# Understanding Customer Dissatisfaction With Underutilized Distributed File Servers

Erik Riedel[1]
Garth Gibson

July 1996
CMU-CS-96-158

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213-3890

To appear in the *Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies*, September 17-19, 1996 College Park, MD.

## Abstract

An important trend in the design of storage subsystems is a move toward direct network attachment. Network-attached storage offers the opportunity to off-load distributed file system functionality from dedicated file server machines and execute many requests directly at the storage devices. For this strategy to lead to better performance as perceived by users, the response time of distributed operations must improve. In this paper, we analyze measurements of an Andrew File System (AFS) server that we recently upgraded in an effort to improve client performance in our laboratory. While the original server's overall utilization was only about 3%, we show how burst loads were sufficiently intense to lead to periods of poor response time significant enough to trigger customer dissatisfaction. In particular, we show how, after adjusting for network load and traffic to non-project servers, 50% of the variation in client response time was explained by variation in server CPU utilization. That is, clients saw long response times in large part because the server was often over-utilized when it was used at all. Using these measures, we see that off-loading file server work in a network-attached storage architecture has the potential to benefit user response time. Computational power in such a system scales directly with storage capacity, so the slowdown during burst periods should be reduced.

This research is sponsored by DARPA/ITO through ARPA Order D306, and issued by Indian Head Division, NSWC under contract N00174-96-0002. The project team is indebted to generous contributions from the member companies of the Parallel Data Consortium. At the time of this writing, these companies include Hewlett-Packard Laboratories, Symbios Logic Inc., Data General, IBM Corporation, Seagate Technology, EMC Corporation, Storage Technology Corporation, and Digital Equipment 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.

[1] Department of Electrical and Computer Engineering, Carnegie Mellon University

**Keywords:** file server performance, network-attached storage, distributed file systems

# 1. Introduction

Recent trends in the computer industry have greatly increased the demands for common, shared information repositories. In most cases, these have taken the form of distributed file systems that are shared across a workgroup, organization-wide, or even world-wide. A distributed file system, with a number of machines acting as "servers" and a much larger number of "clients" have become popular due to a number of factors, including separation of administrative concerns, sharing of data, and transparency [Spasojevic96].

Advances in other computing technologies have made possible many novel applications that are placing increasing demands on distributed storage systems. The delivery of video and audio, large-scale parallel applications, and the growth of the Internet have increased demands on distributed information systems both in terms of the resources required by individual applications and the aggregate demands made by a continually increasing number of clients.

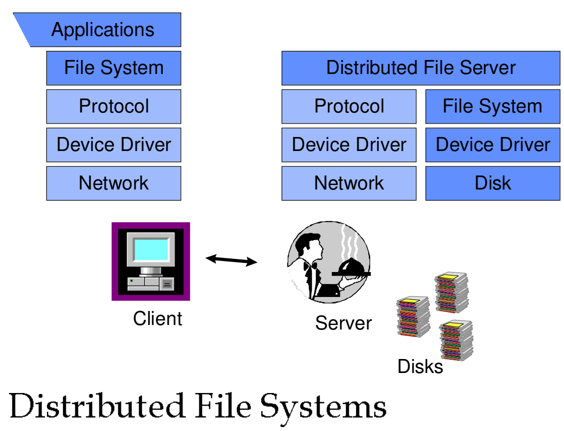


**Figure 1 - Traditional Distributed File System**

At the core of all distributed information systems lies a set of server resources that are becoming increasingly loaded as the demands increase. A traditional distributed file system model, where "storage" is simply embodied in the disk and device driver, is illustrated in Figure 1. This picture explains in part why increasing load on distributed file systems often requires fast file servers - the file server must traverse two protocol stacks for each client request. Data must move from attached disk drives, across the SCSI bus, through the server's memory system, back across the system bus, down the network protocol stack and, finally, onto the network wire. The server has very little "interest" in the data, yet it must move it through its memory hierarchy - possibly several times - in order to satisfy all the protocol layers involved.

In conjunction with this pressure toward using faster machines as file servers, recent years have seen rapid development, both in terms of areal density and in the raw bandwidth that can be provided off the platters of fixed storage devices. On top of these trends, perhaps the largest change comes from standardizing storage interfaces. The adoption of the SCSI interface for storage devices allowed storage vendors to optimize below a common protocol, and application and file system developers to optimize above it. By specifying a separate high-level "logical" interface and a physical interface, SCSI made possible numerous optimizations inside disk controllers including RAID, transparent recovery management, dynamic remapping, and storage migration. A common interface to operating system software allowed users to buy drives based on price and performance, rather than on compatibility requirements with other parts of their computer systems. This model has led to typical, high-performance distributed file systems that today look more like Figure 2. There is one interconnect for communication between clients and servers (IP or IPX over ATM or Ethernet), and another for communication between servers and disks (SCSI).

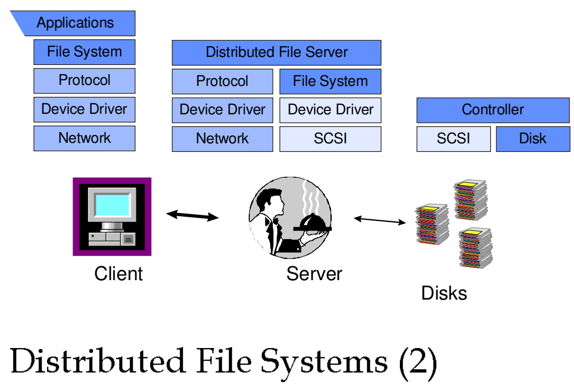


**Figure 2 - Actual Distributed File System Architecture Today**

The difficulty with this architecture is that a good portion of the overall system power is "dissipated" in the server system that bridges the gap between SCSI and the distributed file system protocol used by clients. With relatively slow storage devices and relatively slow networks, this additional overhead has until now been hidden among other limitations. The continued development of disk technology has made possible products with sustained data rates of up to 12 MB/s shipping today and 40 MB/s does not look unreasonable by the end of the decade. Fibre Channel interconnects also eliminate the traditional SCSI bus as a bottleneck. ATM, Fast Ethernet, and Myrinet provide client network rates of 12 MB/s today and 100 MB/s in the near future. These advances mean that the amount of room to "hide" inefficiencies in distributed file server implementations is shrinking dramatically.

The study described in the rest of this paper examines the requirements placed on file server architectures by studying the behavior of current distributed file system technology. Specifically, we have analyzed the system-level behavior of an AFS (Andrew File System) server in our environment. The following sections will present the behavior we have observed and the pressure on file server performance.

