OSD: A Tutorial on Object Storage Devices

Thomas M. Ruwart
Advanced Concepts
Ciprico, Inc.
Plymouth, MN 55441
tmruwart@ciprico.com
Tel: +1-612-850-2918
Fax: +1-763-551-4002

Abstract

Ever since online digital storage devices were first introduced in the late 1950's and early 1960's, the various functions key to storing data on these devices have been slowly migrating into the devices themselves. Early disk drives would send analog signals from the read/write head to a physically separate box that would deserialize and frame data into bytes. This data would then be sent to other processors to perform redundancy checks and data transmission to the requesting computer system. As engineers were able to fit more functionality into smaller spaces at reasonable costs, these key functions were migrated into the disk drive itself to the point where we now have an entirely self-contained unit complete with all the electronics that used to fill a small room.

However, even with the integrated advanced electronics, processors, and buffer caches, these disk drives are still relatively "dumb" devices. They essentially perform only two functions: read data and write data. Furthermore, the disk drives do not know anything about the data that they are storing. Things such as content, structure, relationships, quality of service, …etc. are all pieces of information that are external to the disk drive itself. The basic premise of Object Storage Devices is that the disk drive or, more generically, the storage device, can be a far more useful device if it had more information about the data it manages and was able to act on it.

This paper is intended to provide the reader with an overview of OSD, its history, its current state, and possible futures. It begins by presenting a brief history of Object Storage Devices and then discusses why OSD is an important step in the evolution of storage technologies in general. The basic OSD architecture is compared with current Direct Attached Storage (DAS), Storage Area Network (SAN), and Network Attached Storage (NAS) architectures in terms of management, device and data sharing, performance, scalability, and device functionality. Finally, the current status of OSD and related roadmaps are presented as a frame of reference.

Brief History of OSD

The most active work on OSD has been done at the Parallel Data Lab at Carnegie Mellon University (www.pdl.cmu.edu) originally under the direction of Garth Gibson [1,4,5,6,8]. This work focused on developing the underlying concepts of OSD and two closely related areas called Network Attached Secure Disks (NASD) and Active Disks. Other work has been done at the University of California at Berkeley [Keeton], the Universities of California Santa Barbara and Maryland [3], as well as Hewlett Packard Labs [7,9],

Seagate Technology, and Intel Labs. Topics covered by these early pioneers can be broken down into two main categories: OSD architecture and applied OSD concepts. The basic OSD architecture defined to date specifies a set of object functions that can be implemented over any transport (TCP/IP, SCSI, VI, …etc.) but the initial transport will be SCSI for the sake of ubiquity.

Motivation behind OSD

As disk drives and other types of storage devices become denser and more numerous the block-level methods used to access and manage them are reaching the limits of their scalability. OSD is a protocol that defines higher-level methods of communicating the creation, writing, reading, and deleting of data objects as well as other related functions for getting and setting object attributes. OSD is a level higher than a block-level access method but one level below a file-level access method. OSD is not intended to replace either block-level or file-level access methods but rather to add a needed layer of abstraction that sits between them. It is a technology intended to help make existing and future data storage protocols more effective in several areas that include:

- Storage Management
- Security
- Device and Data Sharing
- Storage Performance
- Scalability
- Device Functionality

These areas are becoming more critical to the success of storage *users* as well as the storage *vendors* who are increasingly concerned over ways to differentiate their products. It is quite possible that the OSD architecture will provide both the users and vendors with a highly flexible base on which to build new storage systems that can accommodate each of these areas more effectively than trying to extend the current block-based or file-based protocols.

