Appears in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics '97)*, Seattle, Washington, June 15-18, 1997.

File Server Scaling with Network-Attached Secure Disks

Garth A. Gibson†, David F. Nagle*, Khalil Amiri*, Fay W. Chang†, Eugene M. Feinberg*, Howard Gobioff†, Chen Lee†, Berend Ozceri*, Erik Riedel*, David Rochberg†, Jim Zelenka†

*Department of Electrical and Computer Engineering
†School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890
garth+nasd@cs.cmu.edu
http://www.cs.cmu.edu/Web/Groups/NASD/

Abstract

By providing direct data transfer between storage and client, network-attached storage devices have the potential to improve scalability for existing distributed file systems (by removing the server as a bottleneck) and bandwidth for new parallel and distributed file systems (through network striping and more efficient data paths). Together, these advantages influence a large enough fraction of the storage market to make commodity network-attached storage feasible. Realizing the technology's full potential requires careful consideration across a wide range of file system, networking and security issues. This paper contrasts two network-attached storage architectures—(1) Networked SCSI disks (NetSCSI) are network-attached storage devices with minimal changes from the familiar SCSI interface, while (2) Network-Attached Secure Disks (NASD) are drives that support independent client access to drive object services. To estimate the potential performance benefits of these architectures, we develop an analytic model and perform trace-driven replay experiments based on AFS and NFS traces. Our results suggest that NetSCSI can reduce file server load during a burst of NFS or AFS activity by about 30%. With the NASD architecture, server load (during burst activity) can be reduced by a factor of up to five for AFS and up to ten for NFS.

1 Introduction

Users are increasingly using distributed file systems to access data across local area networks; personal computers with hundred-plus MIPS processors are becoming increasingly affordable; and the sustained bandwidth of magnetic disk storage is expected to exceed 30 MB/s by the end of the decade. These trends place a pressing need on distributed file system architectures to provide clients with efficient, scalable, high-bandwidth access to stored data. This paper discusses a powerful approach to fulfilling this need. Network-attached storage provides high bandwidth by directly attaching storage to the network, avoiding file server store-and-forward operations and allowing data transfers to be striped over storage and switched-network links.

The principal contribution of this paper is to demonstrate the potential of network-attached storage devices for penetrating the markets defined by existing distributed file system clients, specifically the Network File System (NFS) and Andrew File System (AFS) distributed file system protocols. Our results suggest that network-attached storage devices can improve overall distributed file system cost-effectiveness by offloading disk access, storage management and network transfer and greatly reducing the amount of server work per byte accessed.

We begin by charting the range of network-attached storage devices that enable scalable, high-bandwidth storage systems. Specifically, we present a taxonomy of network-attached storage — server-attached disks (SAD), networked SCSI (NetSCSI) and network-attached secure disks (NASD) — and discuss the distributed file system functions offloaded to storage and the security models supportable by each.

With this taxonomy in place, we examine traces of requests on NFS and AFS file servers, measure the operation costs of commonly used SAD implementations of these file servers and develop a simple model of the change in manager costs for NFS and AFS in NetSCSI and NASD environments. Evaluating the impact on file server load analytically and in trace-driven replay experiments, we find that NASD promises much more efficient file server offloading in comparison to the simpler NetSCSI. With this potential benefit for existing distributed file server markets, we conclude that it is worthwhile to engage in detailed NASD implementation studies to demonstrate the efficiency, throughput and response time of distributed file systems using network-attached storage devices.

In Section 2, we discuss related work. Section 3 presents our taxonomy of network-attached storage architectures. In Section 4, we describe the NFS and AFS traces used in our analysis and replay experiments and report our measurements of the cost of each server operation in CPU cycles. Section 5 develops an analytic model to estimate the potential scaling offered by server-offloading in NetSCSI and NASD based on the collected traces and the measured costs of server operations. The trace-driven replay experiment and the results are the subject of Section 6. Finally, Section 7 presents our conclusions and discusses future directions.

This research was sponsored by DARPA/ITO through ARPA Order D306 under contract N00174-96-0002 and in part by an ONR graduate fellowship. The project team is indebted to generous contributions from the member companies of the Parallel Data Consortium: Hewlett-Packard, Symbios Logic Inc., Data General, Compaq, IBM Corporation, EMC Corporation, Seagate Technology, and Storage Technology Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any supporting organization or the U.S. Government.

