.3	103	2.8	8.4	91	2.8	20.2	126	3.1

Table 3. *Latency variance of read operation under light load.*

the authentication option. Both parties agree on no authentication and digest. Thus, the security authentication phase is skipped.

- Operational parameters negotiating phase (average: 500 ms, minimum: 500 ms, maximum: 501 ms). Buffer size, data PDU size, and other parameters are negotiated.
- Discovery phase (average: 8794 ms, minimum: 8522 ms, maximum: 8923 ms). The initiator sends the SendTargets=all Text command to the target node to get all accessible targets. In our case, we only find the target fcaldisks.

Session establishment for each target (average: 10,136 ms, minimum: 10,014 ms, maximum: 10,205 ms):

- TCP connection setup for the session corresponding to the new target (average: 718 ms, minimum: 590 ms, maximum: 991 ms).
- Initial login again for the session(average: 501 ms, minimum: 500 ms, maximum: 501 ms). The negotiating result is the same as previously negotiated. No authentication and digest is needed, and hence, no security authentication for each target connection.
- Operational parameter negotiating phase for the session (average: 8807 ms, minimum: 8522 ms, maximum: 8986 ms).
- Full featured phase for the session (average: 159 ms, minimum: 118 ms, maximum: 190 ms). In this phase, the initiator collects device and capability information of each device (LUN), reads its master boot record information. These are performed through standard SCSI commands (via CDB).

As can be seen, the whole setup process is very time-consuming. More than 20 s are spent before an initiator can really access data from a target device. However, as we examine deeper, we find this long establishment might rely heavily on a particular implementation. From the captured data, we find the actual interaction time between the initiator and target in each step is pretty short. For example, in one captured data, during the 8710 ms of the discovery phase, only 200 ms is used for the exchange of the iSCSI targets and TCP acknowledgment information. There is no iSCSI traffic during the rest of time (we suspect the driver may have some timer setting, e.g., 10 s, to shift from one session to another). Also note that these data only reflect the setup time for this particular setting. For example, if the authentication phase occurs, the perceived login time will increase. The number of targets also affects the number of sessions, the number of TCP connections, and hence the login time for these connections. Nonetheless, these data provide a coarse range of the time needed for the session establishment. Although this initial setup cost is pretty high, this cost is not recurring and only happens when a session is established, normally after a system reboots.

As can be seen, the whole setup process is very time-consuming. More than 20 s are spent before an initiator can really access data from a target device. However, as we examine deeper, we find this long establishment might rely heavily on a particular implementation.

Performance Analysis of Raw I/O Block Access

In this section we present the performance measurements of low-level raw I/O accesses. We show the performance under light load, heavy load, and trace data from an IBM database application. These measurements are conducted in Windows platforms running Windows 2000. The test code is developed on Microsoft Windows Device Driver Development Kit (NTDDK). The block-level IO access is achieved through DeviceIOControl calls that directly pass the access requests to the SCSI driver. This bypasses the file system, and thus reduces the overhead and uncertainty brought by a file system. Meanwhile, each test lasts for 5 min, which results in tens of thousands of runs for each test. During the test, we generate random logical block addresses (LBAs) within the whole disk block address space. We also generate random disk ID in each run if multiple disks are involved.

Performance under Light Load

Table 2 shows the average read/write latency under light load. Due to the space limitation, Table 3 shows the minimum latency, maximum latency, and standard deviation (std) only for the read opera-