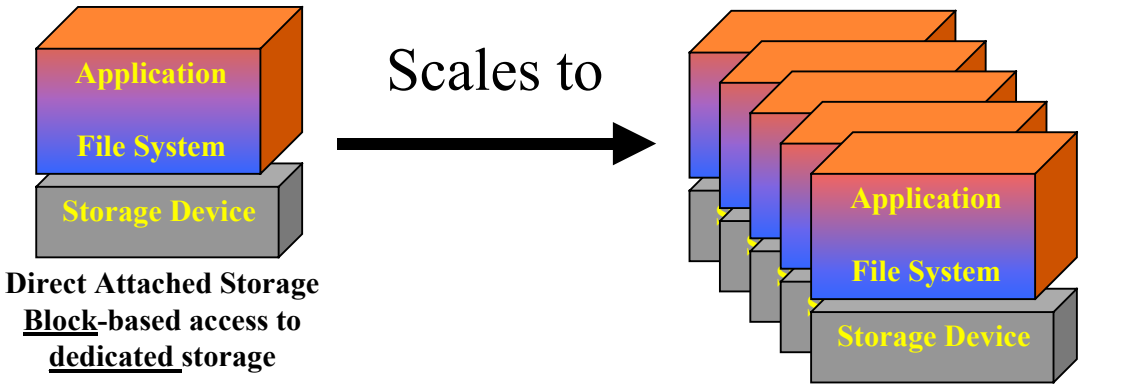
DAS/SAN/NAS Basic Architectures

There are three basic storage architectures commonly in use today. These are Direct Attach Storage (DAS), Storage Area Networks (SAN), and Network Attached Storage (NAS). Each of these is used to solve problems specific to a particular application or installation. Each has its strengths and weaknesses.

	DAS	**SAN**	**NAS**
Storage Management	High/low	High	Medium
Security	High	Medium	Low
Device and Data Sharing	Low	Medium	High
Storage Performance	High	High	Low
Scalability	Low	Medium	Medium
Device Functionality	Low	Low	Medium

Table 1. Capability assessment based on Technology

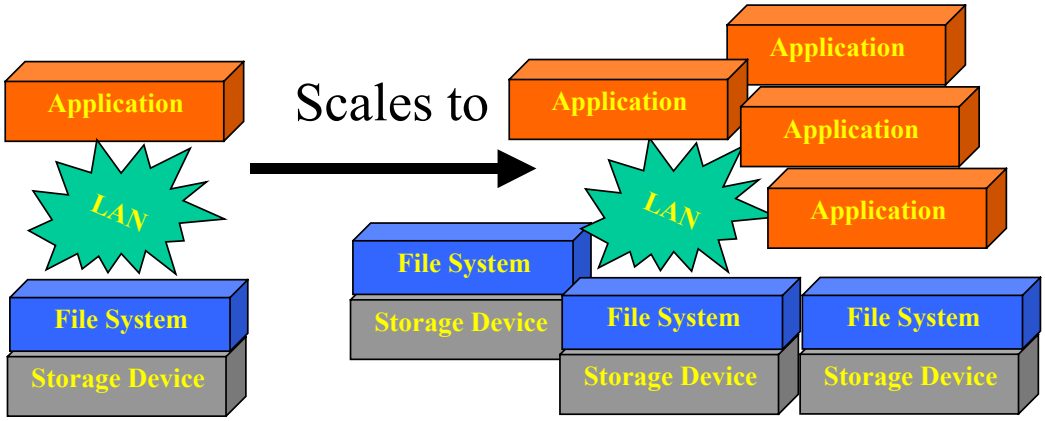
The DAS/SAN/NAS architectures and how they scale from a single subsystem to multiple systems are described in diagrams 1-3. Diagrams 4 and 5 show the basic architecture for OSD and the scaling thereof.



Direct Attached Storage
Block-based access to dedicated storage

Many individual systems and Applications with Block-based access to dedicated storage

Diagram 1. A single DAS scaling to multiple DAS systems. Each DAS system could conceivably add more storage devices but this is intended to show that when the limit of storage device connectivity is reached on a DAS system, the DAS system must be replicated.



Network Attached Storage
Shared File-Based access to dedicated storage

Many individual systems and applications with Shared File-Based access through the LAN to dedicated storage

Diagram 2. A single NAS scaling to multiple NAS and multiple application (clients). Note that the NAS boxes themselves can increase in capacity and that they scale in number independently from the application systems (clients).