2 Related Work

Distributed file systems provide remote access to shared file storage in a networked environment [Sandberg85, Howard88, Minshall94]. A principal measure of a distributed file system's cost is the computational power required from the servers to provide adequate performance for each client's work [Howard88, Nelson88]. While microprocessor performance is increasing dramatically and raw computational power would not normally be a concern, the work done by a file server is data- and interrupt-intensive and, with the poorer locality typical of operating systems, faster microprocessors will provide much less benefit than their cycle time trends promise [Ousterhout91, Anderson91, Chen93].

Typically, distributed file systems employ client caching to reduce this server load. For example, AFS clients use local disk to cache a subset of the global system's files. While client caching is essential for high performance, increasing file sizes, computation sizes, and workgroup sharing are all inducing more misses per cache block [Ousterhout85, Baker91]. At the same time, increased client cache sizes are making these misses more bursty.

When the post-client-cache server load is still too large, it can either be distributed over multiple servers or satisfied by a custom-designed high-end file server. Multiple-server distributed file systems attempt to balance load by partitioning the namespace and replicating static, commonly used files. This replication and partitioning is too often ad-hoc, leading to the "hotspot" problem familiar in multiple-disk mainframe systems [Kim86] and requiring frequent user-directed load balancing. Not surprisingly, custom-designed high-end file servers more reliably provide good performance, but can be an expensive solution [Hitz90, Drapeau94].

Experience with disk arrays suggests another solution. If data is striped over multiple independent disks of an array, then a high-concurrency workload will be balanced with high probability as long as individual accesses are small relative to the unit of interleaving [Linvy87, Patterson88, Chen90]. Similarly, striping file storage across multiple servers provides parallel transfer of large files and balancing of high concurrency workloads [Hartman93]; striping of metadata promises further load-balancing [Dahlin95].

Scalability prohibits the use of a single shared-media network; however, with the emergence of switched network fabrics based on high-speed point-to-point links, striped storage can scale bandwidth independent of other traffic in the same fabric [Arnould89, Siu95, Boden95]. Unfortunately, current implementations of Internet protocols demand significant processing power to deliver high bandwidth — we observe as much as 80% of a 233 MHz DEC Alpha consumed by UDP/IP receiving 135 Mbps over 155 Mbps ATM (even with adaptor support for packet reassembly). Improving this bandwidth depends on interface board designs [Steenkiste94, Cooper90], integrated layer processing for network protocols [Clark89], direct application access to the network interface [vonEiken92, Maeda93], copy avoiding buffering schemes [Druschel93, Brustoloni96], and routing support for high-performance best-effort traffic [Ma96, Traw95]. Perhaps most importantly, the protocol stacks resulting from these research efforts must be deployed widely. This deployment is critical because the comparable storage protocols, SCSI, and soon, Fibre Channel, provide cost-effective hardware implementations routinely included in client machines. For comparison, a 175 MHz DEC Alpha consumes less than 5% of its processing power fetching 100 Mbps from a 160 Mbps SCSI channel via the UNIX raw disk interface.

To exploit the economics of large systems resulting from the cobbling together of many client purchases, the xFS file system distributes code, metadata and data over all clients, eliminating the need for a centralized storage system [Dahlin95]. This scheme naturally matches increasing client performance with increasing server performance. Instead of reducing the server workload, however, it takes the required computational power from another, frequently idle, client. Complementing the advantages of filesystems such as xFS, the network-attached storage architectures presented in this paper significantly reduce the demand for server computation and eliminate file server machines from the storage data path, reducing the coupling between overall file system integrity and the security of individual client machines.

As distributed file system technology has improved, so have the storage technologies employed by these systems. Storage density increases, long a predictable 25% per year, have risen to 60% increases per year during the 90s. Data rates, which were constrained by storage interface definitions until the mid-80s, have increased by about 40% per year in the 90s [Grochowski96]. The acceptance, in all but the lowest cost market, of SCSI, whose interface exports the abstraction of a linear array of fixed-size blocks provided by an embedded controller [ANSI86], catalyzed rapid deployment of technology advances, resulting in an extremely competitive storage market.

