TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/21 pp29-40
Volume 14, Number 1, February 2009

Performance Comparison of IP-Networked Storage*

JU Dapeng (鞠大鹏)[1,2,**], LIU Chuanyi (刘川意)[3], WANG Dongsheng (汪东升)[1,2,3],
LIU Hong (刘 宏)[2], TANG Zhizhong (汤志忠)[3]

1. Research Institute of Information Technology, Tsinghua University, Beijing 100084, China;
2. Tsinghua-Nuctech Data Security Institute, Tsinghua University, Beijing 100084, China;
3. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract: Dramatically increasing amounts of digital data are placing huge requirements on storage systems. IP-networked storage systems, such as the network file system (NFS)-based network-attached storage (NAS) systems and the iSCSI-storage area network (SAN) systems, have become increasingly common in today's local area network (LAN) environments. The emergence of new storage techniques, such as object-based storage (OBS) and content aware storage (CAS), significantly improves the functionality of storage devices to meet further needs for storage sub-systems. However, these may impact system performance. This paper compares the performance of NFS, iSCSI storage, object-based storage devices (OSDs), and CAS-based storage systems in an environment with no data sharing across host machines. A gigabit ethernet network is used as the storage network. Test results demonstrate that the performances of these systems are comparable with CAS being much better than the others for write operations. The performance bottlenecks in these systems are analyzed to provide insight into how future storage systems may be improved and possible optimization methods. The analysis shows how the I/O interfaces in these systems affect the application performance and that network-based storage systems require optimized I/O latency and reduced network and buffer processing in the servers.

Key words: storage technique; CAS area network; performance comparison; IP-networked storage

Introduction

The era of data and information significantly affects every aspect of our lives. Large amounts of new data are generated by humans every year. One study[1] estimated that in 2007 about 255 exabytes (255 billion gigabytes) of information would be created and replicated with the amount of information created surpassing available storage capacity for the first time. The study predicted that the information added annually would increase more than six fold from 161 exabytes to 988 exabytes between 2006 and 2010, growing by 57% a year. The demands on storage devices and storage systems are growing rapidly and persistently.

With the advent of high-speed local area network (LAN) technologies such as the gigabit ethernet, IP-networked storage has become increasingly common in client-server environments. Ten gigabit ethernet interconnects will soon become commodity, accelerating this trend further.

An IP-networked storage permits access to remote data. The traditional and widely used method for networking storage over IP is to simply employ a network file system (NFS)[2]. In this approach, the server makes a subset of its local namespace available to the clients

Received: 2008-03-26; revised: 2008-09-22
* Supported by the National Natural Science Foundation of China (No. 60273006) and the Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList)
** To whom correspondence should be addressed.
E-mail: judapeng@tsinghua.edu.cn; Tel: 86-10-62773691

and the clients access metadata and files on the server using a remote procedure call (RPC)-based protocol.

An alternate approach for accessing remote data is to use an IP-based storage area networking (SAN) protocol such as iSCSI[3]. iSCSI is a block-level protocol that encapsulates standard SCSI commands into TCP/IP packets. iSCSI connects a SCSI initiator port on a host to a SCSI target port on a storage subsystem. The initiator and the target look like a client and a server in an NFS system. In an iSCSI storage system, a remote disk exports a portion of its storage space to a client as an SCSI device. The client handles the remote disk no differently than its local disk. It runs a local file system that reads and writes data blocks to the remote disk. Remote blocks are accessed by encapsulating SCSI commands into TCP/IP packets. Thus, iSCSI 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.

The two techniques for accessing remote data employ fundamentally different abstractions. A network file system accesses remote data at the granularity of files, while SAN protocols access remote data at the granularity of disk blocks. In the network file approach, the file system resides at the server, whereas in the SAN approach it resides at the client. Consequently, the network I/O consists of file operations (file and metadata reads and writes) for network file systems and block operations (block reads and writes) for SAN protocols. NFS and iSCSI provide fundamentally different data sharing semantics. NFS is inherently suitable for data sharing since it enables files to be shared among multiple client machines while iSCSI supports a single client for each volume on the block server. Consequently, iSCSI permits applications running on a single client machine to share remote data, but it is not directly suitable for sharing data across machines.

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 encrypted. This adds more overhead and further exacerbates the iSCSI performance.

The block interface to iSCSI storage systems is very narrow and cannot convey additional semantics to the storage for self management. Object-based storage devices (OSDs) store and manage data containers called objects which can be viewed as a convergence of file and block technology[4]. Files have associated attributes which convey some information about the data that is stored within. Blocks, on the other hand, enable fast, scalable, and direct access to shared data. An OSD is capable of managing its capacity and presenting file-like storage objects to its hosts. Objects can be created and destroyed and can grow and shrink in size during their lifetimes. A single command can be used to read or write any consecutive stream of bytes constituting a storage object. In addition to mapping data to storage objects, the OSD storage management component maintains other information about the storage objects as attributes, e.g., size, usage quotas, and associated user name. In an OSD-based iSCSI-SAN, the server components include the iSCSI server and the object storage server. The object storage server module manages the physical storage media and processes SCSI object commands.

Content aware storage[5] (CAS) is a special kind of OSD, in which object identifiers are not user-defined names but cryptographic hash values generated based on their content. Since the hash values for a specified hash function are inherently globally unique and it is computationally infeasible to find two distinct inputs that hash to the same value[6], the hashes for different objects are unique. CAS uses this method to provide integrity checking of data, space efficiency, write-once characters, and overwrite protection. CAS is especially useful for storing fixed content data.