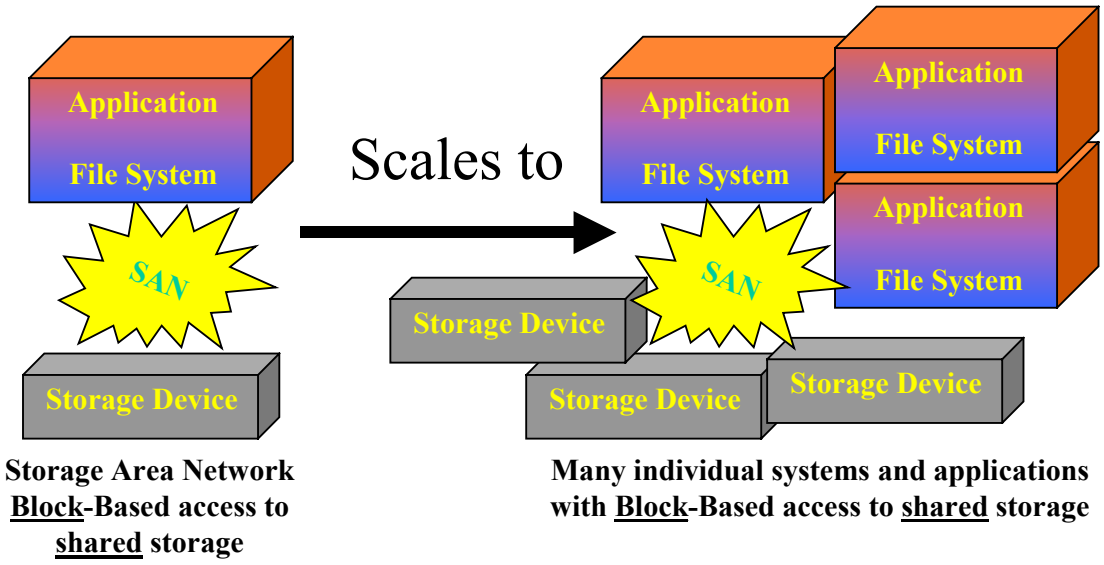


Diagram 3. A single SAN scaling to a larger SAN. Note that the storage devices and application (client) systems scale independently. There is implied device sharing and data sharing in this diagram.

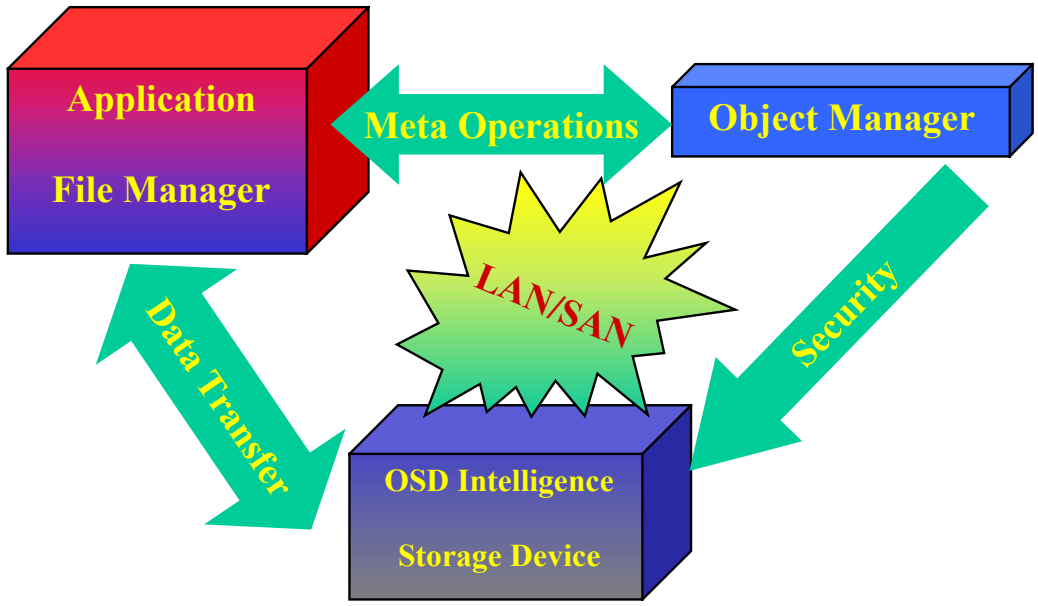


Diagram 4. A basic OSD architecture. Unlike DAS/SAN/NAS the Object Manager is a separate entity from the OSD and the application system (client). The transport for OSD can be either a LAN or a SAN.