The level of indirection introduced by SCSI has also led to transparent improvements in storage performance such as RAID; transparent failure recovery; real-time geometry-sensitive scheduling; buffer caching; read-ahead and write-behind; compression; dynamic mapping; and representation migration [Patterson88, Gibson92, Massiglia94, StorageTek94, Wilkes95, Ruemmler91, Varma95]. However, in order to overcome the speed, addressability and connectivity limitations of current SCSI implementations [Sachs94, ANSI95], the industry is turning to high-speed packetized interconnects such as Fibre Channel at up to 1 Gbps [Benner96]. The disk drive industry anticipates the marginal cost for on-disk Fibre Channel interfaces, relative to the common single-ended SCSI interface in use today, to be comparable to the marginal cost for high-performance differential SCSI (a difference similar to the cost of today's Ethernet adapters) while their host adapter costs are expected to be comparable to high-performance SCSI adapters [Anderson95].

The idea of simple, disk-like network-based storage servers whose functions are employed by higher-level distributed file systems, has been around for a long time [Birrel80, Katz92]. The Mass Storage System Reference Model (MSSRM), an early architecture for hierarchical storage subsystems, has advocated the separation of control and data paths for almost a decade [Miller88, IEEE94]. Using a high-bandwidth network that supports direct transfers for the data path is a natural consequence [Kronenberg86, Drapeau94, Long94, Lee95, Menascé96, VanMeter96]. The MSSRM has been implemented in the High Performance Storage System (HPSS) [Watson95] and augmented with socket-level striping of file transfers [Berdahl95, Wiltzius95], over the multiple network interfaces found on mainframes and supercomputers.[1]

[1]Following Van Meter's [VanMeter96] definition of network-attached peripherals, we consider only networks that are shared with general local area network traffic and not single-vendor systems whose interconnects are fast, isolated local area networks [Horst95, IEEE92].

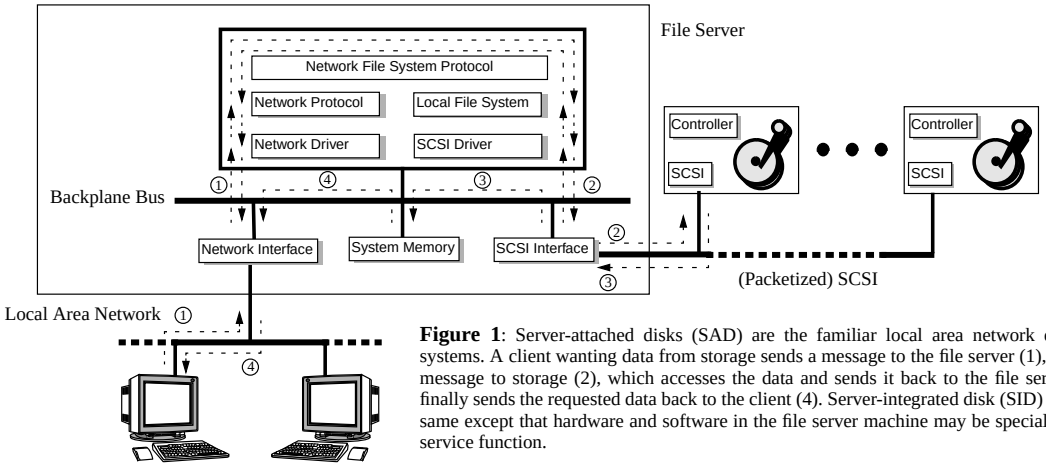

Figure 1: Server-attached disks (SAD) are the familiar local area network distributed file systems. A client wanting data from storage sends a message to the file server (1), which sends a message to storage (2), which accesses the data and sends it back to the file server (3), which finally sends the requested data back to the client (4). Server-integrated disk (SID) is logically the same except that hardware and software in the file server machine may be specialized to the file service function.

Striping data across multiple storage servers with independent ports into a scalable local area network has been advocated as a means of obtaining scalable storage bandwidth [Hartman93]. If the storage servers of this architecture are network-attached devices, rather than dedicated machines between the network and storage, efficiency is further improved by avoiding store-and-forward delays through the server.

Our notion of network-attached storage is consistent with these projects. However, our analysis focuses on the evolution of commodity storage devices rather than niche-market, very high-end systems, and on the interaction of network-attached storage with common distributed file systems. Because all prior work views the network-based storage as a function provided by an additional computer, instead of the storage devices itself, cost-effectiveness has never been within reach. Our goal is to chart the way network-attached storage is likely to appear in storage products, estimate its scalability implications, and characterize the security and file system design issues in its implementation.