Several studies have compared the performance of IP-networked storage systems. Some of these studies have focused on the iSCSI performance based on the data path overhead and latency[7,8]. One report[9] compared a commercial iSCSI target implementation and an NFS system using metadata intensive benchmarks. The overhead introduced by the iSCSI systems, compared to systems with directly-attached storage, is evaluated in the context of commodity iSCSI systems[10]. The tests used commodity personal computers with several disks as storage nodes and a gigabit ethernet network as the storage network. The Linux kernel was instrumented to provide detailed information about the I/O activity and the overhead for the various kernel I/O layers. The NFS and iSCSI

performance for environments with no data sharing across machines was compared by Radkov et al.[11] in terms of protocol interactions, network latency, and sensitivity to different application workloads using a Linux-based storage system testbed. The results show that iSCSI and NFS are comparable for data-intensive workloads, while the iSCSI outperforms the NFS by a factor of two or more for metadata intensive workloads. They identified aggressive metadata caching and aggregation of metadata updates in iSCSI to be the primary reasons for this performance difference.

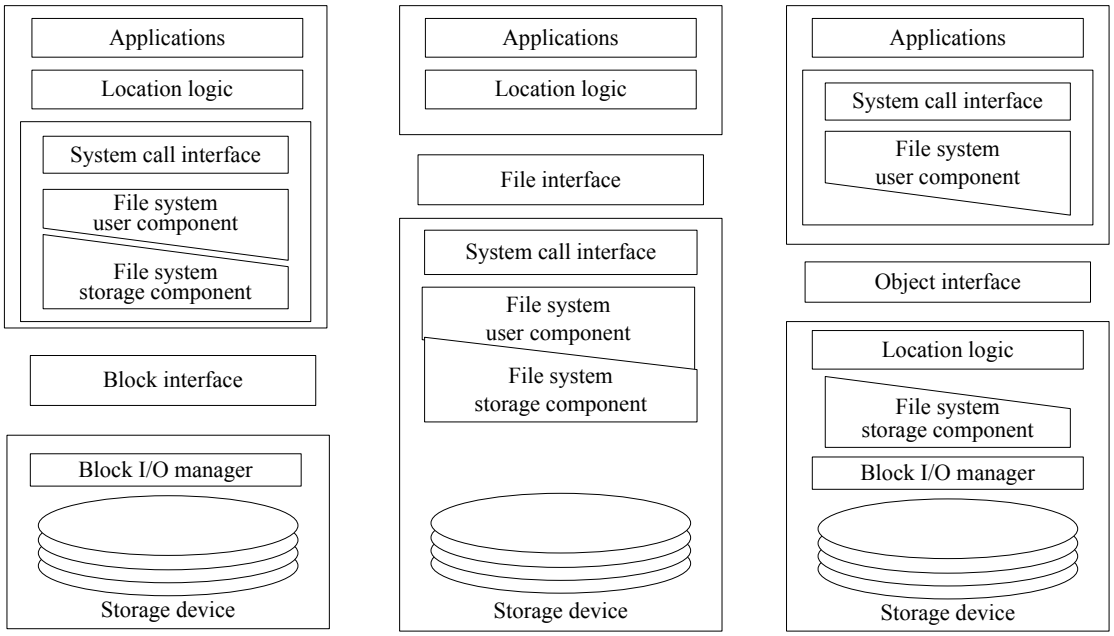
Du et al.[12] compared a reference implementation of the OSD T10 with iSCSI and NFS. Their tests showed that in general the raw read and write performance of an OSD operation is slower than that of the corresponding iSCSI operation with larger transfer sizes yielding better throughput for both the iSCSI and OSD systems. The throughput saturated before reaching the network bandwidth limit of 1 Gbps. The OSD file system throughput was significantly lower than that of the NFS and iSCSI systems. Liu et al.[13] compared the performance of OSD systems with iSCSI and NFS systems and found that the write performance of the object-based storage systems is much better, with the CPU usage on the client side greatly reduced.

CAS, which is very useful for archival storage, digital information preservation, and data de-duplication, has received much attention recently. There are several studies focusing on building CAS prototypes for different research goals with performance evaluations. Most CAS prototypes were designed before iSCSI was available. Thus, the CAS systems could only be compared with the NFS system as the only available IP-networked storage at the time. Most CAS performance studies have been performed on wide-area networks rather than local-area network. Pond[14], an OceanStore[15] prototype, outperformed NFS in wide-area networks by up to a factor of 4.6 on read-intensive phases, but underperformed NFS by as much as a factor of 7.3 on write-intensive phases. The Pond write performance was limited by the speed of the erasure coding and threshold signature generation. CFS[16] is a wide-area cooperative p2p read only storage system which delivers data to clients as fast as file transfer protocol (FTP). Ivy[17] is a multi-user read/write p2p network file system built atop the CFS storage layer which uses file system semantics much like those of an NFS v3 file server. On a wide-area network Ivy is two to three times slower than NFS. The main performance bottlenecks are the network latency and the cost of generating digital signatures for the data stored in the distributed hash. Venti[18] is several times slower than directly accessing disks over a 100-Mbps ethernet. CASPER[19], a distributed file system, was evaluated for benchmarks running at client-server bandwidths of 10 Mbps, 1 Mbps, and 100 Kbps.

The current work differs from previous studies in that this study compares the CAS performance based on iSCSI with that of NFS, iSCSI storage, and iSCSI-based OSD to present CAS performance characteristics for modern storage systems.

These storage architectures are compared in Fig. 1. In a modern IP-SAN system, both OSD and CAS can be networked using the iSCSI protocol.