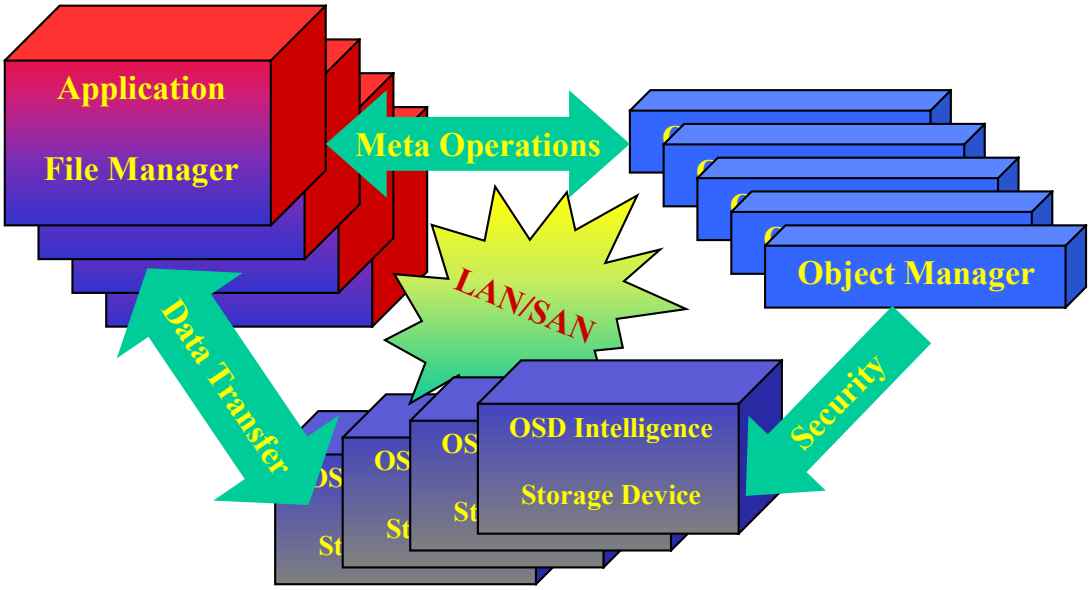


Diagram 5. Scaling a basic OSD architecture allows for increasing the number of OSD indefinitely as well as the application systems (clients). The Object manager can scale from a single system into a fully distributed cluster to accommodate the OSD and application system scaling. The transport for all these components can be either a LAN or SAN.

Basic OSD Architecture

One of the many motivations behind OSD was to take the strengths of each of the DAS/SAN/NAS architectures and incorporate them into a single framework. The basic OSD architecture and its scalability are shown in diagrams 4 and 5. There are many similarities between and OSD architecture and the DAS/SAN/NAS architectures. These include the use of Fibre Channel, Ethernet, TCP/IP, and SCSI protocols as transports and protocols. There are also several significant differences between OSD and the DAS/SAN/NAS architectures. These differences include the use of the following logical components:

- Object Manager
- OSD Intelligence
- File Manager

The Object Manager is used as a global resource to find the location of objects, mitigate secure access to these objects, and to assist in basic OSD management functions. This can be a single OSD that assumes these functions or it can be a completely separate, fully redundant cluster of systems. An Object Management Cluster would allow for scalability

in the number of objects that can be managed as well as the access performance of the Object Manager itself. It is important to note that the Object manager does not contain any user data or object meta-data nor does any of the data during a data transfer operation move through the Object Manager. The Object Manager is strictly used to facilitate location and secure access of objects.

The OSD Intelligence is the software (firmware) that runs on the storage device. It is responsible for interpreting the various OSD methods (commands): Create Object, Delete Object, Read Object, Write Object, and Get/Set Attributes. Furthermore, the OSD Intelligence can also provide the following capabilities:

- Object attribute interpretation
    - Object structure and relationship awareness
    - Object content awareness
    - Quality of Service (QoS)
    - Access Patterns
    - Security
- Sense of time
- Awareness and ability to communicate with other OSDs
- Device and data management

The OSD intelligence facilitates the communication of the OSD to the Object Manager for security purposes but mainly manages data processing and transfers between itself and the File Manager on the client requesting the data transfer. Since the OSD now has the intelligence to perform basic data management functions (such as space allocation, free space management, …etc.) those functions can be moved from the File SYSTEM manager to the OSD. The File SYSTEM manager now becomes simply a File Manager: an abstraction layer between the user application and the OSD. The File Manager provides backward compatible API for legacy codes to access files on OSD and, more importantly, it provides the security mechanisms required to ensure data privacy and integrity. More advanced capabilities of OSD can be exposed through the File Manager for user and system programs that wish to use them.

DAS/SAN/NAS/OSD Comparison

There are not actually any "new" data management functions in the OSD model. Rather it is simply a rearrangement of the existing functions in a general sense. From the user application point of view, the application creates, reads, writes, and deletes files as it always has. It does not know where the data is stored nor should it care. It does have certain data requirements (storage management, security, reliability, availability, performance, …etc.) that must be met and OSD provides a mechanism to specify and meet these requirements far more effectively than DAS/SAN/NAS. The following sections compare and contrast DAS/SAN/NAS to OSD in terms of the requirements listed in Table 1.