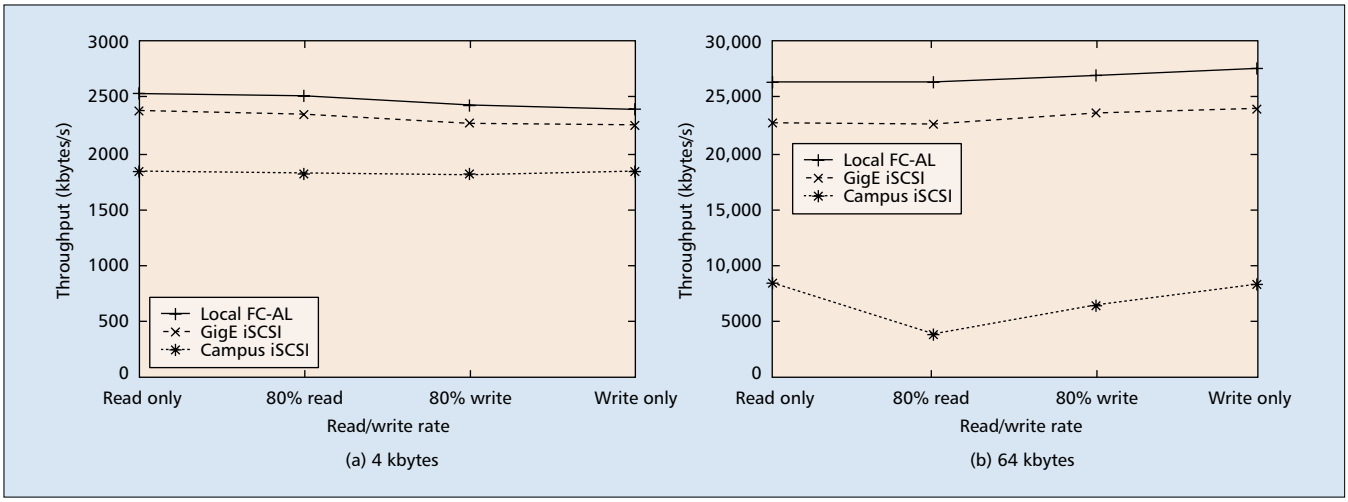

Figure 2. *Throughput of read/write under heavy load.*

tions. The latency in light load reveals the shortest response time a client can achieve since there is little waiting time for each request. As can be seen, among the three configurations, local SAN has the lowest latency. The iSCSI disk access in Gigabit LAN also has a very low latency, close to the corresponding latency of local SAN access. However, the disk access to iSCSI disks across the campus has a much higher latency. For example, when the data size is 64 kbytes, its latency is 70 percent more than the latency of accessing a local SAN disk.

In order to better examine the constituents of the access latency, we can break down the data path of the iSCSI access into two parts (channels) at the storage gateway. The first one is the iSCSI channel where commands, and data are transmitted over the IP network running the iSCSI protocol. The second one is the FCP channel where the FCP protocol is used to access disks in the SAN. From the table, we observe that iSCSI over Gigabit LAN has a relatively lower latency as compared with iSCSI over the campus network. The difference becomes larger with the increase of data size. This is because iSCSI over Gigabit LAN has a lower transfer time in the IP channel given the same transfer time in the FC channel. The IP network bandwidth between the test machine on site 2 and the gateway is 1 Gb/s, while the available bandwidth between the test machine in site 1 and the gateway is only 100 Mb/s (the bottleneck is the link between the test machine and the router in site 1). Moreover, the security measure (VPN) between them adds more overhead. Thus, the campus iSCSI storage has a much lower bandwidth and higher latency.

Generally speaking, a write operation takes a longer time than a read operation since the seek time for a write operation is a little longer (around 0.8 ms on average) than a read operation. This is demonstrated when the data size is small. For example, when the data sizes are 1 kbytes and 4 kbytes, the latency of the read operation is 0.5 ms less than that of the corresponding write operation for the direct SAN access. However, when the data size grows larger, the latency difference becomes smaller. At a data size of 64 kbytes, the write latency becomes smaller than the corresponding read latency. This phenomenon is caused by the external transfer mechanism in the FC-AL disk. The Seagate 39102FC disk has 1024 kbytes of memory of which 967.5 kbytes are used as cache. For the read operation, a disk device requires to win the FC channel arbitration before transmitting data. In order to reduce the arbitration overhead, the data is transmitted only after a certain portion of requested data is extracted from a disk. The percentage is determined by a parameter BUFFER_FULL_RATIO in the Mode Select Disconnect/Connect Control Page, the default value is 80. Thus, in our case, there is little data overlapping between the internal disk transfer and external FC-AL transfer. However, for the write operation, when the emptiness of a disk buffer reaches a specified BUFFER_EMPTY_RATIO (also in the Disconnect/Connect Control Page), which normally is easy to satisfy, the device sends receiver ready (XFER_RDY) to the initiator and starts to arbitrate. After the disk starts to receive data from an initiator (the host or gateway), it writes immediately, thus saving time on the data delivery path. This time saving becomes more prominent when the data size becomes larger.