Although OSD and CAS appear appealing, their impact on the system is not well defined. The adoption of an object interface in OSD and CAS introduces further overhead. Moreover, additional layers in the I/O path, which include many computations such as the computation cost of chunking algorithms or cryptographic hash algorithms, are performance bottlenecks. Thus, the performance characteristics of iSCSI-based OSD and CAS storage area networks (OSDN and CASN) must be understood to evaluate their impact on various applications. There should be an extra effort directed toward measuring and analyzing the performance parameters, and comparing the performance of OSD and CAS with traditional storage models. While there is some work on performance comparisons of iSCSI storage and NFS, there are few performance analyses comparing OSD with NFS and iSCSI systems. To our knowledge, there are no papers comparing the performance of CAS with iSCSI storage systems. This paper compares the performance of NFS, iSCSI storage, OSDN, and CASN systems as specific examples of IP-networked storage systems. The objective is to study the real performance of these storage systems, the factors affecting their performance, and where performance bottlenecks occur. These systems were studied on a testbed to obtain real performance measurements for these four typical IP-networked storage systems in terms of the raw I/O performance. The testbed uses a single client machine accessing one remote data storage (i.e., no data sharing across



(a) Block interfaced model (SAN or DAS) (b) File interfaced model (NAS) (c) Object interfaced model (OSD)

Fig. 1 SAN, DAS, NAS, and OSD storage model architectures

machines). Linux is currently the only open-source platform able to implement NFS, iSCSI, and OSD, so a Linux-based storage system platform was used to compare their performance. Since there is no open-source CAS system yet, a prototype of the CAS area networked storage (CASN) on Linux was used with the open-source Intel iSCSI[20] as the storage transport protocol. The results show that the performance of IP-networked storage systems such as NFS, iSCSI storage, OSDN, and CASN systems are comparable, with CASN much better than the others in terms of write operation performance, though some are more suitable for specific applications and scenarios.

1 Implementation of CASN

There are open source codes available for implementing all the IP-networked storage systems except CASN. The NFS was implemented based on NFS v3[21,22] with Intel iSCSI[20,23] to implement both iSCSI and OSD area networks (OSDN). The CASN was based on the Intel iSCSI and OSD T10[24] .

As shown in Fig. 2, the CASN system consists of (1) clients; (2) application servers, such as an e-mail server, a multimedia server running specific applications; (3) CAS servers; (4) metadata server in charge of metadata interactions with clients and security mechanisms such as authentication of clients and data access control; (5) a network connecting client hosts and application servers, which is usually the Internet; and (6) a storage area network (SAN) used to connect application servers and the CAS devices through a 1-Mbps IP-SAN based on Intel iSCSI v2.0[20,23].

1.1 Client, application server, and CAS appliances

Clients deliver application requests to a specified application server and handle responses returned from the application server.

Application servers are located in the middle layer of the storage network architecture, communicating with both the clients and CAS servers. A specific application server, such as a web server, an e-mail server, a file server, or a multimedia server, handles requests from various clients through packages usually deployed on the Internet. To avoid frequent metadata communication with the metadata server (MDS), application servers also locally cache triples (file path, offset, and object ID).

Every application server is bound with a CAS appliance which divides files to be archived into objects, uses the OSD T10 standard interface to transfer objects to the CAS servers and stores file-object mappings as well as metadata describing the files and objects on the

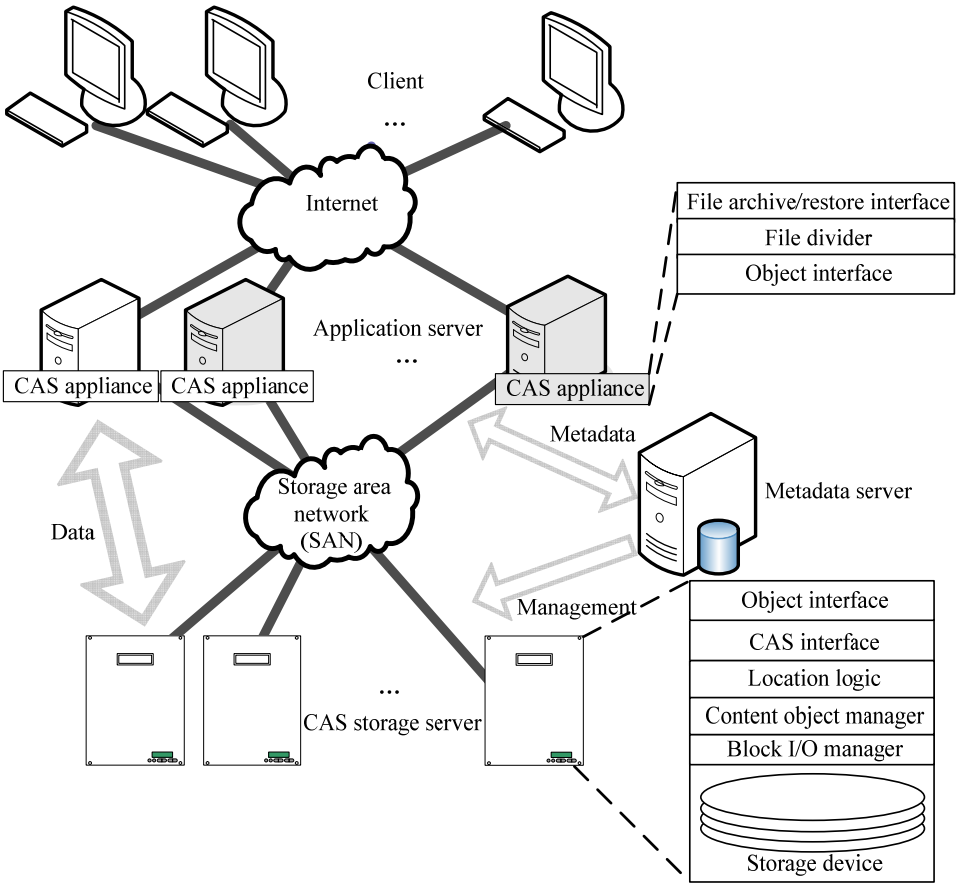


Fig. 2 CASN architecture

MDS for future restore and data retrieval operations. The CAS appliances transfer data from or to the CAS servers across the OSD T10 standard object interface.