Storage Management[1]

Current estimates show that the cost of managing storage resources is about seven times the cost of the actual hardware over the operational life of the storage subsystems. This is independent of the type of storage (i.e. DAS/SAN/NAS). Given the tremendous growth in storage systems, storage resource management has been identified as the single most important problem to address in the coming decade. The DAS and SAN architectures rely on external storage resource management that is not always entirely effective and is in now way any kind of a standard. The NAS model has some management built into it but it too suffers from a lack of standards. The OSD management model relies on self-managed, policy driven storage devices that can be centrally managed and locally administered. What this means is that the high-level management functions can come from a central location and the execution of the management functions (i.e. backup, restore, mirror, …etc.) can be carried out locally by each of the OSDs and on an OSD peer-to-peer basis (i.e. a disk OSD backing itself up to a tape library OSD).

The DAS architecture is very simple to manage if there is only one system involved with some number of storage devices attached to it. All the management functions can be done from the one system that these devices are attached. However, if there is more than one system with storage devices attached, then it becomes increasingly difficult to manage all the storage devices because the management is distributed among all the systems that the storage devices are attached to. There is no central point of management in this case.

This problem is solved to some extent in a SAN configuration because ideally any one of the systems has access to all of the storage devices and management can be centralized on any one of these systems. A similar argument can be made for NAS devices since the network is a LAN and presumably any system on the LAN can see all of the NAS devices and hence can manage them all from a single system. Furthermore, the NAS devices have more "intelligence" built into them by their very nature (i.e. there is an OS with a file system, a communications stack, …etc.). This extra intelligence lends itself to the idea of self-managed storage making the overall task of managing storage resources somewhat easier. But is there a limit to the size of a system or the granularity of performance that can be managed in the NAS architecture?

The point here is that *centralized management* of storage resources (devices, space, performance, …etc.) with distributed administrative capabilities (i.e. the ability to carry out management functions locally) is essential to future storage architectures. In order to achieve this, the OSD architecture is designed to be self-managed thus more fully utilizing the OSD Intelligence built into each OSD. The devices will know how to manage each of several resources individually or through an aggregation of OSDs. These resources include (but not limited to):

- Space they have available at any given time
- Bandwidth has been requested
- Latency requirements of outstanding sessions

[1] In this section, the term "management" refers to the ability to install, configure, monitor, and administer the physical and logical storage devices as well as the space on these devices.

- The number of operations it is capable of performing in a given amount of time

Finally, OSD defines the concept of "object aggregation" whereby a hierarchy of OSDs can be made to appear as a single larger OSD. The resource management of this large aggregated OSD is done either through a single OSD at the top of the aggregation or can be done to each of the individual OSD devices in order to achieve maximum resource management flexibility.

Security

Security is second only to management in importance with respect to a data storage system. There are two basic threats that a secure system must guard against: External and Internal threats. External threats are attacks that come from outside the data storage system and outside the machines that are allowed access to the data on the storage subsystem. Internal threats are either benign or intentional. Benign threats are accidental access, modification, or corruption of data on a storage system. Intentional threats are intended to cause problems. In any case, multiple levels of security are necessary to authenticate, authorize access, ensure data integrity, and enforce data privacy.

Data security is becoming increasingly complex as the deployed systems and associated data storage systems grow in number and complexity. On the complexity scale, a DAS system is only as secure as the system that it is connected to. Assuming that the system is 100% secure, then access to the DAS device is very restricted.

By putting storage devices on a SAN however, there are more opportunities for access to the storage devices through other hosts that share the SAN. Generally, SANs are isolated and connected only to "trusted" host systems but there are still many other opportunities to connect to a SAN (i.e. through unused ports on a switch) and breach security. Since the SAN storage devices themselves do not have any notion of restricted access it is up to the host systems and SAN network infrastructure to enforce secure access to the storage devices.

NAS devices also have only as much security as the networks they are on and the firewalls and other security measures they implement. Because NAS devices tend to be on LANs the access restrictions may not be as stringent as those on SANs. However, since the NAS devices have some intelligence, they can implement more effect security measures than SAN devices.

The OSD concept incorporates a security model that includes four security levels:

- Authentication – you are who you say you are
- Authorization – you have permission to access to an object
- Data integrity – data is not modified or corrupted
- Data privacy – data is not to be seen by anyone else

The authentication is performed by the OSD transport layer. For example, for OSD over iSCSI over Ethernet, IPSEC would perform authentication. The remaining three levels are performed by the OSD itself. The authorization security mechanism is capability-based whereby the OSD manager gives capabilities to the clients and the clients present

these capabilities to the OSD. Finally, data integrity and data privacy are achieved through the use of cryptography. These are all features that make OSD security different from NAS security and certainly better than DAS and SAN security.