Section 2 provides a brief overview of AFS and presents our measurement methodology, tools, and environment. Section 3 provides a summary of some of the workload characteristics we observed. Section 4 discusses the factors that affect AFS performance as perceived by users. Section 5 discusses the potential available through the use of network-attached storage devices. Finally, we conclude in Section 6 and discuss avenues of future work.

# 2. Experimental Methodology

## *2.1. Andrew File System*

At Carnegie Mellon (and at hundreds of other large institutions around the world) the Andrew File System is used by nearly all computer users. The major contribution of AFS over previous distributed file systems such as the Network File System (NFS), was the focus on scalability of server resources. The goal of AFS was to support a campus-wide network of workstations and users with a relatively small amount of file server resources [Howard88]. The primary way in which AFS addressed this goal is through the use of local disk for extensive client-side caching. Each client workstation in an AFS environment dedicates a portion of its local disk space as a cache for frequently accessed remote data. Data in client caches is kept up-to-date through the use of a strong consistency protocol based on callbacks. When a client accesses a particular file from an AFS server, the server marks a callback for that data and client and promises to inform the client when the data is changed. Rather than having a large number of clients constantly checking in at the file server to see if data has changed, the responsibility for cache invalidation lies with the server.[2]

In the Spring of 1996, our lab upgraded its AFS server in response to our users' complaints about AFS performance. A major motivation in writing this paper is to identify and detail the performance reasons behind the upgrade and determine the implications for AFS distributed file systems built on network-attached storage architectures.

## *2.2. Measurement Environment*

The measurements reported here were taken from a single file server over the course of a two month period at the beginning of 1996. This server contained all of the project volumes used for research in the Parallel Data Laboratory (PDL). The server was a Sun SPARCstation 4/60 with 24 MB of memory serving 20 volumes representing a total of 8 GB of data in 4 partitions. The clients were fifteen Alpha AXP machines (Turbochannel models 300, 400, 500 and 600 and PCI models 200 and 400), nine IBM RS/6000 250s located in a single laboratory, and fifteen additional machines of varying types, ranging in power from DECstation 5000s to a SPARCstation 20, in this lab and in the offices of students and faculty. The workload, a diverse set of activities one would expect from a medium-sized research group, included software development, document preparation, data analysis and simulation.

The School of Computer Science network, to which all these machines are connected, consists of an Ethernet segment for each floor of its building, with an additional segment for the central machine room where all AFS servers are housed, all of which are connected to a single bridged backbone. The *cs.cmu.edu* AFS cell, in which our

[2] Another goal of AFS is to serve as a wide-area distributed file system that can span the entire globe. In order to facilitate this, AFS provides a single global namespace that is divided at the top level of the hierarchy into a number of *cells*, each of which represents a specific organization or administrative domain. The basic unit of distribution in AFS is a *volume*, a related set of files assigned for a specific purpose and representing a specific allocation of disk space. Each cell contains a set of well-known *database server* machines that maintain a mapping of which volumes reside on which of a number of *file server* machines. The file server machines have disks attached that are divided into logical *partitions*, each of which holds some number of volumes.

measurements where taken, consists of 25 (primarily SPARCstation) dedicated servers providing home directories; repositories for shared, locally-maintained software collections, and volumes assigned to specific research projects. Larger projects often "own" an entire server which houses all of that group's project volumes. The server under test was running AFS 3.2 with local patches under the Mach 2.6 operating system and the clients were running several different operating systems with AFS versions ranging from locally modified 3.1 to 3.4beta.

### 2.3. Analysis Tools

Traces of file server activity were taken with the aid of a tracing package developed by the Coda group at Carnegie Mellon [Mummert94]. A number of trace points, including most system calls, all accesses into the buffer cache, and all disk requests, within the operating system were annotated with log entries. Logs were collected in a kernel buffer and periodically extracted and shipped over the network to a second machine that gathered the traces on its local disk and periodically transfer them to tape. This facility allowed the collection of very detailed system traces without much effect on performance. The Coda group measured a performance impact of between five and seven percent in their studies. Traces were collected almost continuously over a two month period resulting in over 4 GB of data.

In addition to this data, client and server AFS activity was measured through the use of the AFS *xstat* facility which collected hourly summaries of operations performed, aggregate performance per operation type, as well as details on request sizes.[3] We also used *rxdebug* and *vos* to collect information on active clients and volume use patterns from the server. Statistics of the server and clients over three months represented an additional 400 MB of raw data.

To track performance of the network connecting our machines, we collected statistics derived from a periodic measurement of the round-trip time to the server and client network segments. Our measurement machine (ozone) executed a 30-second ping every 5 minutes noting the average round-trip time and packet loss rate to a selected number of clients (one on each floor with client machines) and to the server.

We developed a set of scripts to process the trace and summary data and used the Matlab numerical computation and visualization system to provide plots and statistical tests. In the following sections we will provide plots of measured data as well as means, variances, and Pearson $r$ correlation coefficients, and $r^2$ coefficients of determination. We use the Pearson coefficient of determination to quantify how much of the variation in a set of measurements can be accounted for by the characteristics of underlying system factors [Kirk90].

## 3. Workload Characteristics

In this section, we summarize a number of basic parameters of the workload recorded in our traces. Specifically the effectiveness of client caches, the mix of AFS operations at both the clients and our server, and the transfer size distributions at the server.

[3] Due to the highly distributed nature of AFS and our desire to measure a real workload, it was not possible to track all of the clients that made requests to this particular server, nor can we determine exactly what client activity was directed to this particular server. This introduces some amount of "noise" into our data, making some variations more difficult to explain.

