An Analysis of File Migration in a Unix Supercomputing Environment

Ethan L. Miller
Randy H. Katz
University of California, Berkeley

ABSTRACT

The supercomputer center at the National Center for Atmospheric Research (NCAR) migrates large numbers of files to and from its mass storage system (MSS) because there is insufficient space to store them on the Cray supercomputer's local disks. This paper presents an analysis of file migration data collected over two years. The analysis shows that requests to the MSS are periodic, with one day and one week periods. Read requests to the MSS account for the majority of the periodicity; as write requests are relatively constant over the course of a week. Additionally, reads show a far greater fluctuation than writes over a day and week since reads are driven by human users while writes are machine-driven.

1 Introduction*

Over the last decade, computers have made incredible gains in speed. This speedup has encouraged the processing of larger and larger amounts of data; however, storing this data on magnetic disk is not feasible. Instead, most data centers with large data sets use tertiary storage devices such as tapes and optical disks to store much of their data. These devices provide a lower cost per megabyte of storage, but they have longer access times than magnetic disk. By studying the tradeoffs between cheaper and slower tertiary storage and more expensive and faster disk storage, response time can be improved without increasing storage costs.

The problem is especially acute at computer centers, such as the National Center for Atmospheric Research (NCAR), that deal with large amounts of data that can never be deleted. Data grows at the rate of several terabytes per year [20]. The cost of storing this data on shelved magnetic tape is relatively low, as cartridge tapes are inexpensive. However, storing even 1% of the total data in magnetic disk would be expensive, requiring hundreds of gigabytes of Cray disk storage.

This paper analyzes file migration behavior on the NCAR system described in [1] and [18]. The first section will provide some background on the problem, discussing current mass storage systems and previous work on them. The next section will describe the NCAR system in more detail. We will then present our trace-gathering methods.

* This work was supported in part by University Corporation for Atmospheric Research contract S9128, and an NSF Fellowship.

The main part of the paper is a two-part analysis of the gathered trace data—analyzing the usage patterns for the entire mass storage system (MSS), and studying the behavior of individual files. The first part of the analysis includes system behavior over the course of a day, week, and longer periods. It characterizes user behavior with respect to the entire MSS, showing at what rate data and files are read and written. Other characteristics of the mass store at NCAR, such as request latency and interrequest distribution, are also discussed. The second part of the analysis provides insight for designing migration algorithms, as it focuses on how individual files are treated. This part of the analysis will discuss file size distribution and individual file reference patterns.

We will finally present some implications of our findings on migration algorithms, and suggest some directions for future research.

2 Background

2.1 History

File migration systems are used by many large computer installations, such as NCAR [1,18] and NASA [7,19], to store more data than what would cost-effectively fit on magnetic disk. Tertiary storage, which usually consists of tape and optical disk, lies at the bottom of the "storage pyramid," as shown in Figure 1. Cost and speed increase going up the pyramid, while the size of the memory level increases towards the bottom of the period. CPU cache is at the top of the pyramid; it have the highest cost per byte and is the smallest and fastest of the levels. At the bottom of the pyramid are tape and optical disk, which have slow access speeds, on the order of seconds or minutes, and very low cost, under $10/GB.

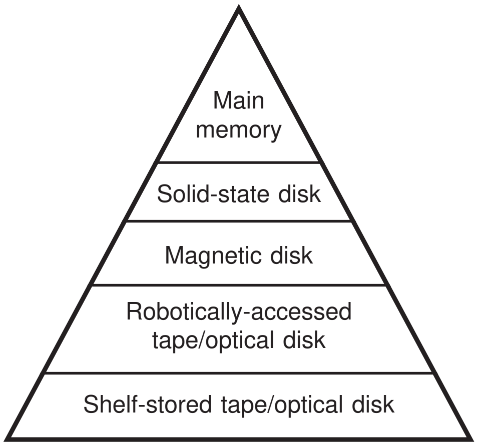

Figure 1. Memory and storage hierarchy in large computer systems. This is also called the "storage pyramid."

Early mass storage systems used manual tape mounting, since it was cheaper to hire system operators than it was to have a robot manage tape mounts. However, by 1978, several companies had introduced automated tape systems [2], and automated tape storage became part of the mass storage systems in most large computer centers. Several of these centers were studied in the early 1980s; these included Brookhaven National Laboratory [5], the University of Illinois [10], and the Stanford Linear Accelerator Center [14,15]. These will be discussed in a later section.

Since these studies, many complex mass storage systems have been implemented, including those at NASA Ames, NCAR, and SDSC [7,11,12]. However, no studies on these systems have been published. Instead, the data management staff at these sites collect huge amounts of data to plan for new equipment purchases and tune their systems. While this guarantees good performance for each system, it does not provide any guidelines for building future systems.

2.2 Mass Storage Devices

Currently, there are two major types of tertiary storage devices in common use—tape and optical disk. Both of these are high-density removable media. The tradeoffs between the two media are presented in Table 1. Two types of magnetic tape, helical scan and longitudinal (linear) scan, are presented. The numbers for the tapes come from [4], while the optical disk statistics come from [16]. Table 1 includes figures for the IBM 3490 and the Ampex D-2.

Currently, the IBM 3480 tape format is standard at most supercomputer installations, though some are beginning to move to helical scan tapes that provide higher density. The IBM 3480 uses linear recording, which provides high speed at the expense of recording density.

Category	Optical Disk Jukebox	Linear Tape	Helical-Scan tape
Media capacity (GB)	1.2	0.4	25
Random access speed	7 sec	13 sec	60+ sec
Transfer rate (MB/sec)	0.25	6.0	15
Media cost/GB	$80	$25	$2

Table 1. A brief comparison of optical disk and tape. The linear tape is an IBM 3490 (high-density version of the 3480), and the helical-scan tape is an Ampex D-2.

The D-2 drive, on the other hand, uses helical scan techniques (similar to conventional VCR recording) to greatly increase recording density. With a new generation of linear tape being developed, however, both types of tape may be close in cost, performance and capacity.

The major tradeoffs among the three media are access latency and transfer bandwidth. Optical disks have a much lower access latency than either type of magnetic tape, but their bandwidth is also considerably lower. Thus, a system which performs many small I/Os to tertiary storage, such as a database system, would be best served by optical disk, since the dominating factor in calculating time per byte is access time to the first byte. For supercomputing installations, however, magnetic tape is better. While the time to get the first byte of data is longer for tape than for optical disk, the time to get *all* of the data is often lower for tape. Files on supercomputing installations tend to be large [20], so the difference in transfer time between optical disk and tape is substantial. In general, more expensive drives have higher transfer rate and storage density, though neither longitudinal scan nor helical scan seems intrinsically better. A new technology, optical tape [16], also looks promising because of its high density storage and high transfer rate.