The structure of the CAS appliance is shown in the upper right part of Fig. 2. Files are archived and retrieved from the exposed file archive/restore interface of the CAS appliance. If a file operation is archiving, the file divider module is employed to divide the file into objects and the hash value of each object's content is calculated as the object ID. The metadata information is stored into the MDS for future access and retrieval.

The key function of the CAS appliance is the file division, i.e., how to divide files into objects so as to most reduce the inter-file duplication. To reduce the complexity of this typical component, the CASN first used a fixed size object division method.

1.2 MDS and security protocol

The MDS functions can be classified into two types:

(1) Managing the metadata for the files and objects. When a new file is created, the file attributes (such as the file path, inode information, and application attributes) are extracted and stored into an entry in the File_Attibutes_Table. Then the file is divided into objects by the file divider in the CAS appliance. The file-object mapping is then stored into the File_Object_Table. The object metadata is stored in the Object_Metadata_Table. When an application server wants to restore or retrieve a file, it first passes the file path and the byte offset within the file to the MDS. The MDS translates the request into the object index and searches the File_Object_Table to find which CAS server has the objects. When the location information is received, the application server directly communicates with the CAS server to transfer the data. Further requests for the same object will not involve the MDS. With this out-of-band mechanism, the MDS is only in the control path but not the data path, so it cannot become a system bottleneck. Moreover, since the CASN, MDS, and application servers are all in the same high speed local area network, the latency of the interactions between servers can be ignored.

(2) Security authentication and access control. The MDS handles the client authentication, device operation authentication, and object operation access control.

The T10 SCSI OSD Standard[24] includes an integrated security protocol that protects the storage and specifies a credential-based access control policy for the core components. The resulting protocol is based on a secure capability-based model, enabling fine-grained access control that protects both the entire storage device and individual objects from unauthorized access. A security protocol conforming to this standard was implemented to check not only a host's identity but also authenticate qualified storage operations granted to a host. Moreover, it also provides data integrity checking by using a cryptographic hash function to prevent a malicious server from fulfilling a read request with fraudulent data.

In the CASN, a client wishing to perform an I/O request should follow the steps described in Fig. 3. An application server can request an I/O operation to a CAS server only if its ID has been successfully authenticated. When an application server requests an operation, it contacts the MDS to obtain the capability including the operation permission and a capability key to generate an integrity check value. The MDS creates a capability key with a key shared between the MDS and the CAS server, and a credential which contains the capability and capability key. After receiving the credential returned from the MDS, the application server copies the capability included in the credential to the capability portion of the command description block (CDB) and generates an integrity check value of the CDB with the received capability key. The CDB with the digested hash value called the request integrity check value is sent to the CAS server. When the CAS server receives the CDB, it checks the validity of the CDB with the request integrity check value.

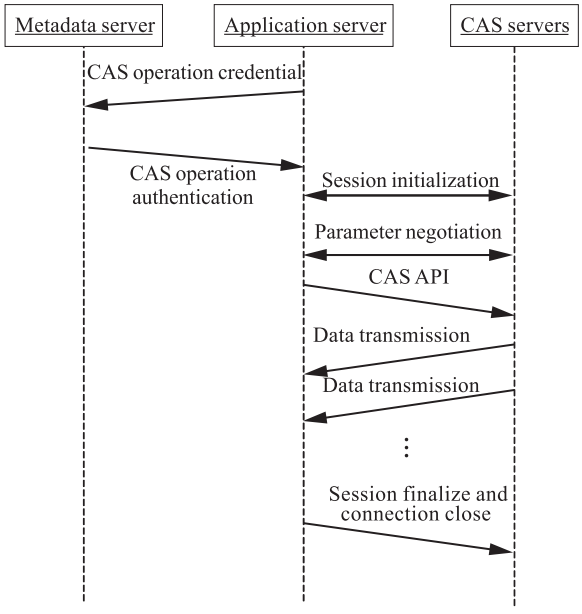


Fig. 3 CASN security protocol

Therefore, both the entire storage server and individual objects can be protected from unauthorized access by this security mechanism. To access objects, a user must acquire cryptographically secure credentials from the MDS. Each credential contains a capability that identifies a specific object, a list of operations that may be performed on that object, and a capability key that is used to securely communicate with the CAS server. Before granting access to any object, each CAS server checks whether the requestor has the appropriate credentials. Here, the credential is a data structure containing a capability prepared by the MDS and protected by an integrity check value.

1.3 CAS server and interface

The inner architecture of a CAS server is shown in the lower part of Fig. 2. This autonomic system consists of commodity components such as processors, I/O channels, disk controllers, disks or disk arrays, and usually a large amount of memory, as well as network adapters to connect with other CAS servers, MDS, and application servers to construct the SAN. The CAS API is wrapped with the OSD standard command library to work seamlessly with CAS servers from other venders while making use of the existing IP-SAN transport protocol. The location logic layer below the interface locates the object metadata inside the CAS devices. The next layer is the content object manager which is located above the block I/O manager and behaves as an abstract CAS device driver, supporting object management and immutability. Its functions include defining and managing metadata, mapping between the metadata and the data, managing free space, and allocating (de-allocating) object space. The bottom layer is the block I/O manager, which separates a logical device from the underlying physical devices, regardless of the actual type (SCSI device, SATA device, or disk array). In the CASN, this layer is supported by the

logical volume management (LVM) mechanism of the Linux kernel.

