International Journal of Information Technology and Management

Vol. V, Issue No. I, August-2013, ISSN 2249-4510

A STUDY ON COST-EFFECTIVE, HIGH-BANDWIDTH STORAGE ARCHITECTURE

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

International Journal of Information Technology and Management
Vol. V, Issue No. I, August-2013, ISSN 2249-4510

A Study On Cost-Effective, High-Bandwidth Storage Architecture

Anuradha

Assistant Professor, Dronacharya Institute of Management & Technology, Kurukshetra, Haryana - 136118

Abstract – This paper describes the Network-Attached Secure Disk (NASD) storage architecture, prototype implementations of NASD drives, array management for our architecture and three file systems built on our prototype. NASD provides scalable storage bandwidth without the cost of servers used primarily for transferring data from peripheral networks e.g. SCSI to client networks e.g. ethernet. Increasing dataset sizes, new attachment technologies, the convergence of peripheral and inter-processor switched networks, and the increased availability of on-drive transistors motivate and enable this new architecture. NASD is based on four main principles: direct transfer to clients, secure interfaces via cryptographic support, asynchronous non-critical-path oversight, and variably-sized data objects. Measurements of our prototype system show that these services can be cost effectively integrated into a next generation disk drive ASIC. End-to-end measurements of our prototype drive and file systems suggest that NASD can support conventional distributed file systems without performance degradation. More importantly, we show scalable bandwidth for NASD-specialized file systems. Using a parallel data mining application, NASD drives deliver a linear scaling of 6.2 MB/s per client drive pair, tested with up to eight pairs in our lab.

Keywords: File systems management, Distributed systems, Input/output and Data Communications.

----------♦----------

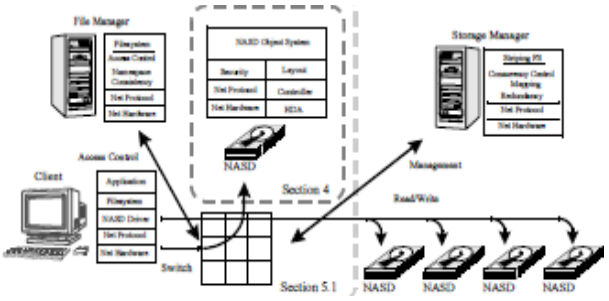
INTRODUCTION

Demands for storage bandwidth continue to grow due to rapidly increasing client performance, richer data types such as video and data-intensive applications such as data mining. For storage subsystems to deliver scalable bandwidth, that is, linearly increasing application bandwidth with increasing numbers of storage devices and client processors, the data must be striped over many disks and network links.

With 1998 technology, most office, engineering, and data processing shops have sufficient numbers of disks and scalable switched networking, but they access storage through storage controller and distributed fileserver bottlenecks. These bottlenecks arise because a single “server” computer receives data from the storage (peripheral) network and forwards it to the client (local area) network while adding functions such as concurrency control and metadata consistency. A variety of research projects have explored techniques for scaling the number of machines used to enforce the semantics of such controllers.

Scaling the number of machines devoted to store-and-forward copying of data from storage to client networks is expensive. This paper makes a case for new scalable bandwidth storage architecture, Network-Attached Secure Disks (NASD), which separates management and file system semantics from store-and-forward copying. By evolving the interface for commodity storage devices (SCSI-4 perhaps), we eliminate the server resources required solely for data movement. As with earlier generations of SCSI, the NASD interface is simple, efficient and flexible enough to support a wide range of file system semantics across multiple generations of technology. To demonstrate how NASD architecture can deliver scalable bandwidth, we describe a prototype implementation of NASD, a storage manager for NASD arrays, and a simple parallel file system that delivers scalable bandwidth to a parallel data-mining application.

Figure 1 illustrates the components of a NASD system and indicates the sections describing each.



OBJECTIVES OF THE STUDY

The objectives of the current study are as follows:

- To study the architecture of NASD architecture.
- To study the cost-effective, high bandwidth storage architecture.