### 3.1. Client Caching

As shown in previous work, the hit ratio for data in the local AFS cache is extremely good [Spasojevic96, Howard88]. Table 1 gives the average hourly hit ratio across the twenty clients for which we have the most complete data. This data emphasizes the well-established fact that there is a high degree of temporal locality in user access streams, and that local disk caching in AFS removes a considerable burden from the file server. The data shown represents measurements from a single week of traces - specifically the week of January 29, 1996 to February 4, 1996. This representative week-long period will be used throughout the rest of the paper.

| | Average | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 97.0 | 99.6 | 98.9 | 92.7 | 94.2 | 90.5 | 99.4 | 81.7 | 95.8 | 98.8 | 96.5 |
| metadata | 61.8 | 98.0 | 81.5 | 36.9 | 15.6 | 22.1 | 76.9 | 16.6 | 17.5 | 33.9 | 20.8 |
| | | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| data | | 99.1 | 99.4 | 98.7 | 99.9 | 99.6 | 99.3 | 99.5 | 99.3 | 98.4 | 99.1 |
| metadata | | 62.3 | 22.9 | 45.0 | 99.9 | 98.2 | 99.3 | 99.1 | 96.9 | 99.0 | 95.2 |

**Table 1 - Client Cache Hit Ratio**

### 3.2. Operation Distribution

Table 2 shows a breakdown of the most frequently used AFS operations and their relative popularity. The Clients column shows the total for the 20 clients reported above over the course of the same week. Note that the number of client and server requests does not match up because this is not a closed system - there were additional clients making requests of the PDL server, and the PDL clients made use of other AFS servers (as we will discuss in more detail later). The total amount of data transferred by clients was 993 MB in FetchData requests and 520 MB in StoreData requests. The server provided a total of 750 MB of data via FetchData and accepted 955 MB via StoreData requests.

| AFS Operation | Clients | | Server | |
|---|---|---|---|---|
| | total | fraction | total | fraction |
| FetchStatus | 748,620 | 68.0% | 412,695 | 43.4% |
| StoreStatus | 20,085 | 1.8% | 22,642 | 2.4% |
| FetchData | 174,717 | 15.9% | 62,288 | 6.5% |
| StoreData | 46,630 | 4.2% | 32,414 | 3.4% |
| CreateFile | 15,407 | 1.4% | 17,089 | 1.8% |
| RemoveFile | 17,242 | 1.6% | 20,422 | 2.1% |
| BulkStatus | 0 | 0.0% | 244,636 | 25.7% |
| GetTime | 50,568 | 4.6% | 122,393 | 12.9% |
| GiveUpCallbacks | 28,343 | 2.6% | 17,298 | 1.8% |
| total | 1,101,612 | | 951,877 | |

**Table 2 - Distribution of AFS Operations**

### 3.3. Request Sizes

Table 3 shows the distribution of request sizes over the course of a week. As seen in previous studies, small requests dominate the mix, while most of the bytes are moved in large requests [Spasojevic96, Baker91]. 80% of reads and 65% of writes are for less than 8 kilobytes. However, for StoreData requests, more than two-thirds of the bytes are moved at the largest request size. This means that system designers must consider optimizations that maximize the bandwidth of the largest requests without adversely affecting the latency of the majority of small operations.

| Request Size | FetchData | | StoreData | |
|---|---|---|---|---|
| up to 128 bytes | 19,503 | 31.3% | 7,607 | 23.5% |
| 129 bytes to 1 K | 3,663 | 5.9% | 3,196 | 9.9% |
| 1 K to 8 K | 24,858 | 39.9% | 10,035 | 31.0% |
| 8 K to 16 K | 2,127 | 3.4% | 2,244 | 6.9% |
| 16 K to 32 K | 1,889 | 3.0% | 2,510 | 7.8% |
| more than 32 K | 10,245 | 16.4% | 6,789 | 21.0% |
| total | 62,285 | | 32,381 | |

**Table 3 - Distribution of Request Sizes**

## 4. Impacts on User-Perceived Performance

### 4.1. Server Utilization

These statistics provide some idea of the typical work being performed by an AFS file server, but how does the performance of the server figure into customer purchasing and system sizing decisions? The Parallel Data Laboratory recently upgraded its AFS server from a dedicated SPARCstation 1 to a brand-new dedicated SPARCstation 20 with about 5 times the rated performance. This upgrade was done to a large extent in response to the increasingly vocal complaints of slow performance by our users. In fact, little data was consulted in the decision to upgrade this server. In an attempt to understand what effect the resources available on our server has on user performance, we took a look at the load on the original server after the upgrade. Given the traces described above, we can in hindsight attempt to better understand how server load relates to file system performance and customer satisfaction.

The top chart of Figure 3 shows the fraction of the server CPU spent in the AFS *fileserver* process over the course of a week, averaged over ten minute intervals. As we can see, the CPU on the server is mostly idle. Although we do see a number of peak periods in which the utilization reaches as high as 65%, the mean CPU utilization is less than 3%. This is a disturbing result. Were we wrong to spend about $10,000 for a new, fast file server to replace a slow, inexpensive server that is only 3% utilized?