From the std data in Table 3, we find that the variances are actually pretty small (i.e., less than 3 ms). We also see that the variances do not show much difference among the three scenarios. This might be attributed to light load in all the three scenarios. In the current configuration, the campus network is also in an overprovisioning state (bandwidth is not fully utilized). However, their min and max values reveal that the direct access to Fibre Channel has a better predictable response time for an individual request. Its latency range is much smaller than that of iSCSI access (both the GigE and Campus iSCSI).

Performance under Heavy Load

In this test we study how different operations affect performance under heavy load. Four scenarios are compared: pure read, read dominated access, write dominated access, and pure write. In the second scenario 80 percent of operations are reads and 20 percent are writes. In the third scenario, 80 percent of operations are writes and 20 percent are reads. Since the data size has considerable effect on performance, we also con-

duct tests on two data sizes, 4 kbytes and 64 kbytes, which represent the requests of small data size and large data size respectively. There are six disks involved in the test.

Figures 2a and 2b display the throughput with data sizes of 4 kbytes and 64 kbytes, respectively. During the test, the logical block address and the accessed disk are randomly generated. The operation is also randomly generated but follows the distribution of the specified percentage. The throughput is the aggregate throughput of all threads.

As seen from the figures, in the case of 4 kbytes, due to the higher seek time for the write operation, the pure read access performs the best, the read dominated access follows, while pure write access attains the lowest performance. However, the difference is not significant. For example, in all three configurations, the throughput difference between pure read and pure write is less than 5 percent. On the other hand, for a data size of 64 kbytes, the performance relationship reverses. The pure write access achieves highest throughput while the pure read achieves lowest throughput. This is because potentially more parallel transmissions occur between internal and external channels for the disk write access. We have not enabled the write back caching, read caching, and prefetching. Those features will definitely improve the achieved performance. However, the impact will largely rely on the particular applications and data access patterns.

From the figures, we also notice that the three configurations can achieve around 26 Mbytes/s, 24 Mbytes/s, and 8 Mbytes/s of throughput, respectively, at the data size of 64 kbytes, while only around 2.5 Mbytes/s, 2.3 Mbytes/s, and 1.8 Mbytes/s, respectively, at the data size of 4 kbytes. At the smaller data size, the achieved throughput is far less than the available network bandwidth. Theoretically, at the data size of 4 kbytes, each individual disk's I/O bandwidth is less than 0.5 Mbyte/s. Thus, the disk I/O becomes a bottleneck to achieve high throughput. However, at a large data size, for the campus network-based iSCSI the throughput reaches 65 percent of the network bandwidth. Taking the overhead of TCP processing, data encryption, and firewall, and the interference of cross traffic into consideration, we believe this throughput is approaching its limit. Thus, network bandwidth becomes the bottleneck for performance improvement.

Performance of Trace Data

We then conduct performance measurement based on the trace data. The trace data came from IBM. It was collected from an eight-disk RAID system meant for database access. The total number of requests is 8626. Among these requests, 70 percent of operations are reads. The LBA range is 9840–2,012,792.

The original data has the elapsed time between consecutive requests. However, in our current test, we ignore arrival time, but focus on the access pattern, and study the performance achieved in the three configurations. In order to maintain proper load, we generate 32 threads to issue the requests in the order of trace data. We then measure the aggregate throughput and the average response time per request.

The test result shows that the throughputs for the three configurations (campus, GigE, FC-AL) are 3.6 Mbytes/s, 8.3 Mbytes/s, and 9.2 Mbytes/s, respectively. The response times are 107 ms, 47 ms, and 42 ms, respectively. As can be seen, the direct FC-AL access achieves the highest throughput and lowest response time. The Gigabit-based iSCSI disk access can reach 90 percent of the FC-AL throughput, while campus-based iSCSI disk access can only reach 40 percent of direct FC-AL access throughput. The data is pretty consistent with our previous tests.