3 Taxonomy of Network-Attached Storage

Simply attaching storage to a network underspecifies network-attached storage's role in distributed file systems' architectures. In the following subsections, we present a taxonomy for the functional composition of network-attached storage. Case 0, the base case, is the familiar local area network with storage privately connected to file server machines — we call this *server-attached disks*. Case 1 represents a wide variety of current products, *server-integrated disks*, that specialize hardware and software into an integrated file server product. In Case 2, the obvious network-attached disk design, *network SCSI*, minimizes modifications to the drive command interface, hardware and software. Finally, Case 3, *network-attached secure disks*, leverages the rapidly increasing processor capability of disk-embedded controllers to restructure the drive command interface.

3.1 Case 0: Server-Attached Disks (SAD)

This is the system familiar to office and campus local area networks as illustrated in Figure 1. Clients and servers share a network and storage is attached directly to general-purpose workstations that provide distributed file services.

3.2 Case 1: Server Integrated Disks (SID)

Since file server machines often do little other than service distributed file system requests, it makes sense to construct specialized systems that perform only file system functions and not general-purpose computation. This architecture is not fundamentally different from SAD. Data must still move through the server machine before it reaches the network, but specialized servers can move this data more efficiently than general-purpose machines. Since high performance distributed file service benefits the productivity of most users, this architecture occupies an important market niche [Hitz90, Hitz94]. However, this approach binds storage to a particular distributed file system, its semantics, and its performance characteristics. For example, most server-integrated disks provide NFS file service, whose inherent performance has long been criticized [Howard88]. Furthermore, this approach is undesirable because it does not enable distributed file system and storage technology to evolve independently. Server striping, for instance, is not easily supported by any of the currently popular distributed file systems. Binding the storage interface to a particular distributed file system hampers the integration of such new features [Birrell80].

3.3 Case 2: Network SCSI (NetSCSI)

The other end of the spectrum is to retain as much as possible of SCSI, the current dominant mid- and high-level storage device protocol. This is the natural evolution path for storage devices; Seagate's Barracuda FC is already providing packetized SCSI through Fibre Channel network ports to directly attached hosts [Seagate96]. NetSCSI is a network-attached storage architecture that makes minimal changes to the hardware and software of SCSI disks. File manager software translates client requests into commands to disks, but rather than returning data to the file manager to be forwarded, the NetSCSI disks send data directly to clients, similar to the support for third-party transfers already supported by SCSI [Drapeau94]. The efficient data transfer engines typical of fast drives ensure that the drive's sustained bandwidth is available to clients. Further, by eliminating the file manager from the data path, its workload per active client decreases. However, the use of third-party transfer changes the drive's role in the overall security of a distributed file system. While it is not unusual for distributed file systems to employ a security protocol between clients and

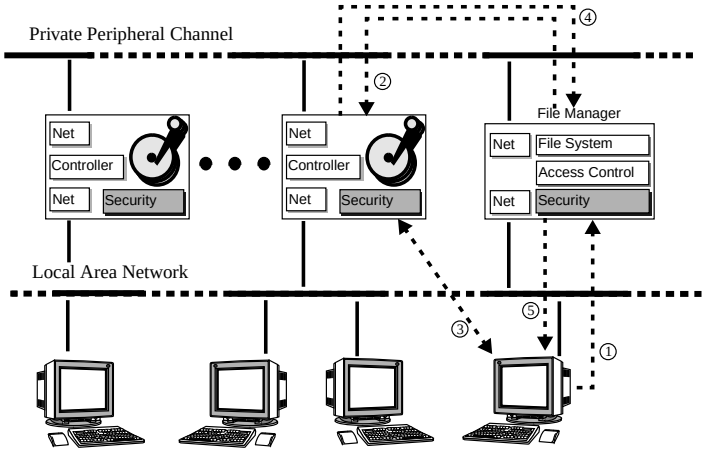

Figure 2: Network SCSI (NetSCSI) is a network-attached disk architecture designed for minimal changes to the disk's command interface. However, because the network port on these disks may be connected to a hostile, broader network, preserving the integrity of on-disk file system structure requires a second port to a private (file manager-owned) network or cryptographic support for a virtual private channel to the file manager. If a client wants data from a NetSCSI disk, it sends a message (1) to the distributed file system's file manager which processes the request in the usual way, sending a message over the private network to the NetSCSI disk (2). The disk accesses data, transfers it directly to the client (3), and sends its completion status to the file manager over the private network (4). Finally, the file manager completes the request with a status message to the client (5).

servers (e.g. Kerberos authentication), disk drives do not yet participate in this protocol.