Another primary consideration is price per gigabyte. As can be seen in Table 1, magnetic tape has a lower cost per gigabyte stored than optical disk. For systems with terabytes of data stored on tertiary storage, such as NCAR, this cost difference alone is enough to favor using tape exclusively as the tertiary store. The lower cost and higher transfer rate make magnetic tape the obvious choice for supercomputer centers which deal with sequentially-read large files.

Most installations today have one or more cartridge tape robots to automatically mount some of their tape libraries. The StorageTek 4400 [9] is an example of a tape robot, or automated cartridge system (ACS). This system can provide access to 1.2 TB of data (6000 IBM 3480 cartridges, holding 200 MB each). Loading a cartridge takes approximately 6 seconds; from there, tape characteristics are those of the IBM 3480 tape drives in the tape silo.

2.3 Previous Work

There have been several studies of actual file migration systems, but most are quite old and deal with different computing environments. We will summarize them here, and in a later section will compare the results of studying the current NCAR environment with the results of the earlier studies.

In [15] and [14], Smith studied the file system at the Stanford Linear Accelerator Center. His data dealt with Wylbur text editor data sets, and tracked the references to those data sets. He found that the best algorithms had access to the entire reference string for a file. Since this is often not feasible, the criterion he suggested was to migrate off disk the files with the highest value of *last reference time*$^{1.4}$ $\times$ *file size*. This algorithm, called Space-Time Product (STP**1.4), was the best of the algorithms examined which did not make use of any file history other than the last reference time. The analysis in the paper also did not consider the possible effects of transfer time and access latency in minimizing average file reference time; instead, the analysis attempted to minimize file miss rate.

Smith also made several observations about file system activity. He noted that usage followed a weekly pattern, with activity highest on weekdays and lower on weekends and holidays. He also has extensive data on file sizes and interreference intervals. Because of the size of the data set in our NCAR study (over 900,000 files), it would be very difficult to perform the same computations over the entire file set. The data set in Smith's paper has a granularity of one day and does not distinguish between reads and writes.

None of the acceptable migration algorithms would have had much effect on average file access time at NCAR. As noted in the paper, a miss ratio of 1% would mean a loss of 6.26 person-minutes per day, given the file usage rates and the number of users on the system. For STP, this miss ratio would require a disk system that held 1.5% of the total tertiary storage, and would require 300 tracks, or about 1 MB, of data to be transferred each day.

Lawrie, et. al., in [10], considered the file migration patterns on the University of Illinois Cyber 175. Again, the system examined is quite different from the one studied in this paper. Interestingly, Lawrie reported that, though his system was quite different from SLAC, his results matched Smith's closely. This paper also examined several migration algorithms, and compared them against Smith's STP algorithm on their data. They found that STP was better than the algorithms they tried, which included pure LRU, pure length (migrate large files first), and SAAC, which migrated files that became less active. In all cases, STP outperformed these algorithms, though only by a slim margin.

Two recent studies focused on a workstation file system at Berkeley [17] and the Common File System (CFS) at the National Center for Supercomputing Applications (NCSA) [8]. At Berkeley, Strange found that there were more file reads than file writes, though more data was written than read. He also found that, as expected, less data was used on weekends (even though the system was primarily used by graduate students). In this system, algorithms using a space-time product to identify files to migrate would work well. However, files were much smaller than typical supercomputer files. Even the file system with the largest files averaged under 50 KB/file. As Table 4 shows, this is far smaller than typical supercomputer files. The file system profile at NCSA, on the other hand, is quite similar to that at NCAR. File sizes are similar, and file reference rates are close to those in this study. This gives us high confidence that NCAR is typical of supercomputer mass storage systems.

Other papers have simply presented data gathered from existing mass storage systems without analyzing the data and suggesting possible algorithm changes. Systems analyzed include Brookhaven [5], NCAR [1,18], and NASA [7]. In addition, many large sites internally publish a summary of statistics gathered from their machines. They use these statistics for two purposes: to better tune their systems, and to justify new equipment purchases.

3 NCAR system configuration

In this section, we describe the system on which we gathered the file migration traces. Rather than describe the entire NCAR network, we focus on the parts which are relevant to the study. However, the rest of the network will be briefly described, since the mass storage system is shared by all of the systems at NCAR, so their presence might affect mass storage systems performance.

3.1 Hardware Configuration

The CPU in the study was a Cray Y-MP 8/864 (`shavano.ucar.edu`), with 8 CPUs and 64 MWords* of main memory. Each CPU has a 6 ns cycle time. `Shavano`, like other Cray Y-MPs, has several 100 MB/sec connections to its local disks and two 1 GB/sec connections to a solid state disk (SSD). There are about 56 GB of disks attached directly to the Cray; 47 GB of this space is reserved for application scratch space and files over a few days old are purged from it regularly.

The mass storage system (MSS) at NCAR is composed of an IBM 3090—used as a bitfile† server—with 100 GB of online disk on IBM 3380s, a StorageTek

* Each Cray word is 8 bytes long.

Automated Cartridge System 4400 with 6000 200 MB IBM 3480-style cartridges, and approximately 25 TB of data in shelved tape. The MSS tries to keep all files under 30 MB on the 3090 disks, and immediately sends all files over 30 MB to tape. Usually, the tapes written are those in the cartridge silo. Files on the MSS are limited to 200 MB in length, since a file cannot span multiple tapes. While the Cray supports much larger files on its local disks, they must be broken up before they can be written to the MSS.

The MSS at NCAR is shared by the entire NCAR computing environment, which includes the Cray Y-MP, an IBM 3090 which runs the MSS, several VAXen, and many workstations. Figure 2 shows the network connections between the various machines at NCAR. The disks and tape drives attached to the MSS processor have direct connections to the Crays via the Local Data Network (LDN), providing a high-speed data path. All machines connected to the MSS (including the Crays) are connected to the 3090 by a custom hyperchannel-based network called the MASnet. Data going out over the MASnet must pass through the 3090's main memory, so it is a slower path than the direct connection the Crays have. The few workstations with connections act as gateways to the networks which connect to the rest of the workstations at NCAR. These gateways are also the fileservers for the local networks. Many of these smaller machines have their own local lower-speed disks, about 5.5 GB of which are mounted by the Cray via NFS (Network File System). According to the monthly report published by the NCAR systems group [20], `shavano` puts more data on the network than any other node, but several other nodes receive more data. In particular, several of the Sun workstations receive comparable amounts of data. It is likely that these workstations, which are the gateways to internal networks of desktop workstations, are receiving a large amount of image traffic.