Data size	1 kB (ms)		4 kB (ms)		16 kB (ms)		64 kB (ms)		256 k (ms)	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
iSCSI read	18.5	27.1	20.5	25.4	34.8	9.0	49.1	10.5	93.5	14.6
SMB read	19.1	28.4	22.1	25.1	40.0	8.8	62.3	16	125.1	19.4
iSCSI write	22.2	29.0	22.2	24.6	40.3	26.2	44.7	19.5	102.4	34.3
SMB write	50.1	8	50.6	8.5	53.3	8.3	81.2	17.1	192.5	50.7

Table 4. *File access latency comparison (SMB vs. iSCSI); kB: kbytes.*

Performance Analysis of File-Level Access

In this section we conduct latency comparison on the file access level. We measure the latency for accessing different file sizes in different schemes and different platforms. During the test, for each file size we run file accesses 100 times (each time a different file with the same file size is accessed) and take the average value as the latency.

SMB Protocol vs. iSCSI Protocol

Table 4 displays the latency and latency variance on the access of file objects within these two schemes in the Windows platform: file access through SMB and iSCSI. The local file system for the file accesses is NTFS. The latency covers the file open time and file read/write time. As can be seen, for the read operation, when the file size is small, the file access time of the iSCSI-based read is slightly shorter than the time in the SMB-based read. The difference becomes significant when the file size grows larger. At the size of 256 kbytes, the read operation of the iSCSI scheme takes 93.5 ms, while the SMB read takes 125 ms.

The difference in latency becomes more prominent in the write operation. Even with small file size (e.g., 1 kbyte), the iSCSI-based write spends 22 ms, while the SMB-based write spends 50 ms. When the file size becomes larger, the gap becomes much larger. For a file size of 256 kbytes, the difference is 90 ms. Clearly we observe that iSCSI-based file access achieves lower latency than networking file access.

Compared to the block level access under light load, the results also show that the file access of both schemes has a much higher variance. For example, at small file size, the variance can be larger than the average latency. It is also difficult to determine which scheme behaves more consistently based on the results.

NFS Protocol vs. iSCSI Protocol

In the Linux platform, the remote file system is an NFS server (v. 2.0). The client side is the same

Data size	1 kB (ms)		4 kB (ms)		16 kB (ms)		64 kB (ms)		256 k (ms)	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
iSCSI read	4.36	3.6	4.43	4.4	7.13	3.7	19.89	3.5	61.04	5.3
NFS read	9.00	2.4	9.80	2.1	12.07	2.2	25.63	2.8	75.34	9.3
iSCSI write	16.34	4.0	17.68	2.7	21.12	4.7	38.28	3.3	161.4	9.6
NFS write	39.58	3.5	41.0	3.6	44.79	2.6	67.34	3.6	197.24	5.5

Table 5. *File access performance comparison (NFS vs. iSCSI); kB: kbytes.*

machine as in the Windows platform; however, it runs Redhat Linux (kernel 2.4.18-3). The iSCSI driver is Cisco iSCSI driver for Linux (version 2.1.2.1). We disable the buffering effect and use TCP protocol for NFS communication. The file system used for the test is ext2. Table 5 displays the latency on the access of file objects with the NFS and iSCSI schemes, respectively. The tests are based on synchronous access without caching.

For the write operations, we find the iSCSI scheme outperforms the NFS scheme. When the packet size is small (less than 64 kbytes), the iSCSI scheme has about only half the latency of the NFS scheme. For example, the NFS scheme takes 44.79 ms to write a file of size 16 kbytes, while the iSCSI scheme only takes 21.12 ms. The results also show that the variance of latency of both schemes in the Linux platform is pretty small as compared to the Windows counterpart. As can be seen, the read operations also have a similar trend.

The big difference in latency between these two schemes lies in the networking file access protocol. For each file access, since NFS is a stateless protocol and runs on the file level, it requires the exchange of several commands before data communication can occur. For example, to read a file the client side needs to send an ACCESS command several times to check the access permission for the file's directories. It then issues a command to look up the specified file and checks its access permission by sending an ACCESS command. In the iSCSI scheme, the client side retains the states of the files and its directories. This saves several round-trip times. Also, iSCSI uses TCP, while NFS uses RPC over TCP;[4] for large data transfers, the iSCSI transfer is a little more efficient.