Device and Data Sharing

Concurrent device and data sharing is nonexistent on DAS systems unless the data is exported through an NFS or CIFS share to other systems. At that point the system essentially becomes a NAS device. Again, a SAN partially solves the problem by allowing any system connected to the SAN to access any device connected to the SAN. This is ideal for device sharing because the SAN provides a very high performance connection between any system and any device on the SAN. However, the problem of *data sharing* is left to the file systems to figure out. There are several ways to solve the problem of data sharing on a SAN, each with its own strengths and weaknesses. It is beyond the scope of this paper to describe these other than to say that data sharing is not always optimal on a SAN particularly in heterogeneous system environments (i.e. NT/Windows versus UNIX-based systems).

NAS devices are very good at sharing data even in heterogeneous system environments. The problem that NAS devices run into in this area is performance. There is a significant amount of overhead involved in performing each data transfer between the requesting system and the storage device where the bits reside. Furthermore, the store-and-forward model used by virtually all NAS devices can become a problem if not used correctly.

In the OSD model, the protocol is system agnostic and therefore system heterogeneous by nature. Since the OSD *is* the storage device and the underlying protocol is supported on either a SAN (SCSI) or a LAN (iSCSI), device sharing becomes simple. Data sharing is accomplished as a result of this as well. The objects contained on an OSD are available to any system that has permission to access them. It is *interpretation* of the object that needs to be common among the systems that becomes important for effective data sharing. That interpretation is outside the scope of OSD but the ability to access the object is there.

Storage Performance

Performance requirements differ from application to application but they come down to three basic components that can be described as:

- Bandwidth – the number of bytes per second that can be transferred between the requesting system and the storage device
- Latency – the time from the receipt of a request until the first byte of data is received
- Transactions rate – how many transactions of a particular size can be processed each second

The performance of DAS can be managed fairly closely because there is only one system talking to the device at any given time. This system can therefore reorder the request queue to a DAS device to minimize latency, manage available bandwidth, and maximize the number of transactions per second.

Similarly, on a SAN, any given system is presumably one of many accessing a storage device at any given time. On an individual basis, any given system can realize the same performance as a DAS provided no other systems are using the target storage device or any other required resources (hubs, switch ports, …etc.). The device-sharing capability of SAN however, makes the task of managing the storage performance exceedingly difficult. This is because the storage devices cannot differentiate between access requests and thus cannot give preferential treatment to any single request or set of related requests. Therefore, the bandwidth, latency, and transaction rates are not manageable on a SAN without some knowledge of the requesting system or the data being accessed. Neither of these pieces of information is available to the device in a standard SAN configuration.

A NAS device can address some of these issues since it can know something about the files being accessed and the host requesting access. The practice of file "tagging" is used to identify certain performance characteristics of files when they are accessed. For example, if a high-definition video file is being read from a NAS device, it could know that is must transfer this file using 80MB/sec of 120MB/sec of available bandwidth on a specific network connection leaving the remaining 40MB/sec to transfer other files through that same network interface. This preferential treatment of requests has the effect of providing guaranteed bandwidth, latency, and/or transactions per second. But again, the tremendous overhead of NAS makes it difficult to compete with either DAS or SAN for raw performance in these three categories.

The OSD model is very performance conscious. It is designed to allow performance characteristics of objects to be an attribute of the object itself and independent of the OSD where it resides. If the high-definition video file given in the previous example were on an OSD, it would have an attribute that specified an 80MB/sec delivery rate as well as a certain quality of service (i.e. a consistent 80 MB/sec). Similarly, there could be different attributes for the same object that describe delivery performance for editing rather than playback. In editing-mode, the OSD may have to skip around to many different frames thus changing the way the OSD does caching and read-ahead. Similarly, for latency and transaction rates, an OSD can manage these more effectively than DAS and SAN because it has implicit and explicit knowledge of the objects it is managing. The NAS concept of "file-tagging" is generalized and extended in the OSD model to accommodate current applications as well as future unforeseen application performance and functionality requirements.

Scalability

The term scalability means many different things. Hence another term, *extensibility* will be used in this section to expand upon the term scalability. Many of the items listed under the heading of "extensibility" can be accomplished by NAS devices. It is a question of the degree at which a storage device is extensible that is important. The OSD model is a single open model, not a specific proprietary implementation that is intended to provide the fundamental architecture that can extend far into each of the extensibility dimensions yielding years of opportunity and growth of storage systems built on the OSD model.]