3.2 System Software

The Cray Y-MP is primarily used for climate simulations— both the extensive number crunching necessary to generate the data, and the less computationally-intensive processing used in visualizing it. The Cray has two primary modes of operation; it can either run in primarily interactive mode, where programs are short and run as the user requests them, or in batch mode, where jobs are queued up and run when space and CPU time are available. There is no explicit switch between operating modes, but short interactive jobs typically have higher priority. There is less CPU time for running batch jobs during the day, because scientists are looking at results from previous batch jobs. At night, however, the CPU is mainly used to run large jobs requiring hours of CPU

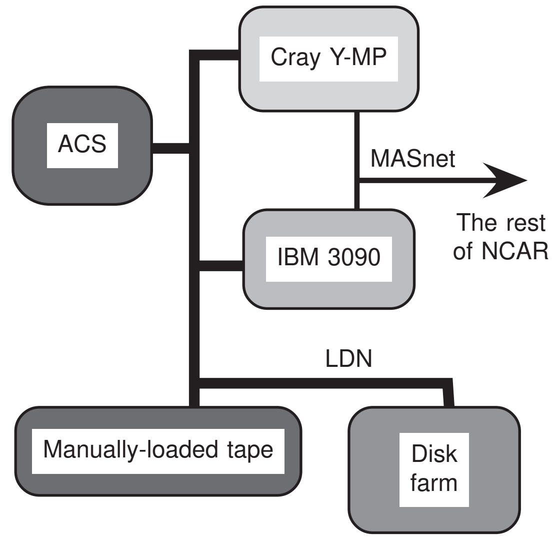

Figure 2. Network connections between machines at NCAR.

time. The MSS request patterns reflect these two different uses of the CPU, as will be shown below.

The software which runs the MSS is based on concepts in the Mass Storage Systems Reference Model [3]. It consists of software on the mass storage control processor (MSCP), which is the IBM 3090, and one or more bitfile mover processes on the Cray. Users on the Cray make explicit requests (via the UNICOS commands `lread` and `lwrite`) to read or write the MSS. These commands send messages to the MSCP, which locates the file and arranges for any necessary media mounts. The MSCP then configures the devices to transfer directly to the Cray. For disk and tape silo requests, these mounts are handled without operator intervention, but an operator must intervene to mount any non-silo tapes which are requested. After the data is ready to be transferred, the MSCP sends a message to a bitfile mover, which manages the actual data movement. When transfer is complete, the bitfile mover returns a completion status to the user.

3.3 Applications

The Cray at NCAR runs two types of jobs—interactive jobs, which finish quickly and require a short turnaround time, and batch jobs, which may require hours of CPU time but have no specific response time requirements.

A typical climate simulation, such as the Community Climate Model [21], might take 1 hour and produce 500 MB of data which would be stored on a tertiary store. This is an example of a batch job, since a researcher would submit the job and allow it to run overnight or longer. These jobs use a large amount of temporary disk storage as well as CPU time. The Y-MP at NCAR is configured with small, 300 MB user parti-

† A bitfile is a stream of bits stored by the file system. It is the same as a plain file in UNIX.

tions. Each user is allocated a few megabytes on one partition, which would be insufficient for storing the output of even one run of a climate model. Thus, the initial input to a climate model must come from the MSS, and any results must go back to the MSS. If the results are needed later, they must be retrieved from the MSS.

Interactive jobs, such as a “movie” of the results of a climate simulation, have much more stringent response time requirements. Typically, a user will initiate a command and expect a response quickly. According to [19], an interactive request must be satisfied in just a few seconds, or interactive behavior is lost. Nevertheless, the average response time to satisfy MSS requests is over 60 seconds; possible solutions to this problem will be discussed later.

4 Tracing Methods

4.1 Trace Collection

The data used in this study was gathered from system logs generated by the mass storage controller process and the bitfile mover processes. Approximately 50 MB of data was written to these logs per month. The system managers at NCAR use the data to plan future equipment acquisitions and improve performance on the current system. The logs also serve as proof that a requested transaction took place. The system managers occasionally use them to refute users who claim their files were written to the MSS and then disappeared.

The system log, as written by the mass storage management processes, contains a wealth of information. Much of it is either redundant or unnecessary for migration tracing. Information such as project number and user name are not needed for migration studies, since the user identifier is also reported. The traces are designed to be easily human-readable, so fields are always identified and dates and times are in human-readable form. In addition, each MSS request is assigned a sequence number, since there are several records in the system log which correspond to the same I/O. This is useful for assembling a single record for a migration trace. By processing the traces to remove redundant information and transforming the rest of the information into a form more easily machine-readable, the traces were cut from 50 MB per month to 10-11 MB per month. They could not be reduced further because file names are long and could not be compressed without losing information.

4.2 Trace Format

Once the system logs were copied to a local host, they were processed into a trace in a format that is easy for a trace simulator or analysis program to read. The traces were kept in ASCII text so they would be easy to read on different machines with different byte orderings. A list of the fields in the trace is in Table 2.

Field	Meaning
source	Device the data came from
destination	Device the data is going to
flags	Read/write, error information, compression information
start time	time in seconds since the previous start time
startup latency	time in seconds to start the transfer
transfer time	time in milliseconds to transfer the data
file size	file size in bytes
MSS file name	file name on the MSS
local file name	file name on the computer
user ID	user who made the request

Table 2. Information in a single trace record.

Very little information is common between two consecutive records except temporal information. Even so, the traces are compressed by recording times as differences from some previous time, as suggested in [13]. The start time for a MSS request is recorded as the elapsed time since the start time of the previous request, while the latency until the first byte is transferred (the startup latency) and the transfer time are recorded as durations. Start time and startup latency are measured in seconds, while transfer time is measured in milliseconds. These were the precisions available from the original system logs. The only other commonality between consecutive requests might be the requesting user, so there is a bit in the flag field which indicates that the request was made by the same user who made the previous request. Directories, too, might be common between consecutive requests, but they would be harder to match. Future versions of the trace format may allow for full or partial paths to be obtained from previous records.

5 Observations

The traces for this study were collected over a period of 24 months, from October, 1990 through September, 1992. Traces were available from the time the MSS came on-line in June, 1990, but the MSS was very lightly used for the first few months. We decided to omit this data and study the “steady-state” system.