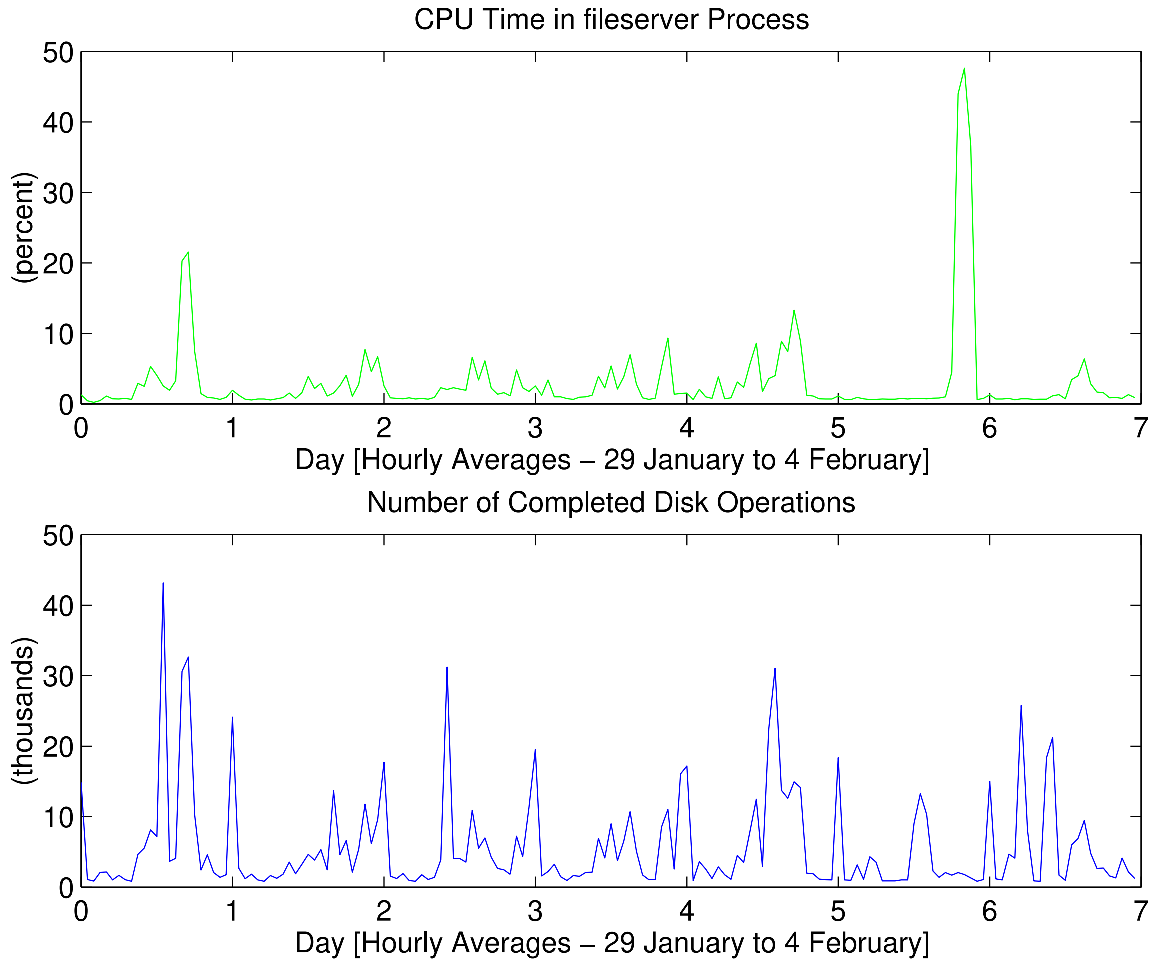


**Figure 3 - CPU and Disk Utilization**

A similar effect is seen in the plot of disk activity in the lower chart of Figure 3. This chart shows the total number of physical disk accesses completed in each of the same 10 minute intervals. It is harder to talk about percentage utilization in this case, but the three drives on this server should be able to sustain considerably more than the 50,000 accesses/hour (14 accesses/second) that correspond to the highest point on the chart. The average is less than one access/second over three disks. Again, a negligible total average load.

Simply looking at these numbers, we might be tempted to conclude that this five year old machine is performing adequately and there is no need for an upgrade at all.[4] So how do we explain our users' complaints? We clearly needed some other measure that we could use to gauge users' perception of the performance of the system. Since overall utilization is not the problem, we surmised that looking at response time might prove more enlightening.

[4] In fact, the upgrade policy at large AFS sites is rumored to be generally insensitive to utilization as well. The algorithm used can roughly be paraphrased as, when customers complain, begin with the oldest component of the system and continue to replace equipment with newer models until complaints subside.

### 4.2. Client Response Time

The client data that we collected provided hourly samples of the number and total elapsed time of all AFS operations of each type completed by that client in that hour. We chose to use the average response time for FetchStatus operations as our measure of user-visible performance because 1) it is the most frequently-called operation, 2) in the absence of outside influences, it does an approximately constant amount of work on each call (since data fetches in AFS may be as large as several hundred kilobytes, but most files are much smaller than this, FetchData delays are expected to be much more variable) and 3) we found an $r^2$ coefficient of determination suggesting that 50% of the variation in the response times of FetchStatus and the per-kilobyte latencies of FetchData are correlated, as shown in Figure 4.

If we again look at the average response time in Figure 4, we see significant variation - ranging over an order of magnitude. We hypothesize that users of AFS, accustomed to local disk access times (due to high local cache hit ratios described above) will be significantly affected by high variance in response times, particularly when the effect lasts for significant lengths of time, such as the hourly intervals shown in this chart. Based on this, we began searching for the causes of high variance in user response time.

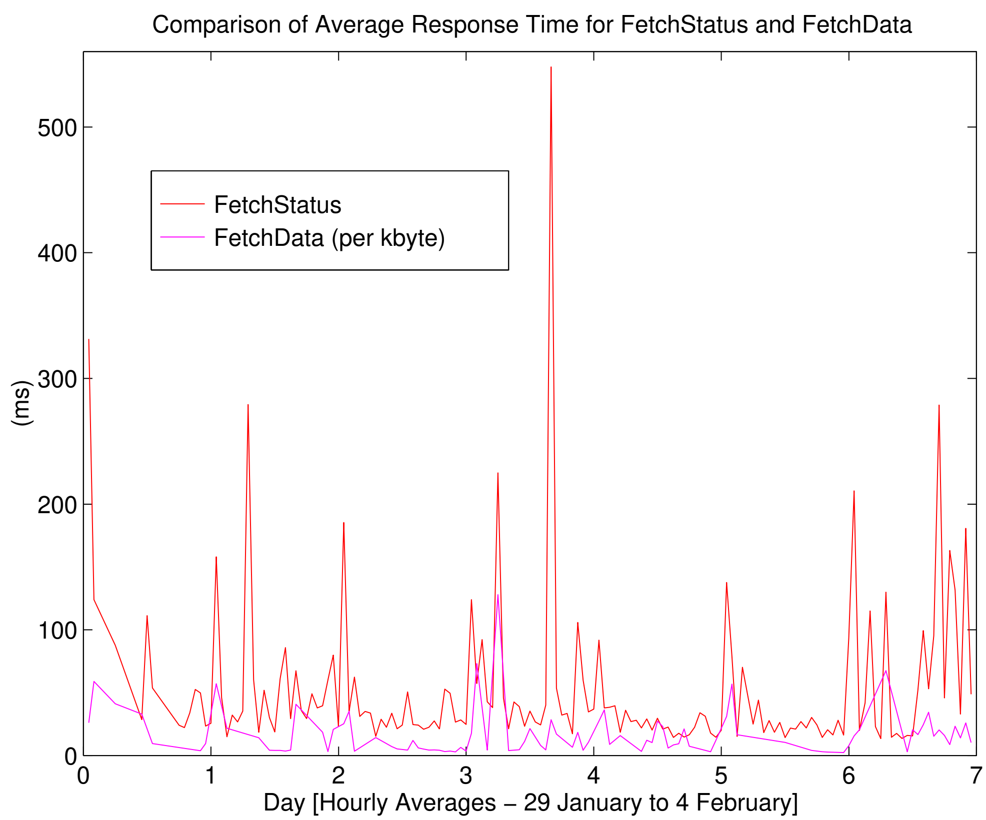


**Figure 4 - FetchStatus and FetchData Performance**

In order to convince ourselves that our AFS server upgrade had indeed been worthwhile, we performed an experiment to compare the performance of our old server and our new server under the same workload. The numbers in Table 4 show the results of this controlled experiment. One test client was constantly performing **`stat()`** calls at random into a directory of 2,000 files. At the same time, a second client was running a "competing" workload by continuously reading a large file from the same partition on the same server. Both clients flushed their caches at the end of a cycle so that all operations were handled at the server. The table shows the average response time of the FetchStatus operations that resulted from the `stat()` calls, the number of FetchStatus operations completed in the five minute measuring interval, and the average throughput of the competing process.