REVIEW OF RELATED LITAERATURE

Aggarwal (2010) studied that if the server offers a simple file access interface to clients, the organization is known as a distributed file system. If the server processes data on behalf of the clients, this organization is a distributed database.

In organization, data makes a second network trip to the client and the server machine can become a bottleneck, particularly since it usually serves large numbers of disks to better amortize its cost.

Anderson (2006) explained the limitations of using a single central fileserver are widely recognized. Companies such as Auspex and Network Appliance have attempted to improve file server performance, specifically the number of clients supported, through the use of special purpose server hardware and highly optimized software.

Baker(2011) suggested a method to transparently improve storage bandwidth and reliability many systems interpose another computer, such as a RAID controller. This organization adds another peripheral network transfer and store-and-forward stage for data to traverse.

Provided that the distributed file system is reorganized to logically “DMA” data rather than copy it through its server, a fourth organization reduces the number of network transits for data to two. This organization has been examined extensively and is in use in the HPSS implementation of the Mass Storage Reference Model.

Benner(2006) studied that organization also applies to systems where clients are trusted to maintain file system metadata integrity and implement disk striping and redundancy. In this case, client caching of metadata can reduce the number of network transfers for control messages and data to two. Moreover, disks can be attached to client machines which are presumed to be independently paid for and generally idle. This eliminates additional store-and-forward cost, if clients are idle, without eliminating the copy itself.

Figure 2 illustrates the principal alternative storage architectures:

(1) A local file system,

(2) A distributed file system (DFS) built directly on disks,

(3) A distributed file system built on a storage subsystem,

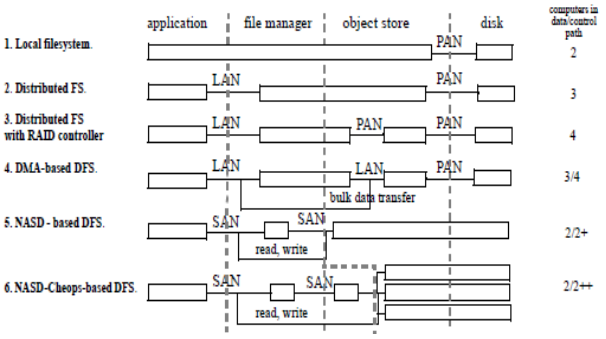
(4) A network-DMA distributed file system,

(5) A distributed file system using smart object-based disks (NASD) and

(6) A distributed file system using a second level of objects for storage management.

Acharya (2008) observed that the simplest organization aggregates the application, file management (naming, directories, access control, and concurrency control) and low-level storage management. Disk data makes one trip over a simple peripheral area network such as SCSI or Fiber channel and disks offer a fixed-size block abstraction. Stand-alone computer systems use this widely understood organization.

NASD architecture embeds the disk management functions into the device and offers a variable-length object storage interface. In this organization, file managers enable repeated client accesses to specific storage objects by granting a catchable capability.



once and there is no expensive store-and-forward computer.

The idea of a simple, disk-like network-attached storage server as a building block for high-level distributed file systems has been around for a long time. Cambridge’s Universal File Server (UFS) used an abstraction similar to NASD along with a directory-like index structure.

The UFS would reclaim any space that was not reachable from a root index. The successor project at Cambridge, CFS, also performed automatic reclamation and added undoable (for a period of time after initiation) transactions into the file system interface. To minimize coupling of file manager and device implementations, NASD offers less powerful semantics, with no automatic reclamation or transaction rollback.

Boden (2011) explained that NASD partitions are variable-sized groupings of objects, not physical regions of disk media, enabling the total partition

space to be managed easily, in a manner similar to virtual volumes or virtual disks.

We also believe that specific implementations can exploit NASD's uninterrupted file system-specific attribute fields to respond to higher-level capacity planning and reservation systems such as HP's attribute-managed storage.

Object-based storage is also being pursued for quality-of-service at the device, transparent performance optimizations, and drive supported data sharing.