We identify four levels of security within the NetSCSI model: (1) accident-avoidance with a second private network between file manager and disk, both locked in a physically secure room; (2) data transfer authentication with clients and drives equipped with a strong cryptographic hash function; (3) data transfer privacy with both clients and drives using encryption and; (4) secure key management with a secure coprocessor.

Figure 2 shows the simplest security enhancement to NetSCSI: a second network port on each disk. Since SCSI disks execute every command they receive without an explicit authorization check, without a second port even well-meaning clients can generate erroneous commands and accidentally damage parts of the file system. The drive's second network port provides protection from accidents while allowing SCSI command interpreters to continue following their normal execution model. This is the architecture employed in the SIOF and HPSS projects at LLNL [Wiltzius95, Watson95]. Assuming that file manager and NetSCSI disks are locked in a secure room, this mechanism is acceptable for the trusted network security model of NFS [Sandberg85].

Because file data still travels over the potentially hostile general network, NetSCSI disks are likely to demand greater security than simple accident avoidance. Cryptographic protocols can strengthen the security of NetSCSI. A strong cryptographic hash function, such as SHA [NIST94], computed at the drive and at the client would allow data transfer authentication (i.e., the correct data was received only if the sender and receiver compute the same hash on the data).

For some applications, data transfer authentication is insufficient, and communication privacy is required. To provide privacy, a NetSCSI drive must be able to encrypt and decrypt data. NetSCSI drives can use cryptographic protocols to construct private virtual channels over the untrusted network. However, since keys will be stored in devices vulnerable to physical attack, the servers must still be stored in physically secure environments. If we go one step further and equip NetSCSI disks with secure coprocessors [Tygar95], then keys can be protected and all data can be encrypted when outside the secure coprocessor, allowing the disks to be used in a variety of physically open environments. There are now a variety of secure coprocessors [NIST94a, Weingart87, White87, National96] available, some of which promise cryptographic accelerators sufficient to support single-disk bandwidths.

3.4 Case 3: Network-attached Secure Disks (NASD)

With network-attached secure disks, we relax the constraint of minimal change from the existing SCSI interface and implementation. Instead we focus on selecting a command interface that reduces the number of client-storage interactions that must be relayed through the file manager, offloading more of the file manager's work without integrating file system policy into the disk.

Common, data-intensive operations, such as reads and writes, go straight to the disk, while less-common ones, including namespace and access control manipulations, go to the file manager. As opposed to NetSCSI, where a significant part of the processing for security is performed on the file manager, NASD drives perform most of the processing to enforce the security policy. Specifically, the cryptographic functions and the enforcement of manager decisions are implemented at the drive, while policy decisions are made in the file manager. Because clients directly request access to data in their files, a NASD drive must have sufficient metadata to map and authorize the request to disk sectors. Authorization, in the form of a time-limited capability applicable to the file's map and contents, should be provided by the file manager to protect higher-level file systems' control over storage access policy. The storage mapping metadata, however, could be provided dynamically [VanMeter96a] by the file manager or could be maintained by the drive. While the latter approach asks distributed file system authors to surrender detailed control over the layout of the files they create, it enables smart drives to better exploit detailed knowledge of their own resources to optimize data layout, read-ahead, and cache management [deJonge93, Patterson95, Golding95]. This is precisely the type of value-added opportunity that nimble storage vendors can exploit for market and customer advantage. With mapping metadata at the drive controlling the layout of files, a NASD drive exports a namespace of file-like objects. Because control of naming is more appropriate to the higher-level file system, pathnames are not understood at the drive, and pathname resolution is split between the file manager and client. While a single drive object will suffice to represent a simple client file, multiple objects may be logically linked by the file system into one client file. Such an interface provides support for banks of