| Machine | SPECint92 | Average FetchStatus Response Time (ms) | Number of Operations | Competing Read Transfer (KB/s) |
|---|---|---|---|---|
| SPARCstation 1+ | 14.0 | 25.9 | 8,486 | 212.7 |
| SPARCstation 20 | 69.0 | 16.7 | 15,291 | 343.8 |

**Table 4 - Direct Comparison of Server Platforms**

From this experiment, we see that the increased CPU performance of the newer machine reduces average FetchStatus response time by 35% at periods of high server load. At the same time, the faster machine can complete almost twice as many FetchStatus operations in the same time interval while also providing 62% higher data throughput. Since more server processing power is clearly effective for improving client performance, we expect to be able to find a dependence between server CPU utilization and client response time in our trace data.

### 4.3. Impact of the Network

When we first compared the CPU and disk utilization trace to the FetchStatus response time trace, we were unable to find a significant correlation between times of slow user response and times of high server utilization. This unintuitive result led us to look for other factors that might explain performance at the clients. The most obvious factor in a distributed system is the network between machines, so this is the parameter we examined next.

The top chart of Figure 5 shows the average network round-trip time of pings on the lab and machine room Ethernet segments over one hour periods. We see a mean of 9.0 ms and a standard deviation of 7.2 ms on the server network, and 16.9 ms 15.8 ms on the lab segment, where most of the clients were located. The lower left portion of Figure 5 shows the graphical correlation between the response time of the network and FetchStatus response time.[5] Although not a strictly linear relationship, the Pearson $r^2$ coefficient suggests that 35% of the variation in the response times can be attributed to variation in network performance. To focus on this relationship, the correlation graph in the lower right of Figure 5 reports only those hours where average ping time was larger than 20 ms. In this figure, a linear relationship between server response time and network response time is more plausible. This matches our expectations that the network connecting the machines in a distributed system is a considerable factor in overall performance. It is for this reason that the new server and many of our clients are being outfitted with switched ATM networking dedicated to the PDL in addition to the existing

[5] Directly correlated data, with 100% of the variation explained, would appear as a straight line on these graphs.

Ethernet. However, we also see that network response time is not a complete explanation of client response time variance.

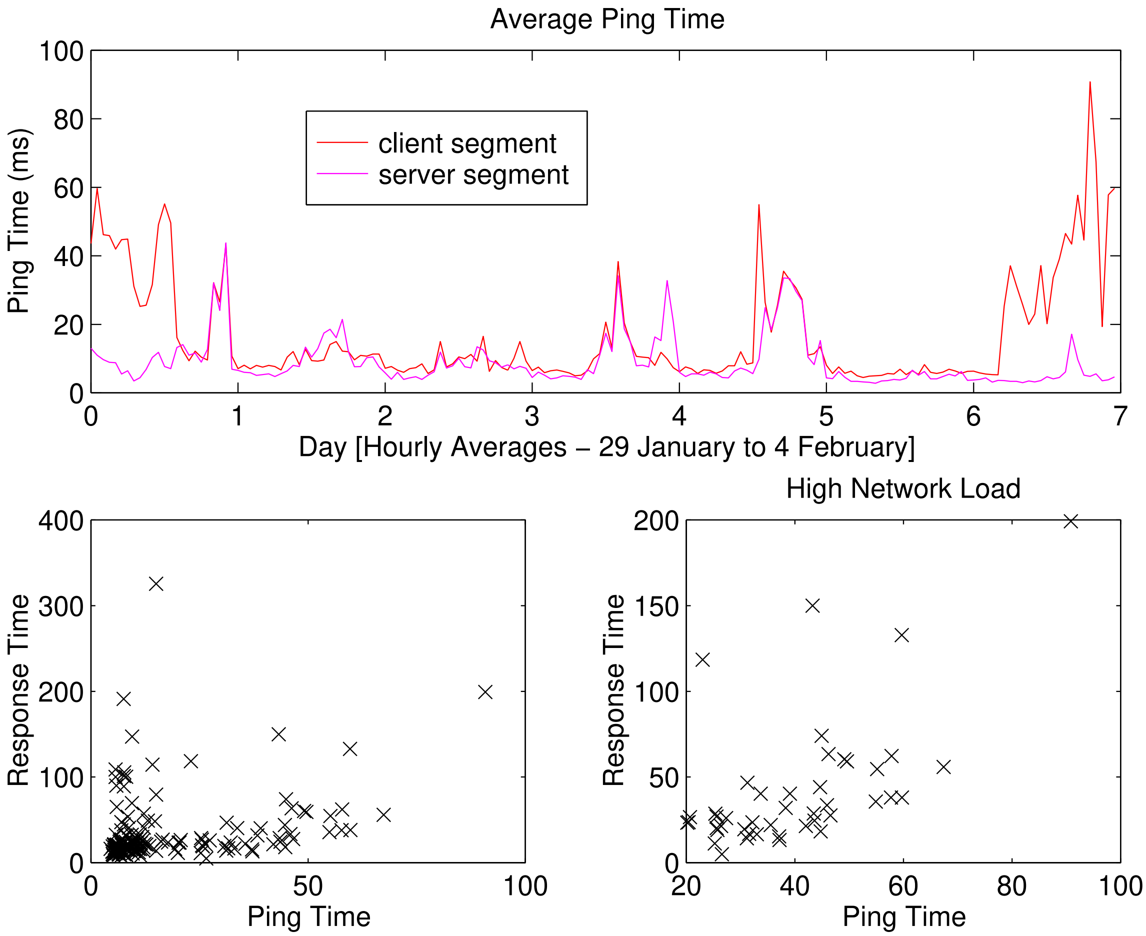


**Figure 5 - Correlation of Response Time with Network Behavior**

### *4.4. Impact of Shared Resources*

Our next step was to again compare server utilization (Figure 3) to average client response time after the periods of high network load are eliminated from the response time trace (see the top chart of Figure 6). Again, we were not able to explain as much of the remaining variance as we expected. Seeking an explanation for this disappointment, we did notice an effect that we had not considered in our initial analysis. Although all of the project volumes for the target group were on the server we were tracing, home directories and shared binaries were being accessed on servers shared across the department.[6] Since we were looking at all FetchStatus operations performed in hour-long intervals, load on these shared servers could have a significant impact on user response time. We see a significant $r^2$ coefficient of 65% between clients of the same system type, suggesting that about 65% of the variation in a single client's response time trace is explained by the variation on the average response time trace of machines of the same system type. At the same time, we see a strong anti-correlation ($r^2$ coefficient of essentially zero) with clients of different system types. The plots at the bottom of Figure

[6] Instrumenting all of the servers and clients in our environment would have been impractical due to the system changes necessary and the sheer volume of users we would have had to persuade to participate.