5.1 Trace Statistics

Overall statistics for the trace period are shown in Table 3. The traces actually include 3,688,817 references, but 175,633 (4.76%) had errors. The most common error was the non-existence of a requested file. In such cases, it was impossible to include the reference in our analysis, since the file never existed and wasn’t stored on any device. It might have been possible to include references that encountered other errors, such as media errors and premature termination, but there were few enough that they would not affect the results.

	Reads	Writes	Total
References	2336747. (66%)	1179047. (33%)	3515794.(100%)
Disk	1419280. (60%)	927722. (39%)	2347002. (66%)
Tape (silo)	480545. (66%)	239162. (33%)	719707. (20%)
Tape (manual)	436922. (97%)	12163. (2%)	449085. (12%)
GB transferred	63926.2 (73%)	23389.9 (27%)	87316.2(100%)
Disk	5080.4 (58%)	3727.9 (42%)	8808.3 (10%)
Tape (silo)	38256.6 (67%)	19081.4 (33%)	57338.1 (66%)
Tape (manual)	20589.2 (97%)	580.6 (3%)	21169.8 (24%)
Avg. file size (MB)	27.36	19.84	24.84
Disk	3.58	4.02	3.75
Tape (silo)	79.61	79.78	79.67
Tape (manual)	47.12	47.74	47.14
Secs to first byte	98.1	38.6	78.18
Disk	32.47	25.39	29.67
Tape (silo)	115.14	81.86	104.08
Tape (manual)	292.58	203.84	290.18

Table 3. Overall trace statistics. The trace covers the period from October, 1990 through September, 1992. The percentages listed under "Reads" and "Writes" are ratios to the value in the "Total" column of that row. The percentages listed under "Total" are percentages relative to the top value in the column.

Table 4 contains data about the massive store that accesses were made to. This table only includes files which were referenced during the trace period. Since we had no data on the actual contents of the MSS, we assumed that only files actually referenced during the trace period existed on the mass store. This is a valid simplification, as there are only three kinds of files that are never explicitly read or written—large temporary files used by Cray applications, small files that fit into the 1 MB allocated for a each user's home directory, and system files such as binaries. The first category, temporary files, would be actively used for their entire lifetime, and discarded when no longer in use, never providing a chance to move them to long-term storage. Small user files, such as `.cshrc`, would never be migrated since they would be used too often. Even if they were migrated, they would only add approximately 4 GB of space to the MSS, assuming each of the 4,000 users filled their entire permanent allocation. System files, likewise, would probably be used often enough so they would not be evicted from disk. Additionally, most system files are read-only, eliminating the need to write any data to the MSS.

5.1.1 Latency to first byte

Figure 3 shows the total latency from when a request is made to the MSS until the data transfer actually starts. This time is composed of several elements—queueing time on the Cray, queueing time on the MSS, media mounting time, and seek time. For the disk, media mounting time and seek time are very short, usually well under a second. While median access time for the disk was 4 seconds, the distribution has a long tail due to queueing at individual disks. Each disk has a relatively low bandwidth, so a large file takes several seconds to satisfy. Any requests for this disk that arrive in the meantime must wait for the long request to finish, generating the long delays in the tail of the disk latency curve.

Number of files	902772
Average file size	25 MB
Number of directories	143245
Largest directory	24926 files
Maximum directory depth	12
Total data in MSS	23 TB

Table 4. Statistics for a file store needed to satisfy all of the traced accesses

Delays were caused by queueing in several places in the system—the Cray, the MSS CPU, the network from disk to Cray, and data transfer to or from the disk itself. Of these, the only delays that differ between disk, tape silo, and shelved tape are the latencies due to the device itself—transfer delays and seek delays. The disks do not transfer data much faster than the tape drives, so queueing delays for them are probably representative of the time spent waiting for data to be transferred off tape. We can then deduce how much extra time is needed by the tape systems to get the first byte of data.

The first observation is that the tape silo is considerably faster than manually fetching the tape. After subtracting off the queueing time exhibited by the disk, the silo is approximately 2 to 2.5 times as fast as the manual tape drives at getting to the first byte. Since the tape silo tape drives are the same as the operator-loaded tape drives, this difference must come from the time to mount the tape rather than from seek time. The StorageTek 4400 ACS can pick and mount a tape in under 10 seconds; after subtracting off average queueing time for the disk, which is 25 seconds, the non-seek overhead for reading an automatically-loaded tape is 35 seconds. According to Table 3, tape accesses take 85 seconds on average, so the average seek is 50 seconds long. When the same analysis is applied to manually loaded tapes, the manual tape mounting time is found to be approximately 115 seconds, or about 2 minutes. This is quite good. However, as Figure 3 shows, 10% of all manual tape mounts were not completed within 400 seconds. Nearly all of the tape silo and disk requests were completed by this time. This is probably the biggest weakness of manual tape mounting—the very long tail of the mounting time distribution. While other data accesses will almost certainly complete in 5 minutes, manual tape mounts may take much longer.

This is just a simple analysis, though. There are several factors that we did not consider which may affect our conclusions here. In particular, queueing time for the tape silo may be different from queueing time for the disks. There are only a few tape robots in the silo, and each is tied up for several seconds with a tape load. If several tape loads come in close together, some of them will have relatively long queueing times. This does not happen with disk, as each disk is tied up for relatively little time with each request.

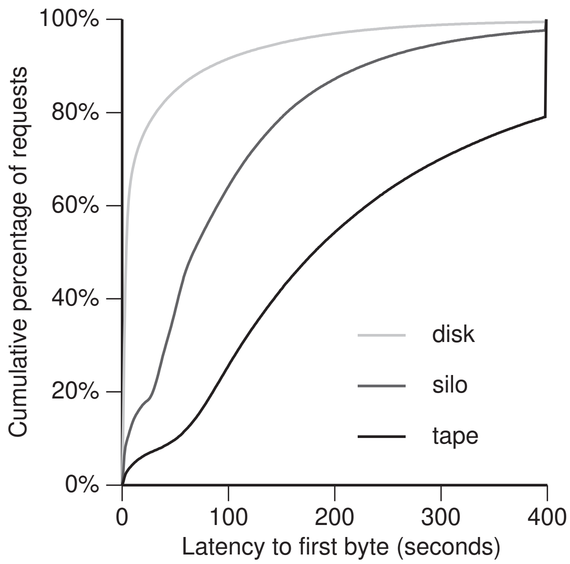

Figure 3. Latency to the first byte for various MSS devices.