A CAS server uses a globally unique object ID (OID) to represent each object which includes the global name space design and management. The CASN uses cryptographic hash functions, such as MD5 or SHA1, to generate the object content hash value represented as the object's ID. Fortunately, the hash values for a specified hash function are inherently globally unique and it is computationally infeasible to find two distinct inputs that hash to the same value[6].

The units of storage accessed in a CAS server are classified as objects. Each object is associated with a metadata structure, similar to the relationship between a file and an inode in file systems. However, unlike file systems, the metadata is part of the object so it contains the necessary and sufficient information to retrieve the real data. In the CASN design, the object metadata is organized as an XML file and the object attributes can be extracted for use as an index for building a search engine to dramatically speed up content search efficiency at the storage level. A metadata memory cache can speed up I/O operations.

When a request comes to a specific CAS server, the CAS server first queries the hash-metadata map table to find the location of the metadata associated with the object, then gets the related metadata by which to retrieve the object content, and finally reads or writes the data from or to the CAS server. This flat namespace in the CAS servers gives significant performance improvements over traditional file systems which need several metadata access steps to traverse the file path tree.

A CAS server needs to keep a hash-metadata map record for each object since each hash digest corresponds to an object. With increasing number of stored objects, traversing the whole map table to find the specified hash-metadata record will cost more and more time. In the CASN, the hash-metadata map table is stored in a red-black tree[25] data structure, with the metadata in the hash-metadata map table stored in a fixed position on the disks, to function like a super-block in a file system glossary. When a CAS server boots up, the system copies the metadata in the hash-metadata map table from the disks into the main memory to reduce the object location time. To increase reliability, the hash-metadata map table cache is periodically written back to the disks when changes occur (new objects are inserted or old objects are deleted).

Sectors associated with an object should be kept contiguous whenever possible. All blocks needed for a particular object are allocated in advance since object sizes do not change dynamically.

A CAS server maintains a list of contiguous free blocks on the disks. Each entry in the list contains the address of the first sector and the number of free contiguous sectors. The space track method is similar to the conventional track method.

The interface of a CAS server is compliant with the OSD T10 standard so as to work seamlessly with CAS servers from various venders. It also directly makes use of the IP-SAN data transfer protocols (such as iSCSI and iFCP) without modification. The interface definition and operation descriptions are listed in Table 1.

The operation API is embedded into the SCSI OSD standard command library at the CAS server interface. The OSD commands are described in detail in the OSD specification[4]. The commands and data are then encapsulated and transferred to the IP storage area network. The CASN transport protocol uses iSCSI[20,23].

Table 1 CAS device interface

CAS API	Function prototype	Description
CREATE	Hash Retval = CREATE(out Hash *h*, in Data data)	Cause the CAS device to allocate and initialize one or more user objects
REMOVE	Status Retval = REMOVE(in Hash *h*)	Delete a user object
READ	Status Retval = READ(in Hash *h*, out Data data)	Request that the CAS device returns data to the application client from the specified user object
QUERY	Status Retval = QUERY(in Hash *h*)	Indicate whether the specified object exists
GET_ATTR	Status Retval = GET_ATTR(in Hash *h*, out Attributes attr)	Return the specified attributes for the specified object
FORMAT	Status Retval = FORMAT(in CAS_ID cas)	Format CAS device

2 Experimental Setup and Method

The performances of four IP-network storage systems, NFS, iSCSI, OSDN, and CASN, were measured on a storage testbed. The storage testbed consisted of a client, an application server, and a storage server connected over an isolated gigabit ethernet LAN.

A Sun Fire V40z server used as the storage server was equipped with 4 AMD dual-core processors at 2.4 GHz (1 MB L2 cache per processor), 16 GB RAM, and 6 Ultra-320 SCSI disks (146 GB per disk). A SUN workstation used as the application server was equipped with 2 AMD dual-core processors at 1 GHz (1 MB L2 cache per processor), 8 GB RAM, and a 250-GB Ultra-320 SCSI disk. The network connection was a 1-Gbps D-Link DGE550T switch. The open source Intel iSCSI was used to implement the iSCSI storage with a 4-KB blocksize. The client shared 10 GB storage with the server. This client server machine combination was used for all four storage configurations with the same disk partition at the server to ensure the disk performance remained constant for all configurations.

The Linux operating system was kernel version 2.6.9. The NFS version was v.3. Both synchronous and asynchronous modes were measured with default NFS parameter settings and with UDP as the transport protocol. The iSCSI system used the Intel iSCSI v2.0 with the default parameter settings. The CASN treated the whole file as an object, with the ObjectIDs generated using the MD5[6] hash algorithm.

The NFS, iSCSI, OSDN, and CASN systems are typical low-level architecture implementations of IP-networked storage systems corresponding to storage domains in the SNIA shared storage model[26]. Every application executes on top of the storage domain layer. All the storage methods are transparent to the applications, regardless of which system is used in the substrate. The performance characteristics of the NFS, iSCSI, OSDN, and CASN systems were evaluated using applications unrelated to the differences in their implementations.

The main goal of the evaluation was to compare the primary performance parameters and to identify system bottlenecks. The throughput and CPU utilization were measured during the transmission of network files to evaluate the system performance characteristics.

IOmeter[27] was used as the testing benchmark since it is a configurable workload generator that has been used extensively for basic evaluations of I/O subsystems. However, IOmeter can only test storage systems with block interfaces; therefore, parts of the IOmeter code were modified. The implemented IOmeter can extract both the test parameters and the test results. The test program acquires the state information from the Linux kernel files to calculate the system performance parameters. The application server initiated read/write operations with files having sizes from 1 KB to 1 GB to the storage server, recording both the operation start and stop times. The read/write throughput was obtained from the data set size divided by the interval between the start and the stop times.