8 show the correlation between the response time seen at millburn (an RS/6000) and response time at other RS/6000s and, in the right plot, the correlation between millburn and some of the Alpha AXP machines in the study. Not surprisingly in hindsight, our mistake was to overestimate the effectiveness of the replication of commonly used binaries and underestimate the frequency with which users' home directories are used in the course of project work. Although most of the user data may be stored on a fast server, binaries and home directories stored on shared, slow servers may be a considerable drag on user-visible performance.[7]

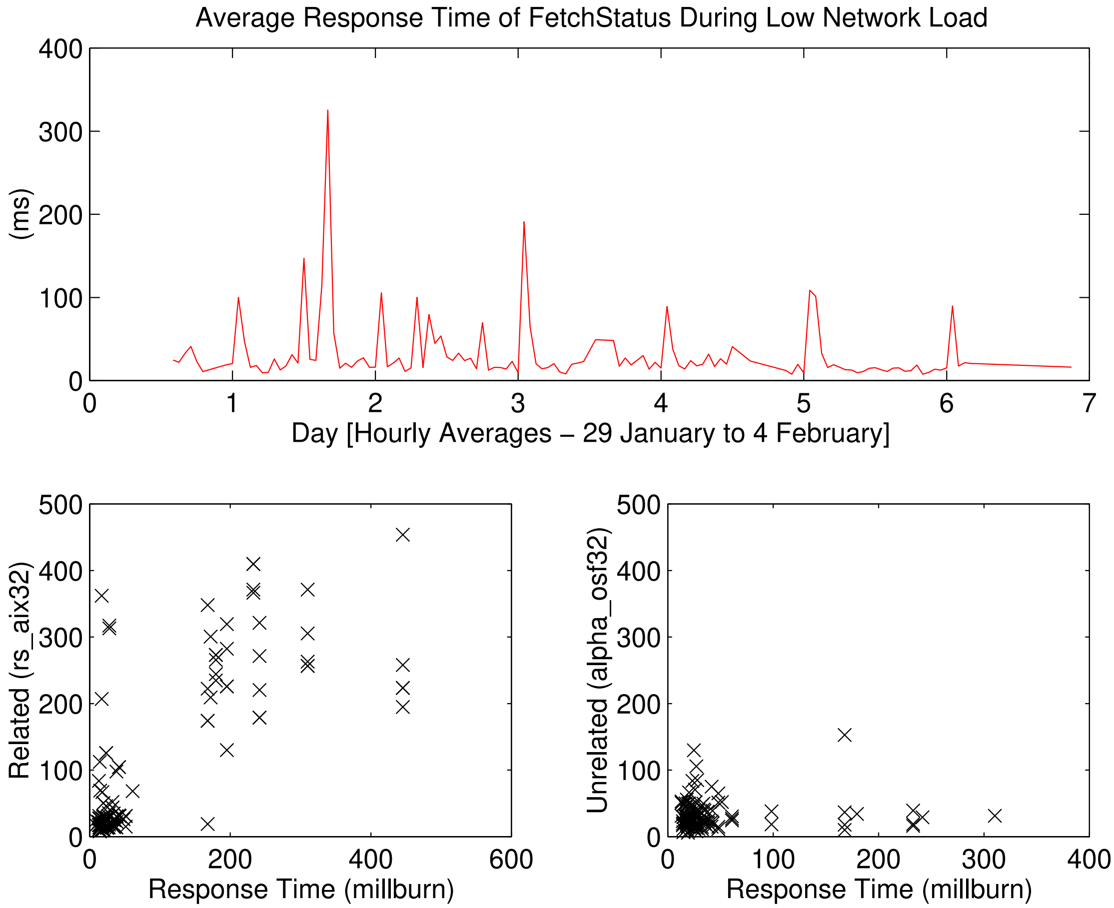


**Figure 6 - Correlation of Response Time by Client System Type**

### 4.5. Impact of Server Utilization

In order to minimize the effect of interaction with servers other than the one we are tracing, we filtered the response time data to include only those periods when a host was active on our server.[8] Figure 7 shows the graphical correlation between average FetchStatus response time and server disk activity and average FetchStatus response time

[7] We will be taking a closer look at this effect and will be placing read-only replication sites of the most-used shared files on our upgraded server to improve our overall performance.

[8] A host was classified as being active on the server in a particular hour if it appeared in the *rxdebug* output at the end of the hour. Since *rxdebug* provides information only for those clients the server has recently interacted with, this does not completely eliminate, but should significantly reduce, the fraction of "foreign" FetchStatus requests in the averages.

and server CPU activity. It is apparent from the leftmost correlation chart of Figure 7 that much of the response time is not correlated with server activity, but as we could not extract the delays associated with central AFS servers, we expect some amount of uncorrelated points. Neglecting data points with less than 500 disk accesses per hour in the center plot, we see an $r^2$ correlation of 25%, as response times are impacted by the amount of disk work (dominated by FetchData operations) the server is already processing when new requests arrive. In the rightmost correlation plot, we see an even closer correlation with CPU utilization (for the same set of points as in the center plot where the disks are busy) which explains just over 50% of the variation in response time. This suggests that poor response times occur when the server CPU and disk are busy (after network and "foreign" server effects have been accounted for). This result fits well with our prior observations that a considerable number of cycles are required to move data from a disk, through the user-level *fileserver* process, back into the kernel, and onto the network, and that these numbers scale with the amount of data being moved [Gibson96].