In the above-mentioned tests, we only measure latency in the raw data path (i.e., we disable the buffering effect). However, in reality, both schemes support caching. The file system built on iSCSI has its internal buffer to cache the accessed data. In fact, we observe that when the same set of data are read out the second time, the data are delivered immediately; no networking activity occurs. On the other hand, the NFS file server can also cache data. The NFS client will also have caching capability in the new version. Nonetheless, the latency in the raw data path reveals the advantage of the iSCSI-based file access scheme.

Conclusion

iSCSI mingles the two most mature technologies, TCP/IP networking technology and SCSI storage technology, and enables remote storage over ubiquitous IP networks. In this study we examine three typical storage organizations and analyze their performance overhead. We conducted both block-level and file-level performance measurements. We show that iSCSI storage with Gigabit connection could have performance very close to directly attached FC-AL storage. We also observe that campus iSCSI storage can achieve reasonable performance, but is restricted by the available network bandwidth. This measurement experience helps us better understand the characteristics of iSCSI storage and can be used to identify performance bottlenecks. In the file-level performance measurement, we measured and compared the file-level access performance on NAS and SAN schemes. We found that the iSCSI-based SAN scheme can outperform the access through NAS schemes in both Windows and Linux platforms. However, the advantages of iSCSI lessen as file size increases. We plan to further study the effect of IP dynamics on iSCSI access and the application of iSCSI protocol.

[4] *By default, the underlying transport protocol of NFS is UDP.*

Acknowledgment

We are grateful to Jim MacDonald for his invaluable assistance. He helped us set up the test environment and ensured proper operation of the testbed. We thank IBM for providing the trace data. We also greatly appreciate the anonymous reviewers for their valuable comments that benefit this article considerably.

References

[1] F. Schmidt, *The SCSI Bus and IDE Interface: Protocols, Applications and Programming*, Addison-Wesley, 1997.
[2] ANSI X3.272-1996, "Fibre Channel-Arbitrated Loop (FC-AL)," Rev. 4.5, June 1, 1995.
[3] D. H. C. Du *et al.*, "Two Emerging Serial Storage Interfaces for Supporting Digital Libraries: Serial Storage Architecture (SSA) and Fibre Channel-Arbitrated Loop (FC-AL)," *Multimedia Tools and Apps.*, 10, 2000, pp. 179–203.
[4] S.-Y. Tang, Y.-P. Lu, and D. H. C. Du, "Performance Study of Software-Based iSCSI Security," *1st Int'l. IEEE Security in Storage Wksp.*, Dec. 2002.
[5] Ethereal Network Analyzer, http://www.ethereal.com
[6] G. A. Gibson and R. Van Meter, "Network Attached Storage Architecture," *Commun. ACM*, vol. 43, no. 11, Nov. 2000.

Additional Reading

[1] J. Satran *et al.*, iSCSI Internet draft, draft-ietf-ips-iscsi-18.txt, IETF, May 2002.

Biographies

YINGPING LU (lu@cs.umn.edu) received his B.S. and M.S. degrees in computer science from Hunan University, China, in 1985 and 1988, respectively. He is currently a Ph.D. candidate at the Computer Science and Engineering Department, University of Minnesota, Minneapolis. His research interests include high-speed networking, high-performance cluster systems, storage area networks, and multimedia computing.

DAVID H. C. DU [F] received a B.S. degree in mathematics from National Tsing-Hua University, Taiwan, China, in 1974, and M.S. and Ph.D. degrees in computer science from the University of Washington, Seattle, in 1980 and 1981, respectively. He is currently a professor in the Computer Science and Engineering Department, University of Minnesota, Minneapolis. His research interests include multimedia computing, storage systems, high-speed networking, high-performance computing over clusters of workstations, database design, and CAD for VLSI circuits. He has authored and co-authored more than 160 technical papers, including 75 refereed journal publications in his research areas. He was an Editor of *IEEE Transactions on Computers* from 1993 to 1997. He currently serves on the editorial boards of *Journal of Cluster Computing*, *Journal of Parallel and Distributed Computing Practices*, and *Journal of Information Sciences and Engineering*. He has also served as guest editor for a number of journals including *IEEE Computer*, *IEEE*, and *Communications of ACM*.