Another observation is the relation between latency to access the first byte and time required for the entire transfer. Both the tapes and the disks can transfer at a peak rate of 3 MB/sec, but the observed rates are usually closer to 2 MB/sec. As a result, the transfer times are similar for the two media. For tape, an average file of 80 MB will take 40 seconds to transfer. This is comparable to the additional 60 second overhead from using tape instead of disk. One possible way to improve perceived response time in the system would be to use cut-through, as in [7]. Under this scheme, a call to open a file returns immediately, while the operating system continues to load the file from the MSS and keep track of how far it has gotten. When future requests are made, the call returns immediately unless the requested data has not yet been read. This scheme works because applications often do not read data as fast as the MSS can deliver it. Instead of delaying the application, then, it allows the application and file retrieval from the MSS to overlap. This system would be difficult to use in the current NCAR configuration, however, since the MSS is not seamlessly integrated with the local disk file system. The bitfile mover processes would have to have special communication protocols with the local file system to let it know how much of the file has been transferred. Nevertheless, it is a useful optimization and should be considered.

5.2 MSS usage patterns

Figure 4 shows the average amount of data transferred each hour of the day. As expected, activity is highest during working hours—from 9 AM to 5 PM. The variation in transfer rate, however, is almost entirely due to reads. The amount of data read jumps greatly at 8 AM when the scientists usually arrive, and slowly tails off after 4 PM as they leave. The fall is slower than the rise because most scientists are more likely to stay late than to arrive early. This suggests that most reads on the system are initiated by interactive requests, since reads peak when people are at work, while writes remain almost constant regardless of the number of humans requesting data. File request rate over the course of a day shows a pattern similar to that of data transfer rate..

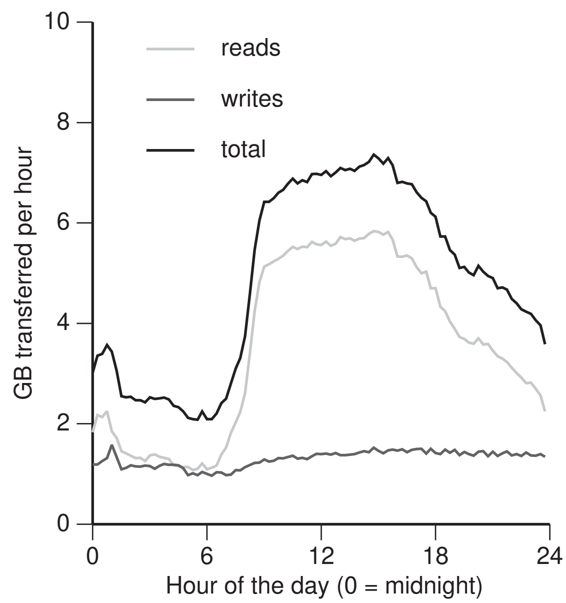

Figure 4. Average data transfer rate over the course of a day.

The weekly data transfer rates, shown in Figure 5, have patterns similar to those in the daily averages. As expected, read activity is lower on the weekends, since there are fewer researchers around to initiate read requests. Write requests, on the other hand, experience little variation over the course of the week, as the Cray CPU runs batch jobs all weekend. There is a small increase in write requests during the day, indicating that users do actually make some write requests; however, the change is small relative to the flood of read requests that users generate.

Note that less data is transferred early Monday morning than on any other day. This low point can be attributed to two factors. First, the Cray might be taken down early on Monday morning for maintenance, as that would cause the least disruption of normal work. Second, any idle time the Cray might have would be on

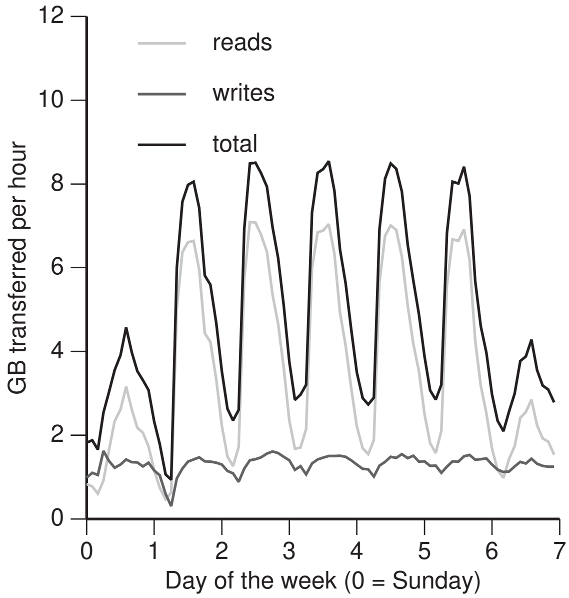

Figure 5. Average data transfer rate over the course of a week.

Monday morning, as the queues from the weekend might have finished.

Over the two years the trace covers, the mass storage system received increasingly large amounts of work. The average data rate for each of the 104 weeks is shown in Figure 6. There are drops in read request rate around Thanksgiving and Christmas for both 1990 and 1991. Note, however, that write request rate does not drop on these holidays. In fact, write requests *increased* at the end of the year. This reinforces our conclusion that reads are interactive while writes are requested primarily by batch jobs, as the Cray doesn't take a Christmas vacation while the scientists do.

The MSS data request rate increases over the period shown by the graph, but this gain is due almost entirely to increases in read requests. MSS write rate appears to be related to the speed with which the computer can generate results, while read rate is set by the number of users that want to read their data back. The lack of increase in write rate suggests that the Cray is already running at full capacity, and that researchers are simply using the machine more for tasks such as visualization of the results. A faster machine would then need a higher write rate to massive storage. There would be at least a corresponding increase in read rate, and it might be greater if the user community gets larger.

5.2.1 Interreference intervals

Figure 7 shows the distribution of intervals between references to the MSS. Since about 3,500,000 files were referenced over a period of 731 days (approximately 6.3×10^7 seconds), the average interval between MSS requests was 18 seconds.

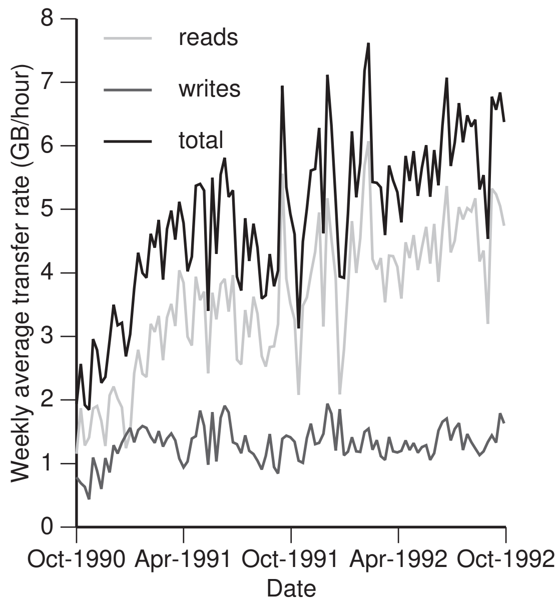

Figure 6. Average data transfer rate over the course of a week.