This is only a partial list of extensibility dimensions but it demonstrates the breadth of characteristics that the OSD model encompasses:

- Density – the number of bytes/IOPS/bandwidth per unit volume. OSD on individual storage devices can optimize these densities by abstracting the physical characteristics of the underlying storage medium and hardware to objects.
- Scalability – what does that word really mean?
    - Capacity: number of bytes, number of objects, number of files, …etc. OSD aggregation techniques will allow for hierarchical representations of more complex objects that consist of larger numbers of smaller objects.
    - Performance: Bandwidth, Transaction rate, Latency. OSD performance management can be used in conjunction with OSD aggregation techniques to more effectively scale each of these three performance metrics and maintain required QoS levels on a per-object basis.
    - Connectivity: number of disks, hosts, arrays, …etc. Since the OSD model requires self-managed devices and is transport agnostic the number of OSDs and hosts can grow to the size limits of the transport network.
    - Geographic: LAN, SAN, WAN, …etc. Again, since the OSD model is transport agnostic and since there is a security model built into the OSD architecture, the geographic scalability is not bounded.
    - Processing Power – Given that the OSD model promotes the development of Active Storage Device technology it is reasonable to consider scaling the processing power on an OSD to meet the requirements of the functions the Active Disk is expected to perform.
- Cost – address issues such as $/MB, $/sqft, $/IOP, $/MB/sec, TCO, …etc.
- Adaptability – to changing applications. Can the OSD be repurposed to different uses such as from a film editing station to mail serving?
- Capability – can add functionality for different applications. Can additional functionality be added to an OSD to increase its usefulness?
- Manageability – Can be managed as a system rather than just a box of storage devices – Aggregated OSD management? Hierarchical Storage management?
- Reliability – Connection integrity capabilities
- Availability – Fail-over capabilities between cooperating OSD devices. Can this scale from 2-way failover to N-way failover?
- Serviceability – Remote monitoring, remote servicing, hot-plug capability, genocidal sparing. When an OSD dies and a new one is put in it's place, how does it get "rebuilt"? How automated is the service process?
- Interoperability – Supported by many OS vendors, file system vendors, storage vendors, middleware vendors.
- Power – decrease the power per unit volume by relying on the policy-driven self management schemes to "power down" objects (i.e. move them to disks and spin those disks down).

The DAS and SAN devices run into significant problems with extending into many of these dimensions. Even though these systems are built from many of the same physical devices, it is the efficiency with which they can be used that is a true differentiator between DAS/SAN and NAS/OSD. As was previously mentioned in the Storage

Performance section, DAS/SAN devices have very good performance but cannot manage that performance effectively or efficiently. A NAS system has the potential to manage performance but suffers from other performance-related issues due to the file-level access protocols (NFS/CIFS) used with NAS subsystems. Many of these extensibility dimensions are "afterthoughts" and were never designed into the NAS model from the beginning.

On the other hand, it is these extensibility features that the OSD architecture is designed to exploit to allow vendors to build more application-specific storage-centric systems thereby allowing storage vendors to more easily differentiate their products to address application requirements. The OSD architecture was designed with extensibility in mind rather than as an afterthought.

How OSD Relates to File Systems – An example in Scalability
Current file system technologies that access disk drives directly are "block-based" in nature. These file systems are responsible for the management of all available disk blocks on the disk storage devices they manage. Hence, the "file system manager" is the program that runs on a computer system that manages all the data structures on a disk storage device that make up a "file system". The file system manager will perform file creation, data block allocation, tracking of which files occupy which data blocks, control of access to these files, file deletion, and management the list of free or unused data blocks. In performing these functions the file system manager examines and manipulates on-disk data structures such as information nodes (inodes) and directory trees.

The file system manager manages two basic types of data: "meta-data" and "user data". Meta-data constitutes the file system *structure* that ultimately contains the user data files. Therefore, the file system manager has the ability to understand the "structure" of the "file system" but not the contents of the user data contained in the file system. Also, from the point of view of the file system manager, a disk storage device is simply a sequential set of disk blocks where a disk block is typically 512 bytes. All the meta-data and user data is mapped into this sequential set of blocks. From the point of view of the storage device, it only knows how to access 512-byte blocks. The storage device has no concept of the structure of these blocks as it relates to the file system or the data contained within the blocks.

The problem with the model of a "block-based" file system is that it can be severely limited in scale. As the number of blocks in the file system grows the task of managing the location of all the files and associated user data blocks grows as well. In 2001 the 180GB disk drive was shipped that contained 360,000,000 disk blocks. Three of these disk drives would constitute over one billion blocks to manage. A terabyte-sized file system would be made up of two billion blocks and a 10-terabyte file system, which is not uncommon these days, would be 20 billion disk blocks.