ISI's Netstation project proposes a form of object-based storage called Derived Virtual Devices (DVD) in which the state of an open network connection is augmented with access control policies and object metadata, provided by the file manager using Kerberos for underlying security guarantees. This is similar to NASD's mechanism except that NASD's access control policies are embedded in unforgeable capabilities separate from communication state, so that their interpretation persists (as objects) when a connection is terminated. Moreover, Netstation's use of DVD as a physical partition server in VISA is not similar to our use of NASD as a single object server in a parallel distributed file system.

Cabrera (2011) studied that NASD security is based on capabilities, a well-established concept for regulating access to resources. In the past, many systems have used capabilities that rely on hardware support or trusted operating system kernels to protect system integrity. Within NASD, we make no assumptions about the integrity of the client to properly maintain capabilities. Therefore, we utilize cryptographic techniques similar to ISCAP and Amoeba. In these systems, both the entity issuing a capability and the entity validating a capability must share a large amount of private information about all of the issued capabilities. These systems are generally implemented as single entities issuing and validating capabilities, while in NASD these functions are done in distinct machines and no per-capability state is exchanged between issuer and validator.

RESEARCH METHODOLOGY

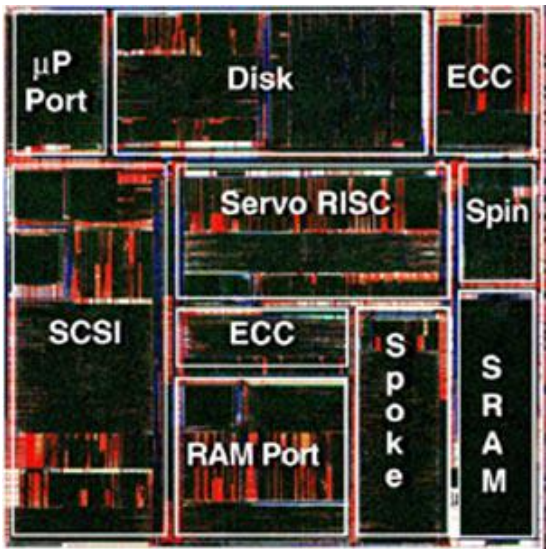
Storage architecture is ready to change as a result of the synergy between five overriding factors: I/O bound applications, new drive attachment technologies, an excess of ondrive transistors, the convergence of peripheral and inter-processor switched networks, and the cost of storage systems.

Traditional distributed file system workloads are dominated by small random accesses to small files whose sizes are growing with time, though not dramatically. In contrast, new workloads are much more I/O-bound, including data types such as video and audio, and applications such as data mining of retail transactions, medical records or telecommunication call records.

The same technology improvements that are increasing disk density by 60% per year are also driving up disk bandwidth at 40% per year. High transfer rates have increased pressure on the physical and electrical design of drive busses, dramatically reducing maximum bus length. At the same time, people are building systems of clustered computers with shared storage. For these reasons, the storage industry is moving toward encapsulating drive communication over Fibre channel, a serial, switched, packet-based peripheral network that supports long cable lengths, more ports, and more bandwidth. One impact of NASD is to evolve the SCSI command set that is currently being encapsulated over Fiber channel to take full advantage of the promises of that switched-network technology for both higher bandwidth and increased flexibility.

The increasing transistor density in inexpensive ASIC technology has allowed disk drive designers to lower cost and increase performance by integrating sophisticated special-purpose functional units into a small number of chips. Figure shows the block diagram for the ASIC at the heart of Quantum's Trident drive.