3 Experimental Results and Analysis

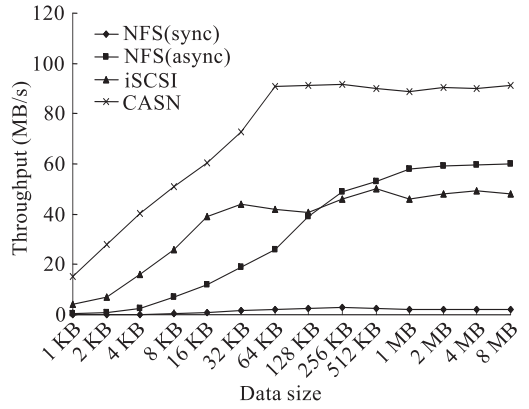
3.1 Performance comparison of NFS, iSCSI storage, and CASN

Figure 4 shows the throughput of read/write operations for the NFS, iSCSI, and CASN systems.

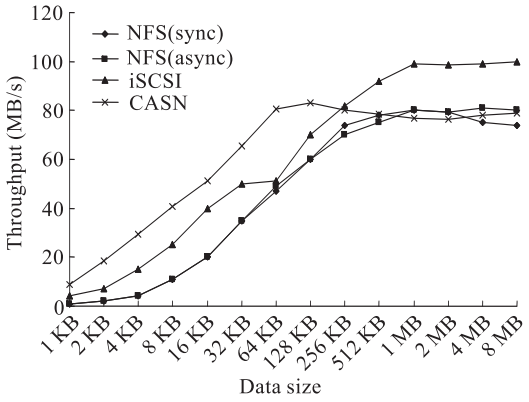
The write throughput in the first part of Fig. 4 demonstrates that in the NFS system the throughput in the asynchronous write operation mode approaches 60 MB/s for data sizes larger than 1 MB, but in the synchronous write mode the average throughput is only 1-2 MB/s because in the synchronous model the server doesn't complete an NFS request (reply to the client) until the file system on the server has flushed all the data/metadata onto the disk, while the asynchronous mode permits the server to reply to client requests as soon as it has processed the request and handed it off to the local file system, without waiting for the data to be written to disk.

The iSCSI write throughput increases continuously and approaches about 50 MB/s for data sizes greater than 512 KB. For smaller data unit sizes (less than 64 KB), the iSCSI write throughput doubles the NFS write throughput.

In the CASN the write throughput increases more quickly, reaching the upper limit for data sizes larger than 64 KB. The CASN has nearly twice the write throughput of the other two systems for any data size because write operations are performed for immutable objects which are appended sequentially to the storage devices, so the CASN can fully make use of the device



(a) Write operation



(b) Read operation

Fig. 4 Throughput for the NFS, iSCSI, and CASN systems on a 1-Gbps network

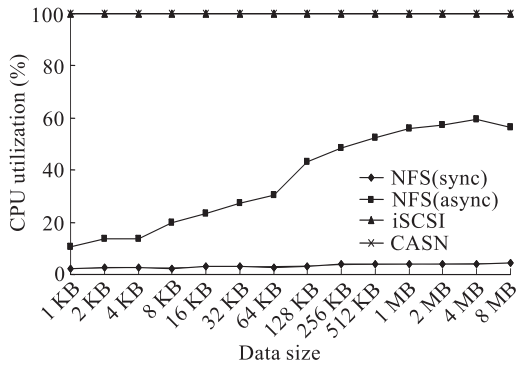
sequential access bandwidth, which is almost one order of magnitude faster than the random access bandwidth. In addition, both the storage server and the application server use large amounts of memories, larger than the transmitted data sizes. The asynchronous I/O mechanism makes full use of the system memories, so applications and I/O operations may run in parallel, improving the overall system performance.

The read throughput in Fig. 4b increases continuously with increasing data size. Both the NFS synchronous and asynchronous read operation throughputs are faster than the write operations, about 80 MB/s. The system bottleneck is the limited network bandwidth in the NFS asynchronous read operation.

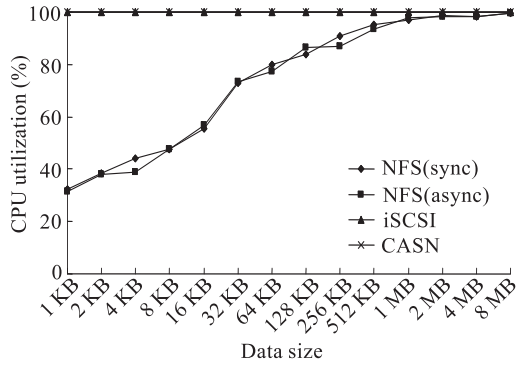
The throughput for the read operations on the iSCSI storage can achieve 100 MB/s, doubles the throughput for write operations and almost equals the limit of the network bandwidth. The iSCSI read operations have similar trends to the NFS results. The big difference between these two schemes lies in the networking file access protocol. For each file access, NFS as a stateless protocol running on the file level requires the exchange of several commands before data communication can occur. In the iSCSI the client side retains the file states and its directories, which saves several round trips. Also, iSCSI uses the TCP protocol, while NFS uses remote procedure call (RPC) over the TCP protocol, so for large data transfers, the iSCSI transfer is a little more efficient.

The CASN read performance is better than that of the other systems for smaller data sizes, tending to a maximum for data sizes larger than 64 KB with about the same rate as the NFS synchronous/asynchronous read performance, but about 20% less than that of iSCSI. The CASN read throughput is less than the write throughput as described earlier.

Figure 5 shows the CPU utilization for the NFS, iSCSI, and CASN systems.



(a) Write operation



(b) Read operation

Fig. 5 CPU utilization with the NFS, iSCSI, and CASN systems on a 1-Gbps network

The CPU utilization results show that the NFS system does not fully utilize the CPU, while the CASN and iSCSI systems fully use the CPU. In the NFS asynchronous mode, the CPU utilization does not reach

100%, so the performance bottleneck is not the computation capacity. Thus, the high throughput is based on its mature cache mechanism.