The OSD model would move the management of these individual blocks to the devices themselves. The file system manager would then only need to manage objects – a far more manageable problem. The fact that a disk device has blocks is completely hidden

from anything outside the disk drive itself. In fact, it does not even have to be a "disk" drive. It could be a solid-state device, a MEMS device, or a quantum crystal device. It no longer matters to the file system manager as long as the device can store and retrieve "objects". Now the file system manager only needs to worry about managing 500,000 objects and the fact that they take up the equivalent of 30 trillion 512-byte blocks is no longer directly relevant.

Functionality

DAS and SAN devices do two things and only two things: they write data and the read data. This is the limit of their functionality. NAS devices can perform more complex tasks such as snapshot backups, hierarchical storage management, data replication, …etc. because the NAS devices know certain attributes of the files they manage. However, most NAS device protocols still lack the extensibility to know and more effectively act upon the data they store.

The OSD model extends beyond the simple attributes of a file and allows for application-specific attributes that can specify relationships to other objects to form structures or functional attributes that can instruct the OSD to perform some operation (i.e. compression, encryption, …etc) on an object. The OSD model is intended to be used with the concept of Active Disks [Acharya] or Active Storage Devices. These devices can have significantly greater functionality than a simple DAS/SAN/NAS device because they can implicitly or explicitly act on the data they store.

It is this concept of Active Storage Devices that makes OSD so compelling for users *and* storage vendors. The reason for this is simple: users need to spend more time working on and with their data than trying to figure out how to manage it. Storage vendors need to have some way to significantly differentiate their storage products in an increasingly commoditized storage market. OSD provides and extensible mechanism to facilitate the incorporation of unique functionality storage devices thereby differentiating them from other storage products based on their *capabilities* not simply bandwidth, transaction rate, or capacity. Furthermore, since these storage devices are intelligent, they can be self-managed, autonomous "appliances" that are tailored to meet the requirements (processing, performance, reliability, …etc.) of *specific* applications.

OSD Roadmap

The concept of OSD has been around and in development for the past 10 years. Much of this work was pioneered by Garth Gibson and his research team at the Parallel Data Lab at CMU funded in part by Seagate. Recently however, an OSD Technical Working group has been formed as part of the Storage Networking Industry Association (SNIA – www.snia.org). The charter of this group is to work on issues related to the OSD command subset of the SCSI command set and to enable the construction, demonstration, and evaluation of OSD prototypes over the next several years. The command specification is to a point where working prototypes have been demonstrated by companies such as Seagate and Intel but no production or enterprise-level products have resulted from these prototypes yet.

Summary

OSD is an enabling technology for the development of active storage devices. By allowing the storage devices to understand, interpret, and act upon the data they store, new classes of storage-centric devices can be brought to market that enhance customer workflows while reducing total cost of ownership. OSD can also allow for more highly differentiated storage products based on capabilities rather than simple capacity, or raw performance thereby enhancing a storage vendor's ability to serve their respective markets.

References

[1] E. Riedel, G. Gibson, and C. Faloutsos, "Active Storage for Large-Scale Data Mining and Multimedia", *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98),* August 1998

[2] K. Keeton, D. A. Patterson, J.. M. Hellerstein, "A case for Intelligent Disks (IDISKs)", *SIGMOD*, August 1998

[3] A. Acharya, M. Uysal, and J. Saltz, "Active Disks", *ASPLOS*, October 1998

[4] E. Riedel, G. Gibson, C. Faloutsos, G. Granger, D. Nagle, "Data Mining on an OLTP System (Nearly) for Free", *SIGMOD*, May 2000

[5] H. Gobioff, G. Gibson, and D. Tygar, "Security for Network Attached Storage Devices", *White Paper CMU-CS-97-185*, October 1997

[6] G. Gibson et al, "Filesystems for Network-Attached Secure Disks", *White Paper CMU-CS-97-118*, July 1997

[7] E. Borowsky et al, "Using Attribute-managed Storage to Achieve QoS", *Hewlett-Packard Laboratories White Paper*

[8] G. Gibson et al, "File Server Scaling weith Network-Attached Secure Disks", *SIGMETRICS '97*, June 1997

[9] E. Riedel (HP), G. Gibson, and C. Faloutsos, D. Nagle (CMU), Active Disks for Large-Scale Data Processing", *IEEE Computer,* June 2001