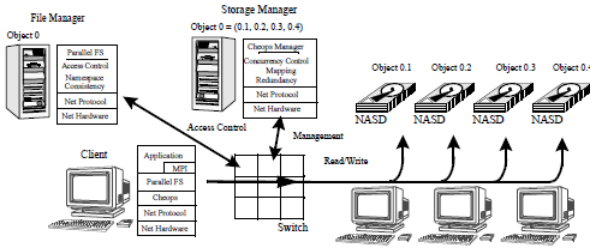
When drive ASIC technology advances from 0.68 micron CMOS to 0.35 micron CMOS, they could insert a 200 MHz Strong ARM microcontroller, leaving 100,000 gate-equivalent space for functions such as on-chip DRAM or cryptographic support. While this may seem like a major jump, Siemen's Tri Core integrated microcontroller and ASIC architecture promises to deliver a 100 MHz, 3-way issue,



In a NASD-adapted file system, files and directories are stored in NASD objects. The mapping of files and directories to objects depends upon the file system.

For our NFS and AFS ports, we use a simple approach: each file and each directory occupies exactly one NASD object and offsets in files are the same as offsets in objects. This allows common file attributes e.g. file length and last modify time to correspond directly to NASD-maintained object attributes. The remainder of the file attributes e.g. owner and mode bits are stored in the object's uninterrupted attributes. Because the file system makes policy decisions based on these file attributes, the client may not directly modify object metadata; commands that may impact policy decisions such as quota or access rights must go through the file manager.

The combination of a stateless server, weak cache consistency, and few file system management mechanisms make porting NFS to a NASD environment straightforward. Data moving operations (read, write) and attribute reads (getattr) are directed to the NASD drive while all other requests are handled by the file manager. Capabilities are piggybacked on the file manager's response to lookup operations. File attributes are either computed from NASD object attributes e.g. modify times and object size or stored in the uninterpreted file system-specific attribute.



AFS is a more complex distributed file system personality, but is also readily mapped to a NASD environment. As with NFS, data-moving requests (Fetch Data, Store Data) and attribute reads (Fetch Status, Bulk Status) are directed to NASD drives, while all other requests are sent to the file manager. Because AFS clients perform lookup operations by parsing directory files locally, there was no obvious operation on which to piggyback the issuing of capabilities so AFS RPCs were added to obtain and relinquish capabilities explicitly. AFS's sequential consistency is provided by breaking callbacks (notifying holders of potentially stale copies) when a write capability is issued. With NASD, the file manager no longer knows that a write operation arrived at a drive so must inform clients as soon as a write may occur. The issuing of new callbacks on a file with an outstanding write capability is blocked. Expiration times set by the file manager in every capability and the ability to directly invalidate capabilities allows file managers to bound the waiting time for a callback.

AFS also requires enforcement of a per-volume quota on allocated disk space. This is more difficult in NASD because quotas are logically managed by the file manager on each write but the file manager is not accessed on each write. However, because NASD has a byte range restriction in its capabilities, the file manager can create a write capability that escrows space for the file to grow by selecting a byte range larger than the current object. After the capability has been relinquished to the file manager (or has expired), the file manager can examine the object to determine its new size and update the quota data structures appropriately.

Both the NFS and AFS ports were straightforward. Specifically, transfers remain quite small, directory parsing in NFS is still done by the server, and the AFS server still has a concurrency limitations caused by its coroutine-based user level threads package. Our primary goal was to demonstrate that simple modifications to existing file systems allow NASD devices to be used without performance loss. Using the Andrew benchmark as a basis for comparison, we found that NASD-NFS and NFS had benchmark times within 5% of each other for configurations with 1 drive/1 client and 8 drives/8 clients. We do not report AFS numbers because the AFS server's severe concurrency limitations would make a comparison unfair.

NEED OF THE STUDY

To fully exploit the potential bandwidth in a NASD system, higher-level file systems should make large, parallel requests to files striped across multiple NASD drives. Cheops exports the same object interface as the underlying NASD devices and maintains the mapping of these higher-level objects to the objects on the individual devices. Our prototype system implements a Cheops client library that translates application requests and manages both levels of capabilities across multiple NASD drives. A separate Cheops *storage manager* (possibly co-located with the file manager) manages mappings for striped objects and supports concurrency control for multi-disk accesses.