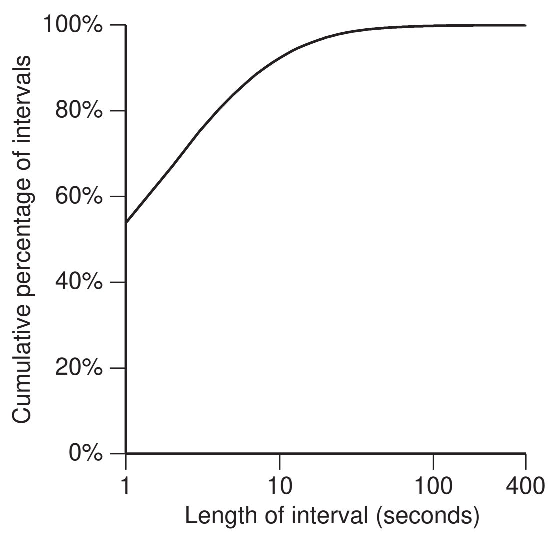

Figure 7. Lengths of intervals between Cray references to the MSS.

Looking at the graph, however, shows that 90% of all references followed another by less than 10 seconds. This distribution suggests that I/Os are clustered. There are several possible explanations for this. Since Cray files can be of (nearly) unlimited length, but files on the MSS cannot exceed 200 MB, clustering could occur since several files are accessed together by the same program. Another possibility is that there are really two distributions for intervals—those made by researchers' interactive requests, and those made by batch jobs. The interactive requests are very likely to be bunched together, since a researcher interested in day 1 of a climate model simulation will usually be interested in day 2, and both days will probably be in separate files.

5.3 File reference patterns

Instead of counting all file references, this part of the analysis included at most one read and one write from any eight hour period. Since files on the MSS were explicitly referenced Unix command, some files were accessed many times in a short time. In a system with automatic migration, this would not be likely to happen.

As expected, most files were not referenced often. Figure 8 shows that only 5% of all files are referenced more than ten times. 50% of the files in the trace were never read at all, and another 25% were read only once. Writes were slightly different—just over 20% of the files were not written during the trace period, but another 65% were written exactly once. Of course, these numbers add up to more than 100%, as many files were read and written one time or less. In all, 57% of the files were accessed exactly once, and 19% were accessed exactly twice. Thus, only a quarter of the files were accessed more than two times. Our observations found that the median number of file references was one, as opposed to [14], which reported the median to be two. Furthermore, fully 44% of all the files in the trace were written exactly once and never read. These numbers confirm the common belief that many files are written to a massive store once and never read again.

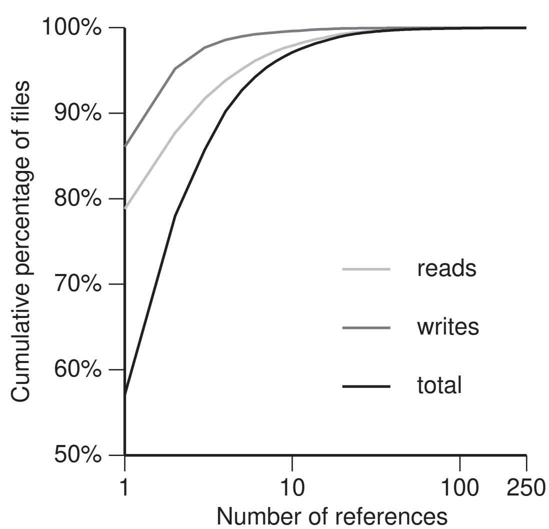

Figure 8. Distribution of file reference counts. During the trace period, 50% of the files had 0 reads and 21% had 0 writes.

Figure 9 shows the distribution of time intervals between references to a given file, called *interreference intervals.* Long interreference intervals mean that a file is referenced infrequently, while short intervals indicate many accesses over a short period of time. As Figure 9 shows, interreference intervals were short. This means that, for files which were rereferenced, the second access came soon after the first. Note, however, that there were still some files that were referenced more than a year after the previous reference to them. These references could not be easily predicted, so it is not sufficient merely to use prediction to improve access time; we must decrease the latency for random requests as well.

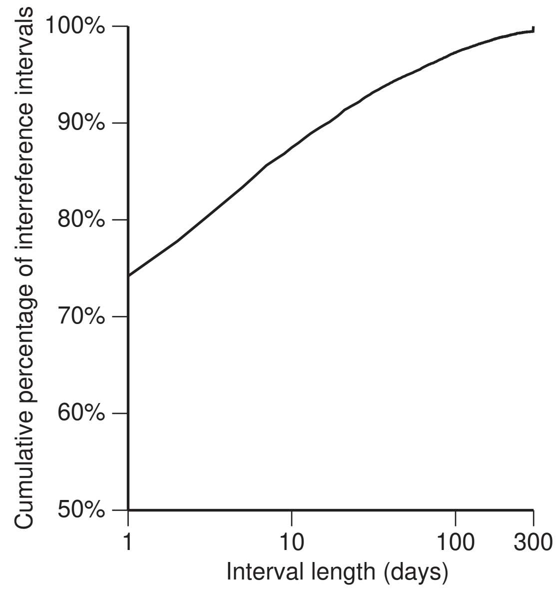

Figure 9. Distribution of intervals between successive references to the same file. 70% of all intervals were less than 1 day.

5.4 File and directory sizes

The dynamic distribution of file sizes transferred between the MSS and the Cray is shown in Figure 10. In this graph, a file is counted once for each access to it. The distributions of files read and files written are similar, though there is a small jump in file writes at approximately 8 MB. However, 40% of all requests are for files 1 MB or smaller. Since reads are more likely than writes to be initiated by a human user (as Section 5.2 shows), this graph suggests that performance on small file reads in a migration system would be especially important. Such small files make up under 1% of the total data storage requirement, so it seems wise to store these files on inexpensive, low-performance disks rather than on tape. If magnetic disk would be too expensive, an optical disk jukebox could provide low latency to the first byte and high capacity.