The iSCSI-based systems were implemented based on a block-layer, so the CPU spends much time encapsulating and de-encapsulating iSCSI packets for the network transmissions. The CPU utilization is more than that for the NFS system. In this 1-Gbps network, the system performance is limited by the bandwidth for processing iSCSI packets rather than the network bandwidth, so the CPU utilization is 100%.

The CPU utilization with CASN is also 100% since the system is also typically CPU-intensive. The hash values for the objects constructing files are needed to be calculated before the data is transmitted. Thus, extra effort is needed to improve the speed of the hash computation.

Overall, the test results show that the performance of the three IP-networked storage systems (NFS, iSCSI, and CASN) are comparable with the CASN read performance better than that of the other two systems when the data size is small, nearly double that of the other two systems for write operations.

3.2 CASN and Intel OSDN performance comparison

Figures 6 and 7 show the OSDN (based on the Intel OSD implementation) and CASN throughput for sequential read and write operations. The CASN throughput is better than that of OSDN when the object sizes are smaller than 512 KB with the CASN throughput being about 50% higher than the Intel OSDN throughput for an object size of 64 KB, because the CAS device uses its own API which is a virtual device on the disks, without directly interacting with the users. The objects are mapped onto metadata by the hash values, with the data then read from the disks. In addition, the entire I/O path in CASN is shorter than that of traditional devices. While the OSDN is compatible with the virtual file system (VFS) interface, read and write operations must go through the file system, increasing the overhead. This overhead becomes relatively larger as the object size decreases. The OSDN performance is about 15% better than that of the CASN on average for objects larger than 4 MB because of the file system cache.

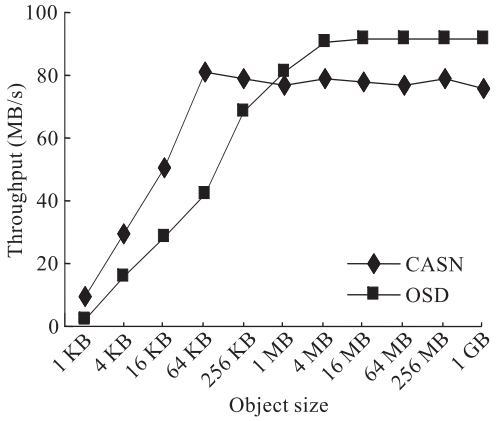


Fig. 6 Read throughput of OSDN and CASN systems

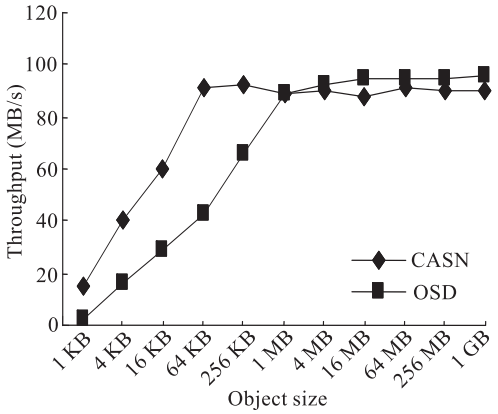


Fig. 7 Write throughput of OSDN and CASN systems

Since every chunk is a non-overwritten object in CASN which is accessed as a unit, the CASN write performance is much better than for the other systems.

4 Conclusions

Various forms of networked storage systems such as NFS, iSCSI storage, OSDN, and CASN have been proposed as candidates for future storage systems. However, there are numerous questions associated with their impact on system performance. This study used public domain implementations of NFS, iSCSI, and OSD and our implementation of CASN based on iSCSI to compare the performance of NFS, iSCSI storage, OSDN, and CASN IP-networked storage systems in an environment where the storage is not shared across client machines.

This paper describes the CASN implementation based on the Intel iSCSI and the OSD T10 standard with comparisons to several other typical IP-networked storage systems including regular storage networks such as NFS, iSCSI, and iSCSI-SAN based on new

storage devices such as OSD and CAS. The analysis identifies bottlenecks in these systems and suggests optimization directions.

The results show that the performance characteristics of the NFS and three iSCSI-based storage systems are comparable, with CASN having twice the write throughput of the NFS and iSCSI systems because write operations of immutable objects in CASN make full use of the device sequential access bandwidth, which is faster than the random access bandwidth. CASN is better than OSDN for sequential read and write operations when the data size is smaller (less than 1 GB) because of the shorter I/O path in CASN.

Building IP-networked storage systems requires optimizing the I/O latency, reducing the network and buffer cache related processing in the servers, and increasing the network bandwidth to accommodate different types of traffic. Thus, there will still be significant improvements in future IP-networked storage systems.

This performance comparison of IP-networked storage systems is a preliminary study, limited to the basic throughput and CPU utilization, measured with IOmeter. Future work will use more types of applications to compare the application performance of IP-networked storage systems in various configurations.

References

[1] Gantz J F, Reinsel D, Chute C, et al. The expanding digital universe: A forecast of worldwide information growth through 2010. An Internet Data Center (IDC) White Paper, sponsored by EMC, 2007.

[2] Sandberg R, Goldberg D, Kleiman S, et al. Design and implementation of the Sun network file system. In: Proceedings of the Summer 1985 USENIX Conference. Portland, USA, 1985: 119-130.

[3] IETF (Internet Engineering Task Force). iSCSI, version 08. IP storage (IPS), Internet draft, Document: draft-ietf-ips-iscsi-08.txt, 2001.

[4] Mesnier M, Ganger G R, Riedel E. Object-based storage. *IEEE Communications Magazine,* **41**(8): 84-90.