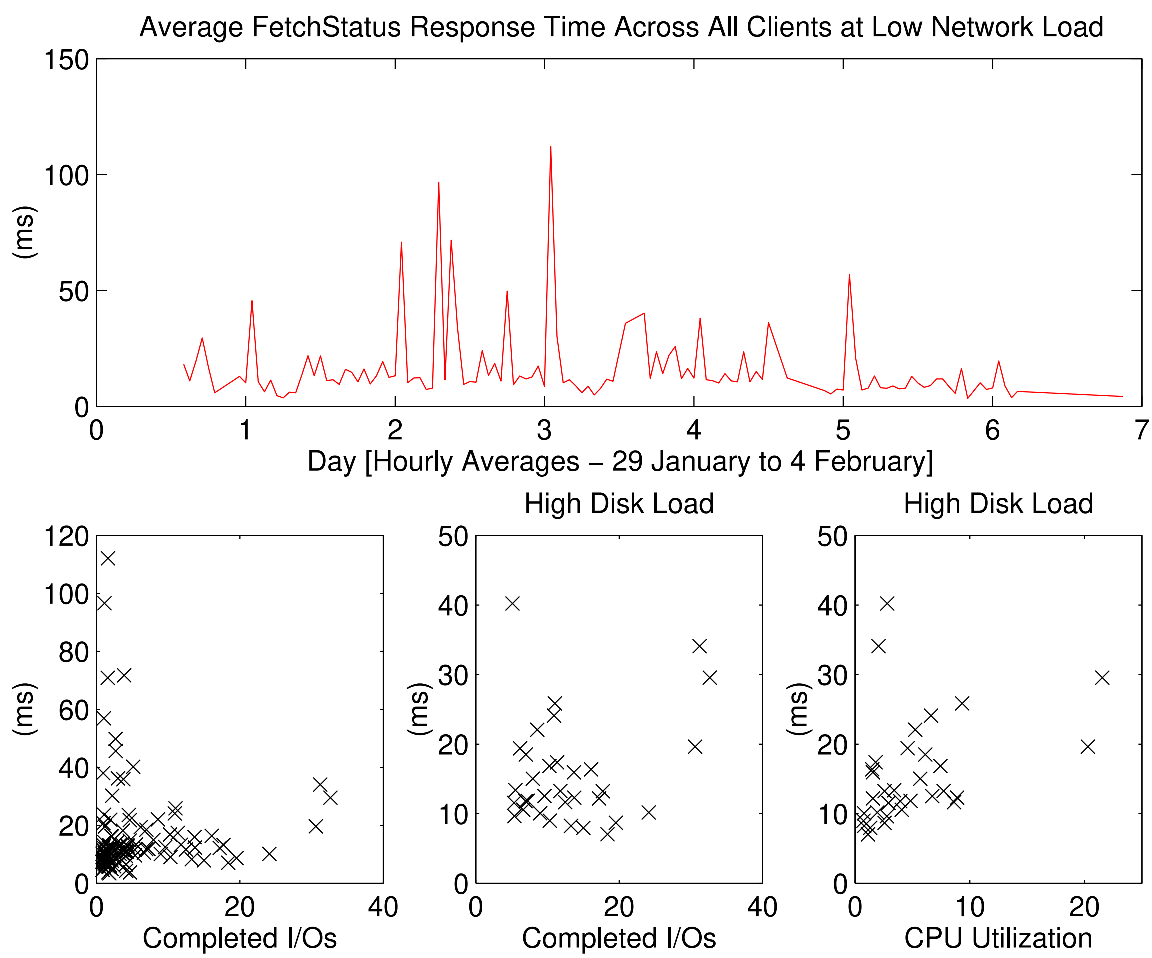


**Figure 7 - Correlation of Response Time with Disk Activity and CPU Utilization**

We have finally discovered the correlation we have been seeking - a faster server CPU benefits AFS users because there are bursts of CPU activity, specifically when data is being transferred, during which server load leads to poor client response times.

# 5. Network-Attached Storage

## *5.1. Opportunities for Network-Attached Storage*

Recalling Figure 2, which shows how the distributed file server machine acts as an intermediary, copying data between the client network and the storage interconnect, we would like to develop techniques for reducing server utilization during periods of intense transfer workloads. In fact, because of the speed, addressability, and distance limitations of SCSI cabling, new storage interconnects such as Fibre Channel are increasingly similar to client network fabrics. With this convergence in mind, we propose that the client and storage networks discussed in Section 1 be combined into a single fabric. As illustrated in Figure 8, this creates the opportunity for disks with sufficient intelligence to perform a significant fraction of the clients' file operations without the need for intervention from the distributed file server [Gibson96].

Eliminating the server machine as a bottleneck for data transfers between storage and applications provides a significant opportunity for improving overall performance. By not involving a third party, common case transfers are considerably faster and the number of requests that can be serviced at any given time should be increased. Data transfer functions are off-loaded to the network-attached devices and the server would be responsible only for "higher-level" distributed file system functionality.

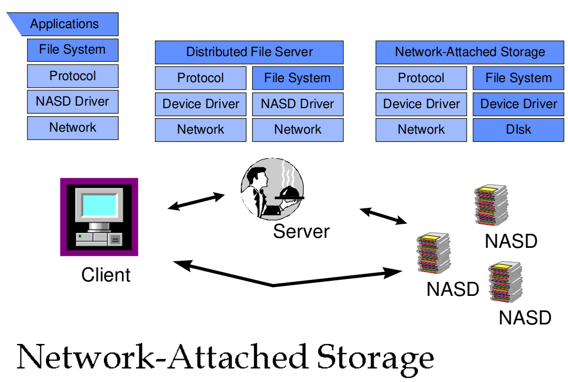


**Figure 8 - Network-Attached Storage Architecture**

There is a range of possible configurations for such a system. At one end of the spectrum, Network SCSI is being promoted by several vendors as a means of providing third-party transfer between clients and drives attached directly to the network [Seagate96]. All commands are processed by a server which uses the SCSI third-party transfer interface to instruct drives to transfer data directly to clients. At the other end of the spectrum, dedicated Network File System (NFS) or Netware servers [NetApp96, NetFrame96] are storage systems that directly implement these distributed file system protocols, backed by specially optimized hardware configurations. Network-attached storage proposes to provide an intermediate point. The distributed file system server would continue to be responsible for operations such a file opens and metadata management, but drives would have sufficient intelligence to handle data transfer requests without server intervention for each individual request. In order to achieve the desired scalability and performance, it may also be necessary to have file status and inquiry functions handled at the drives [Gibson96].

This direct transfer concept is not a new one. In 1991, Randy Katz described the basic advances that make network-attached devices feasible [Katz91]. The High Performance Storage Systems project [Watson95] is exploring these technologies in the context of large MPP and SMP systems based on the framework of the Mass Storage Systems Reference Model [Miller88]. Van Meter provides a survey of current products and major research issues, including security, network protocols, and the changes in operating system paradigms necessary to efficiently support network-attached devices [Van Meter96].

Such an architecture raises several important issues. Can the drive be made sufficiently intelligent at a reasonable cost? How do we ensure the security and integrity of the data being stored? Can enough of the server functionality be off-loaded to significantly improve both throughput and scalability? How effective will this architecture be for meeting the needs of the clients in a distributed system?