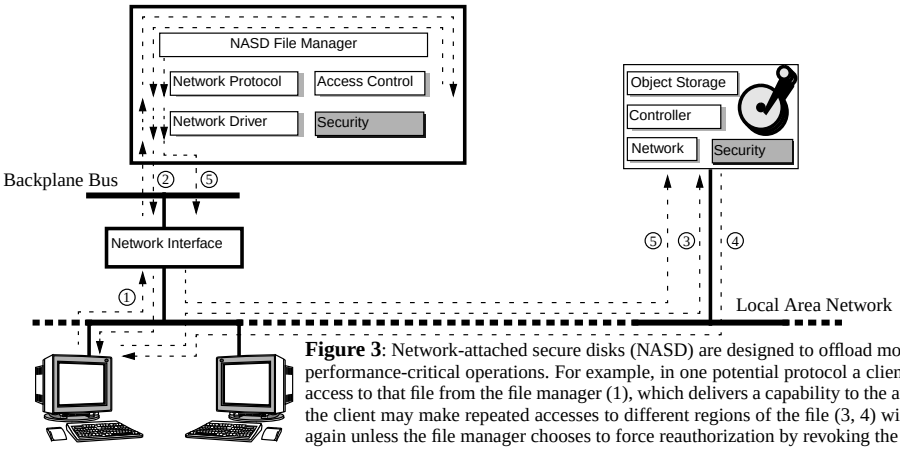

Figure 3: Network-attached secure disks (NASD) are designed to offload more of the file system's simple and performance-critical operations. For example, in one potential protocol a client, prior to reading a file, requests access to that file from the file manager (1), which delivers a capability to the authorized client (2). So equipped, the client may make repeated accesses to different regions of the file (3, 4) without contacting the file manager again unless the file manager chooses to force reauthorization by revoking the capability (5).

striped files [Hartman93], Macintosh-style resource forks, or logically-contiguous chunks of complex files [deJong93].

As an example of a possible NASD access sequence, consider a file read operation depicted in Figure 3. Before issuing its first read of a file, the client authenticates itself with the file manager and requests access to the file. If access is granted, the client receives the network location of the NASD drive containing the object and a time-limited capability to access the object and for establishing a secure communications channel with the drive. After this point, the client may directly request access to data on NASD drives, using the appropriate capability [Gobioff96].

In addition to offloading file read operations from the distributed file manager, later sections will show that NASD should also offload file writes and attributes reads to the drive. High-level file system policies, such as access control and cache consistency, however, remain the purview of the file manager. These policies are enforced by NASD drives according to the capabilities controlled by the file manager.

3.5 Summary

This taxonomy, summarized in Table 1, splits into two classes — SAD and SID offer a specific distributed file system while NetSCSI and NASD offer enhanced storage interfaces. The difference between SID and NASD merits further consideration. Many of the optimizations we propose for NASD, such as shortened data paths and specialized protocol processing, can also be implemented in a SID architecture. However, SID binds storage to a particular distributed file system, requires higher-level (or multiple-SID) file management to offer network striped files and, by not evolving the drive interface, inhibits the independent development of drive technology. For the rest of this paper, we focus on SAD, NetSCSI, and NASD and present a coarse-grained estimate of the potential benefit of network-attached storage. The results suggest that by exploiting the processing power available in next generation storage devices, computation required from the file manager machines can be dramatically reduced, enabling the per-byte cost of distributed file service to be reduced.

	Case 0	Case 1	Case 2	Case 3
FM per byte	X	X		
FM per operation			X	
FM on open/close				X
specialization		X	X	X

Table 1. Comparison of network-attached storage architectures. SAD and SID require the file manager (file server) to handle each byte of data, but SID allows specialization of the hardware and software to file service. NetSCSI allows direct transfers to clients, but requires file manager interaction on each operation to manage metadata.

4 Analysis of File System Workload

To develop an understanding of performance parameters critical to network-attached storage, we performed a series of measurements to (1) characterize the behavior and cost of AFS and NFS distributed file server functionality; and (2) identify and subset busy periods during which server load is limiting.

4.1 Trace Data

Our data is taken from NFS and AFS file system traces summarized in Table 2. The NFS trace [Dahlin94] records the activity of an Auspex file server supporting 231 client machines over a one week period at the University of California at Berkeley[2]. The AFS trace records the activity of our laboratory's Sparcstation 20 AFS server supporting 250 client machines over a one month period[3]

	NFS trace	AFS trace
Number of client machines	231	250
Total number of requests	6,676,479	1,615,540
Read data transferred (GB)	8.1	2.9
Write data transferred (GB)	2.0	1.6
Trace period	9/20/93-9/24/93 40 hours	9/9/96-10/3/96 435 hours[3]

Table 2. Description of the traces used in the experiments. The NFS trace was collected in a study performed at the University of California at Berkeley. The AFS trace was collected by logging requests at the AFS file server in our laboratory.

[2]Some attribute reads were removed from the NFS trace by the Berkeley researchers based on a heuristic for eliminating excessive cache consistency traffic. Because this change is pessimistic to our proposed architecture, we choose to continue to use these traces, already familiar to the community, rather than collect new traces.

[3]The trace covers three periods of activity - 9/9-10, 9/13-15, and 9/20-10/3.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.