[5] Cakeljic Z. Content aware storage. Storage Networking Industry Association (SNIA), 2005.

[6] Rompay B V, Preneel B, Vandewalle J. On the security of dedicated hash functions. In: Proceedings of the 19th Symposium on Information Theory. Veldhoven, Netherlands, 1998: 103-110.

[7] Aiken S, Grunwald D, Pleszkun A R, et al. A performance analysis of the iSCSI protocol. In: Proceedings of the 20th IEEE Symposium on Mass Storage Systems. San Diego, USA, 2003: 123-134.

[8] Lu Y P, Du D H C. Performance study of iSCSI-based storage subsystems. *IEEE Communications Magazine*, 2003, **41**(8): 76-82.

[9] Performance comparison of iSCSI and NFS IP storage protocols. Technical report, TechnoMages, Inc. http://www.technomagesinc.com/papers/ip-paper.html, 2003.

[10] Xinidis D, Bilas A, Flouris M D. Performance evaluation of commodity iSCSI-based storage systems. In: Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05). Monterey, CA, USA, 2005: 261-269.

[11] Radkov P, Yin L, Goyal P, et al. A performance comparison of NFS and iSCSI for IP-networked storage. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST). San Francisco, USA, 2004: 101-114.

[12] Du D, He D S, Hong C J, et al. Experiences building an object-based storage system based on the OSD T-10 standard. DTC Research Report, 2006.

[13] Liu P C, Hong S K, Hsu Y. Security enhancement and performance evaluation of an object-based storage system. *Lecture Notes in Computer Science*, 2007, **4782**: 408-419.

[14] Rhea S, Eaton P, Geels D, et al. Pond: The OceanStore prototype. In: Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST 2003). San Francisco, USA, 2003: 257-270.

[15] Kubiatowicz J, Bindel D, Chen Y, et al. OceanStore: An architecture for global-scale persistent storage. In: Proceedings of ACM ASPLOS (Architectural support for programming languages and operating systems). Cambridge, Massachusetts, USA, 2000, **35**(11): 190-201.

[16] Dabek F, Kaashoek M F, Karger D, et al. Wide-area cooperative storage with CFS. In: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP). Banff, Canada, 2001: 202-215.

[17] Muthitacharoen A, Morris R, Gil T M, et al. Ivy: A read-write peer-to-peer file system. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI). Boston, USA, 2002: 31-44.

[18] Quinlan S, Dorward S. Venti: A new approach to archival storage. In: Proceedings of the 2002 Conference on File and Storage Technologies (FAST). Monterey, California, USA, 2002: 89-101.

[19] Tolia N, Kozuch M, Satyanarayanan M, et al. Opportunistic use of content addressable storage for distributed file systems. In: Proceedings of Usenix 2003 Annual Technical Conference. San Antonio, USA, 2003: 127-140.

[20] Intel's open storage toolkit. Reference implementations of Internet SCSI (iSCSI) and object-based storage devices (OSD). http://sourceforge.net/projects/intel-iscsi, March, 2001.

[21] Pawlowski B, Juszczak C, Staubach P, et al. NFS version 3 design and implementation. In: Proceedings of the Summer 1994 USENIX Conference. Boston, USA, 1994: 137-152.

[22] Callaghan B, Pawlowski B, Staubach P. NFS version 3 protocol specification. Sun Microsystems, Inc. 1995.

[23] Linux-iSCSI project. http://linux-iscsi.sourceforge.net, April, 2004.

[24] International Committee for Information Technology Standards (INCITS). SCSI object-based storage device commands-2 (OSD-2). Project T10/1731-D. http://www.t10.org/ftp/t10/drafts/osd2/osd2r00.pdf, 2004.

[25] National Institute of Standards and Technology (NIST). Red-black tree. http://www.nist.gov/dads/HTML/redblack.html. Accessed on Aug 25, 2008.

[26] Storage Networking Industry Association (SNIA) Technical Council. Shared storage model. 2003.

[27] Intel Server Architecture Lab. IOmeter: The I/O performance analysis tool for servers, February, 1998.

Seven Research Projects Sponsored by the 973 Program and the Major Research Plan

China's Ministry of Science and Technology recently announced approval of China's National Basic Research Program (the 973 Program) and the Major Research Plan projects in 2008. Five Tsinghua research projects will be sponsored by the 973 Program and two by the Major Research Plan. Seven professors were appointed as chief scientists for the projects. Tsinghua has undertaken 28 projects since the 973 Program's inception involving 28 chief scientists and eight Major Research Plan projects involving eight chief scientists. Tsinghua is one of the country's leading institutions for supervision and conduct of the 973 Program projects.

The five research projects approved by the 973 Program are: "Basic Research on Manufacturing Equipment in Super-Large-Scale Integration" led by Professor Luo Jianbin from the Department of Precision Instruments and Mechanology; "Basic Research on the Architecture and Protocols of New Generation Internet" led by Professor Wu Jianping from the Department of Computer Science and Technology; "Research on Theory and Technological Foundations of Integrated Control Systems for Complex Production Manufacturing Processes" led by Academician Chai Tianyou; "Some Fundamental Issues of Functional Ceramics and their Devices for Information Technologies" led by Professor Nan Cewen from the Department of Materials Science and Engineering; and "Astrophysical Research on Black Holes and other Compact Objects" led by Professor Zhang Shuangnan from the Department of Physics. The two projects sponsored by the Major Research Plan are: "Structure and Function of Important Disease-Causing Membrane Proteins" led by Professor Shi Yigong, Vice Director of Tsinghua's Institute of Biomedicine, and "Exploring and Exploiting Exotic Quantum Effects at the Single Atom/Molecule Level" led by Academician Xue Qikun from the Department of Physics.

(From http://news.tsinghua.edu.cn)