The Cheops client and manager is less than 10,000 lines of code. To provide support for parallel applications, we implemented a simple parallel file system, NASD PFS, which offers the SIO low-level parallel file system interface and employs Cheops as its storage management layer. We used MPICH for communications within our parallel applications, while Cheops uses the DCE RPC mechanism required by our NASD prototype. To evaluate the performance of Cheops, we used a parallel data mining system that discovers association rules in sales.

ANALYSIS OF THE STUDY

We have implemented a working prototype of the NASD drive software running as a kernel module in Digital UNIX. Each NASD prototype drive runs on a DEC Alpha 3000/400 (133MHz, 64 MB, Digital UNIX 3.2g) with two Seagate ST52160 Medalist disks attached by two 5 MB/s SCSI busses. While this is

certainly a bulky "drive", the performance of this five year old machine is similar to what we predict will be available in drive controllers soon. We use two physical drives managed by a software striping driver to approximate the 10 MB/s rates we expect from more modern drives.

Because our prototype code is intended to operate directly in a drive, our NASD object system implements its own internal object access, cache, and disk space management modules (a total of 16,000 lines of code) and interacts minimally with Digital UNIX. For communications, our prototype uses DCE RPC 1.0.3 over UDP/IP. The implementation of these networking services is quite heavyweight. The appropriate protocol suite and implementation is currently an issue of active research.

CONCLUSION

Scalable storage bandwidth in clusters can be achieved by striping data over both storage devices and network links, provided that a switched network with sufficient bisection bandwidth exists. Unfortunately, the cost of the workstation server, network adapters, and peripheral adapters generally exceeds the cost of the storage devices, increasing the total cost by at least 80% over the cost of simply buying the storage.

We have presented a promising direction for the evolution of storage that transfers data directly on the client's network and dramatically reduces this cost overhead. Our scalable network-attached storage is defined by four properties. First, it must support direct device-to-client transfers. Second, it must provide secure interfaces (e.g. via cryptography). Third, it must support asynchronous oversight, whereby file managers provide clients with capabilities that allow them to issue authorized commands directly to devices. Fourth, devices must serve variable-length objects with separate attributes, rather than fixed-length blocks, to enable self-management and avoid the need to trust client operating systems.

To demonstrate these concepts, we have described the design and implementation of a NASD prototype that manages disks as efficiently as a UNIX file system. Measurements of this prototype show that available microprocessor cores embedded into the ASIC of a modern disk drive should provide more than adequate on-drive support for NASD, provided there is cryptographic hardware support for the security functions.

Using a simple parallel, distributed file system designed for NASD, we show that the NASD architecture can provide scalable bandwidth. We report our experiments with a data mining application for which we achieve 6.2 MB/s per client-drive pair in a system up to 8 drives, providing 45 MB/s overall.

REFERENCES

- Acharaya, A. et al, Active Disks, *ACM ASPLOS*, Oct 2008.
- Aggarwal (2010), R. and Srikant, R. Fast Algorithms for Mining Association Rules, *VLDB*, Sept 2010.
- Anderson et al. Serverless Network File Systems, *ACM TOCS* 14(1), Feb 2006.
- Baker, M.G. et al. Measurements of a Distributed File System", *ACM SOSP,* Oct 2011.
- Bellare (2006), Keying Hash Functions for Message Authentication, *Crypto '200*6.
- Benner, A.F., *Fibre Channel: Gigabit Communications and I/O for Computer Networks*, McGraw Hill, 2006.
- Boden et al. 2011: A Gigabit-per-Second Local Area Network, *IEEE Micro*, Feb 2011.
- Cabrera (2011), Swift: Using Distributed Disk Striping to Provide High I/O Data Rates, *Computing Systems* 4:4, Fall 1991.
- Corbett (2010) Proposal for a Common Parallel File System Programming Language, *Scalable I/O Initiative Cal Tech CACR 130*, Nov 2010.
- Dennis (2010), "Programming Semantics for Multi-programmed Computations", *CACM* 9, 3, 2010.