The distribution of file sizes on the MSS during the trace period is graphed in Figure 11. In it, each referenced file is counted exactly once, regardless of the number of times it was accessed. The graph shows that, while about half of the files are under 3 MB, these files contain 2% of the data. Algorithms that take file size as an argument could use this fact to simplify their bookkeeping, as all files below a threshold size could be

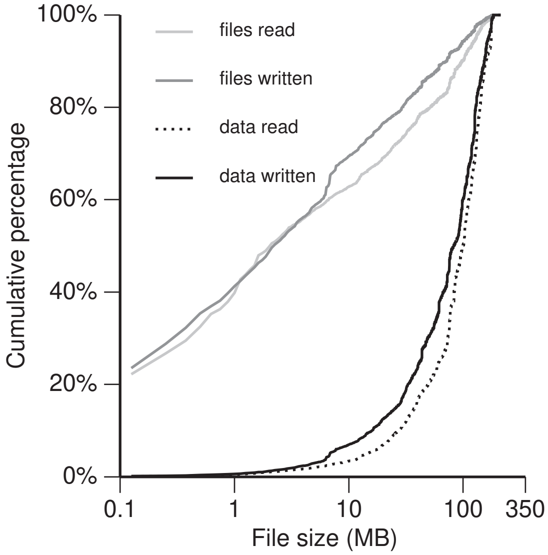

Figure 10. Size distribution of files transferred between the MSS and the Cray. A file is counted once for each time it is requested.

considered equivalent when computing space-time products. Since most files are below this size, the algorithm should run much faster.

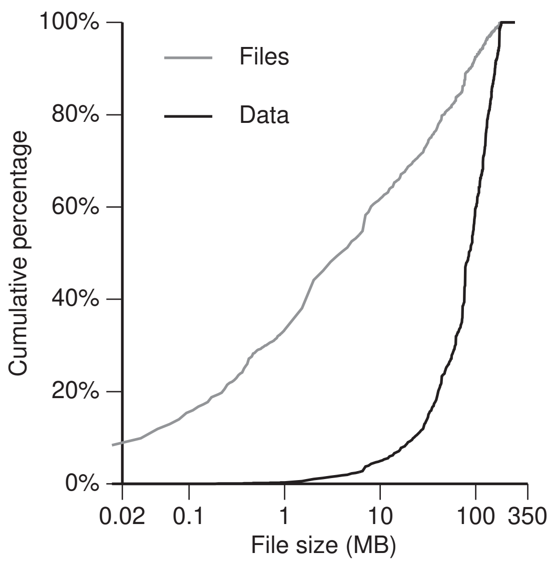

Figure 11. Distribution of file sizes on the MSS. Each file referenced in the trace is counted once.

Directories also tended to be small. Figure 12 shows that 90% of the directories had 10 or fewer files, and 75% had only zero or one file. Even so, over half of all files and data were in large directories that contained more than 100 files. The size and number of directories is very important, as many current systems do not archive directories or file metadata such as inodes. With over 130,000 directories and 900,000 files, the NCAR system needs to store gigabytes of metadata on disk. Future systems must be able to move this information to tape, especially since over 40% of the metadata describes files that will never be accessed again.

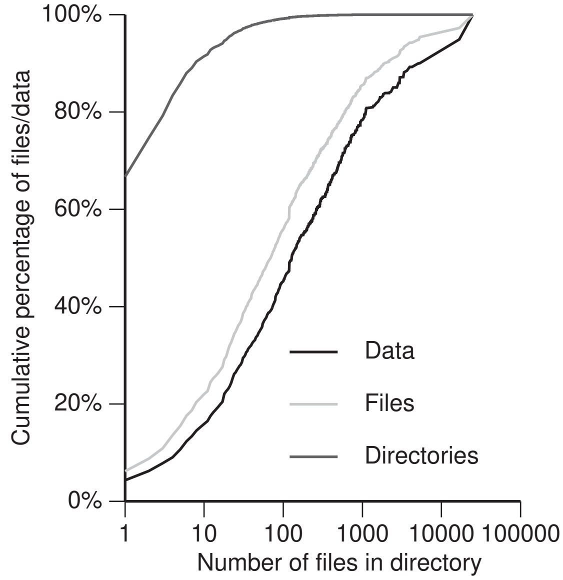

Figure 12. Distribution of directory sizes on the MSS. Note that more than half of the directories had only zero or one file in them (though most of these also had subdirectories). Note also that 5% of the directories held 50% of the files and data.

6 File migration algorithms

The observations made from the NCAR trace data have several implications for future file migration algorithms. The system studied here is quite different from that in the studies done around 1980 [6,15]; while file access patterns are not radically different, the files themselves are larger and there are more of them.

The NCAR system uses two different migration algorithms — one for moving files between the Cray and the MSS, and the other for relocating files on different media within the MSS. Moving files between the Cray and the MSS is entirely manual, so there is no choice in the "algorithm" involved. However, using automatic migration between the Cray and the MSS would still save many file requests. About one third of all requests came within eight hours of another request for the same file. Often, these accesses are generated by batch job scripts which must read or write files on the MSS. If several of these scripts are run at about the same time, the Cray must make a separate request to the MSS for each script; it has no way of keeping track of multiple references to the same file. Better integration of the MSS with the Cray would fix this problem.

Another change since 1980 involves large files. Previous algorithms optimize for low seek time and ignore transfer time. For multi-megabyte files, transfer time dominates the time needed to access a file. On magnetic disk, seek time is far lower than transfer time for megabyte-sized files. Even for robotic tape, however, seek time is comparable to transfer time. A StorageTek robot can load a 3480 tape in under 10 seconds; the drive can transfer 20 MB in this time. The standard algorithms all make the assumption that the retrieval cost is the same for all files (though the storage cost may not be). New algorithms will have to take the difference in access time into account. The NCAR system already does this by storing smaller files on magnetic disk and larger files only on tape. In this way, small files do not suffer the latency penalties of tape. Large files, on the other hand, must wait for a tape to be loaded. However, their transfer time is long enough that the added delay of loading a tape is not as noticeable. The dividing point between storing files on disk and storing them on tape is a subject for future research; however, it is likely that the switchover point will be a function of tape seek speed and transfer rate.

Previous algorithms also make little distinction between reads and writes, primarily because their trace-gathering methods did not allow them to distinguish a read access from a write access. However, this difference is crucial for a file migration algorithm. The read/write ratio to the MSS at NCAR is 2:1, contrasting with conventional wisdom that an MSS serviced more writes than reads. Additionally, humans must wait for the results from reads, while users would not need to wait for writes to tape to complete. This suggests that an algorithm should not wait until it is absolutely necessary to free up space; instead, it should write data to tape relatively quickly, and then mark the file as "deleteable." Since files would be written lazily, their placement on tertiary media could be optimized, making future reads run faster. A mass storage system should be optimized to make read access to files faster at the cost of requiring more work for writes. This will make the system seem faster to its users at little additional cost.