### 5.2. Implications of this Study for Network-Attached Storage

The biggest lesson that we take away from the preceding analysis is that the mean behavior of the system is essentially irrelevant. Even though the system is 97% idle when measured in total, it is the high load periods that matter to customer satisfaction. As Table 5 shows, peak loads, even at the granularity of an hour, are much higher than average loads. Moreover, the distribution of operations measured over the long term, shown on the left of Table 5 and similar to previous studies [Spasojevic96] is not preserved in these peak periods - data activity is nearly twice as common in these peaks. With customer satisfaction sensitive to response time variation, the server performance during peak loads is likely to be more important than at other times.

| Server Operations | Weekly Total | | | Peak Hour | |
|---|---|---|---|---|---|
| | total | fraction | hourly | total | fraction |
| FetchStatus | 412,695 | 70.6% | 1,247 | 6,209 | 45.3% |
| StoreStatus | 22,642 | 3.9% | 134 | 175 | 1.3% |
| FetchData | 62,288 | 10.7% | 370 | 4,219 | 30.8% |
| StoreData | 32,414 | 5.5% | 192 | 147 | 1.1% |
| CreateFile | 17,089 | 2.9% | 101 | 52 | 0.4% |
| RemoveFile | 20,422 | 3.5% | 122 | 2,587 | 18.9% |
| GiveUpCallbacks | 17,298 | 3.0% | 103 | 326 | 2.4% |
| total | 584,848 | | 2,269 | 13,715 | |

**Table 5 - Distribution of Server Operations**

Given a high emphasis on the server performance during peak loads, off-loading the high-cost data movement operations, as proposed by the network-attached storage architecture, should decrease the variance in user response time significantly, even though overall averages will simply be reduced from a small number to an even smaller number. The appropriate analogy is not to system throughput, but something closer to the way reliability is measured. Changing the mean time to data loss (MTTDL) of a system from 10 years to 100 years does not mean that one expects the system to last ten times as long, but that the probability of a failure occurring within the next hour is reduced by an order of magnitude. We suggest that there is an analogous measure for distributed file systems, the mean time until burst (bad) performance (MTTBP) which should be increased so that the probability of poor response times in any given hour of work is decreased. We would expect users to be pleased if the occurrence of a period of bad response time were reduced from once a week to once every 3 months.

## 6. Conclusions

Modern distributed file systems such as AFS very successfully cache file data on client machines. While this ensures that average response time is low, it also ensures large variance in response time because operations that must contact remote servers are much slower. Direct measurement of these remote servers show that their overall utilization can be quite low, 3% in our data, while users are simultaneously sufficiently dissatisfied with performance to pay for a faster server. This study shows that the faster server is in fact needed because, although 97% idle overall, these file servers can be intensely overloaded during bursts of activity, leading to periods of poor response time long enough to disgruntle users.

In addition to focusing our attention on burst server loads, our analysis shows that the distribution of operation types during bursts is different from overall distributions. Servers should be optimized for workloads with much more data transfer than the overall distribution suggests.

These results confirm our intuition that network-attached storage, if it can re-route most data transfer directly to storage devices, has the potential to reduce customer response time in two ways - 1) it avoids the copying steps at the server and 2) it off-loads the work of data transfer from the server, reducing the chance of a burst of overutilization.

Our future work, then, is to evaluate the client performance on such network-attached storage architectures and demonstrate the implications on distributed file system design.

## 7. Acknowledgements

We would like to thank Chris Demetriou and Jim Zelenka for their help is collecting the trace data. In addition, this work was complemented by discussions among all the members of our research group, including David Nagle, Khalil Amiri, Fay Chang, Eugene Feinberg, Howard Gobioff, Chen Lee, Berend Ozceri, David Rochberg, and Hugo Patterson.

This research is sponsored by DARPA/ITO through ARPA Order D306, and issued by Indian Head Division, NSWC under contract N00174-96-0002. The project team is indebted to generous contributions from the member companies of the Parallel Data Consortium. At the time of this writing, these companies include Hewlett-Packard Laboratories, Symbios Logic Inc., Data General, IBM Corporation, Seagate Technology, EMC Corporation, Storage Technology Corporation, and Digital Equipment 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.

# 8. References


[Spasojevic96] Spasojevic, M. and Satyanarayanan, M. "An Empirical Study of a Wide-Area Distributed File System" *ACM Transactions on Computer Systems*. May 1996.

[Howard88] Howard, J. et. al. "Scale and Performance in a Distributed File System" *ACM Transactions on Computer Systems*. Volume 6, Number 1. February 1988, pp. 51-81.

[Mummert94] Mummert, L. and Satyanarayanan, M. "Long Term Distributed File Reference Tracing: Implementation and Experience" *Technical Report CMU-CS-94-213*. November 1994.

[Kirk90] Kirk, R. Statistics: An Introduction. Holt, Rinehart and Winston, Inc. 1990, pp. 155-190.

[Ruemmler93] Ruemmler, C. and Wilkes, J. "UNIX disk access patterns" *Proceedings of the USENIX Winter 1995 Technical Conference*. January 1993, pp. 405-420.

[Baker91] Baker, M. et. al. "Measurements of a Distributed File System" *Proceedings of the 13th Symposium on Operating Systems Principles*. October 1991.

[Gibson96] Gibson, G. et. al. "A Case for Network-Attached Secure Disks" *Technical Report CMU-CS-96-142*. June 1996.

[Seagate96] Seagate Corporation "Barricuda Family Product Brief (ST19171)". June 1996.

[NetApp96] "Network Appliance Advantage" *http://www.netapp.com/products*. July 1996.

[Netframe96] "The ClusterServer 8500 Series" *http://www.netframe.com/products*. July 1996.

[Katz91] Katz, R., "High-Performance Network- and Channel-Attached Storage" *Proceedings of the IEEE.* Volume 80. August 1992.

[Watson95] Watson, R. and Coyne, R. "The Parallel I/O Architecture of the High-Performance Storage System (HPSS)" Fourteenth IEEE Symposium on Mass Storage Systems. September 1995, pp. 27-44.

[Miller88] Miller, S. "A Reference Model for Mass Storage Systems" *Advances in Computers*. Volume 27. 1988, pp. 157-210.

[Van Meter96] Van Meter, R. "A Brief Survey of Current Work on Network Attached Peripherals" *Operating Systems Review*. Volume 30, Number 1. January 1996.