7 Conclusions

This analysis of file movement between secondary and tertiary storage at a supercomputer Unix site provides several important hints for designers of file migration systems. First, humans wait for reads, while computers wait for writes. Any migration policy should consider this, and optimize for reading. The write rate is relatively steady over time, while reads vary greatly. Thus, migration algorithms should move files to tertiary storage whenever resources (tape drives, etc.) are available, and use the extra space to prefetch files which might be read shortly.

Files have become larger and more numerous since the early 1980s. Currently, there are over 900,000 files on the MSS at NCAR averaging over 25 MB each. On the other hand, their reference patterns have not changed much. File rereference rate still drops off sharply after the first few days, though it does level off soon thereafter. Files are also infrequently rereferenced; more than half of the files were only accessed once in two years. Again, this suggests that files can be migrated to a less costly storage medium if they are unreferenced for only a few days.

The NCAR system appears to be a typical large Unix-based scientific computing center. Thus, the analysis in this paper will help system architects design hardware and software best suited for storing and rapidly accessing the terabytes of data that such systems must store. While reference patterns for these data have not changed much in the last decade, more files, larger files and new tertiary storage technologies will require new mass storage systems and new migration algorithms to run them.

Acknowledgments

The authors would like to thank the staff at NCAR for all their help in gathering the traces and understanding their format. Special thanks go to Bernie O'Lear for providing access to the data at NCAR and to Dennis Colarelli for his assistance with the NCAR system. We would also like to thank our colleagues at Berkeley and elsewhere for their helpful comments on drafts of this paper.

References

[1] Edward R. Arnold and Marc E. Nelson. "Automatic Unix backup in a mass-storage environment." In *USENIX — Winter 1988*, pages 131–136, February 1988.

[2] Donald L. Boyd. "Implementing mass storage facilities in operating systems." *Computer*, pages 40–45, February 1978.

[3] Sam Coleman and Steve Miller. "Mass storage system reference model: Version 4." IEEE Technical Committee on Mass Storage Systems and Technology, May 1990.

[4] Ann L. Drapeau and Randy H. Katz. "Striped tape arrays." In *Digest of Papers*. Twelfth IEEE Symposium on Mass Storage Systems, 1993. To appear.

[5] Carrel W. Ewing and Arnold M. Peskin. "The Masstor mass storage product at Brookhaven National Laboratory." *Computer*, pages 57–66, July 1982.

[6] Gordon George Free. "File migration in a UNIX environment." Master's thesis, University of Illinois at Urbana-Champaign, December 1984.

[7] Robert L. Henderson and Alan Poston. "MSS II and RASH: A mainframe UNIX based mass

storage system with a rapid access storage hierarchy file management system." In *USENIX — Winter 1989*, pages 65–84, 1989.

[8] David W. Jensen and Daniel A. Reed. "File archive activity in a supercomputer environment." Technical Report UIUCDCS-R-91-1672, University of Illinois at Urbana-Champaign, April 1991.

[9] David D. Larson, James R. Young, Thomas J. Studebaker, and Cynthia L. Kraybill. "StorageTek 4400 automated cartridge system." In *Digest of Papers*, pages 112–117. Eighth IEEE Symposium on Mass Storage Systems, November 1987.

[10] Duncan H. Lawrie, J. M. Randal, and Richard R. Barton. "Experiments with automatic file migration." *Computer*, pages 45–55, July 1982.

[11] Fred W. McClain. "Mass storage at the San Diego Supercomputer Center." In *Digest of Papers*, pages 81–86. Eighth IEEE Symposium on Mass Storage Systems, November 1987.

[12] Marc Nelson, David L. Kitts, John H. Merrill, and Gene Harano. "The NCAR mass storage system." In *Digest of Papers*. Eighth IEEE Symposium on Mass Storage Systems, November 1987.

[13] A. Dain Samples. "Mache: No-loss trace compaction." Technical Report UCB/CSD 88/446, University of California at Berkeley, September 1988.

[14] Alan Jay Smith. "Analysis of long term file reference patterns for application to file migration algorithms." *IEEE Transactions on Software Engineering*, 7(4):403–417, July 1981.

[15] Alan Jay Smith. "Long term file migration: Development and evaluation of algorithms." *Communications of the ACM*, 24(8):521–532, August 1981.

[16] Ken Spencer. "Terabyte optical tape recorder." In *Digest of Papers*, pages 144–146. Ninth IEEE Symposium on Mass Storage Systems, November 1988.

[17] Stephen Strange. "Analysis of long-term UNIX file access patterns for application to automatic file migration strategies." Technical Report UCB/CSD 92/700, University of California, Berkeley, August 1992.

[18] Erich Thanhardt and Gene Harano. "File migration in the NCAR mass storage system." In *Digest of Papers*, pages 114–121. Ninth IEEE Symposium on Mass Storage Systems, November 1988.

[19] David Tweten. "Hiding mass storage under UNIX: NASA's MSS-II architecture." In *Digest of Papers*, pages 140–145. Tenth IEEE Symposium on Mass Storage Systems, May 1990.

[20] Sandra J. Walker. "Cray Computer, MSS, MASnet, MIGS and UNIX, Xerox 4050, 4381 Front-End, Internet Remote Job Entry, Text and Graphics System, March 1991." Technical report, National Center for Atmospheric Research, Scientific Computing Division, March 1991.

[21] David L. Williamson, Jeffrey T. Kiehl, V. Ramanathan, Robert E. Dickinson, and James J. Hack. "Description of NCAR Community Climate Model (CCM1)." Technical Report NCAR/TN-285+STR, National Center for Atmospheric Research, June 1987.

Author Information

Ethan Miller received a BS in computer science from Brown in 1987, and an MS from Berkeley in 1990. He is currently a PhD candidate in computer science at Berkeley, where he is a member of the RAID project. He is interested in file systems and data storage for high performance computing, including both disk and tertiary storage. His U.S. Mail address is Computer Science Division; 571 Evans Hall; University of California; Berkeley, CA 94720. Electronic mail sent to elm@cs.Berkeley.EDU will also get to him.

Randy Katz has been on the Berkeley faculty since 1983. He received his MS and PhD at Berkeley in 1978 and 1980 respectively. He received his AB degree from Cornell University in 1976. He is the principal investigator of a DARPA and NASA sponsored project to construct high performance, high capacity storage systems for diskless supercomputers. His U.S. Mail address is the same as above, and his e-mail address is randy@cs.berkeley.edu.