Long-term File Activity and Inter-Reference Patterns

Lt. Colonel Tim Gibson
HQ, Pacific Command
Attn: J641
Camp Smith, HI, 96861
tgibson@acm.org

Ethan L. Miller
Department of CSEE
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
elm@csee.umbc.edu [1]

Darrell D. E. Long
Jack Baskin School of Engineering
University of California
Santa Cruz, CA 95064
darrell@cse.ucsc.edu [2]

We compare and contrast long-term file system activity for different Unix environments for periods of 120 to 280 days. Our focus is on finding common long-term activity trends and reference patterns. Our analysis shows that 90% of all files are not used after initial creation, those that are used are normally short-lived, and that if a file is not used in some manner the day after it is created, it will probably never be used. Additionally, we find approximately 1% of all files are used daily. This information allows us to more accurately predict the files which are never used. These files can be compressed or moved to tertiary storage enabling either more users per disk or larger user disk quotas.

1 Introduction

Physical storage devices have long been the slowest components of any computer system. While disk and tape storage devices have improved in the last decade, their performance has not kept pace with rapid increases in processor speed. This presents a challenge to storage system designers because faster CPUs encourage both more and larger files, placing a higher demand on the file system and storage devices. This problem has long been an issue for supercomputer centers which have always managed huge quantities of data, but recent advances in CPU performance have brought traditional supercomputer power to the desktop. Thus, system designers must insure that workstation file systems can keep up with the increased bandwidth and capacity that increased CPU speed brings. While much previous work has focused on improving short-term performance, we focus on long-term reference patterns and their influence on performance over days.

This paper is organized into nine sections. We begin by reviewing previous disk activity studies in Section 2. In Section 3, we briefly discuss our data collection and analysis tools, which differ significantly from those used in earlier studies. We describe the different types of computing environments from which we collected data in Section 4. The software written for this paper analyzes the collected data and generates statistics. The simplest analysis mode provides information about daily activity. This is shown in Section 5. Analysis of long-term trends is shown in Section 6. An interesting product from this research is a comparison of the same file system's activity from either the file name view, or from the operating system's underlying numeric index. This comparison is done in Section 7. We summarize our findings in Section 8 and briefly discuss our future research in Section 9.

2 Related Research

In the 1980's, both Smith [20], and Ousterhout [15] made detailed studies of file activity on computing systems. While their observations are still useful, some of the underlying structure has lost relevance. For example, Smith primarily observed text-based user files for thirteen months; the size and nature of today's multimedia files, unforeseen when Smith collected his data, are much different from the text-based files he studied. Ousterhout's very detailed file traces were conducted over three to four day periods. While Ousterhout's work is very useful, his traces were collected over such a short time that long-term trends cannot be predicted from his data. Baker's distributed file system activity logs from 1991 [2] update Ousterhout's work, concentrate on low-level disk activity, and have the same short trace periods as

[1] Supported by the NASA Ames Research Center under grant NAG 2-1094.
[2] Supported by International Business Machines Corporation (IBM).

This paper appeared at the 24th Annual International Conference on Computer Measurement and Performance (CMG '98), Anaheim, CA, December 1998, pages 976–987.

Ousterhout's traces. More recently, Spasojevic and Satyanarayanan [23] analyzed the long-term behavior of a wide-area distributed file system, focusing on the performance of the Andrew File System. They were primarily interested in issues such as caching, replication, and reliability in a large, distributed file system, and used volumes rather than files as the basis for their study, limiting its usefulness in showing long-term behavior at the file level.

Other studies, [7,10,11,12] are directly applicable to supercomputing centers, but may not apply to smaller computing centers; both the size and number of files at supercomputing centers far exceeds "normal" computing activities. Also, supercomputing centers usually have large tape libraries with tape robots providing near-line storage for their data files, in some cases exceeding hundreds of terabytes. In contrast, smaller computing centers do not have tape robots and only use tape for archiving purposes—a trend that we believe may change as near-line storage robots with either tape, digital-versatile–disk (DVD), or high-capacity magneto-optical drives, become more affordable and more practical.

Our work most closely resembles Strange's disk studies from 1992 [24]. We collect much of the same information as Strange, and in fact corroborate a good number of his findings. However, the traces used here cover twice the time period as Strange's traces. Additionally, our analysis tool maintains a database record for every file residing on the file system. This database information allows us to keep track of how individual records are accessed, modified, and deleted. This temporal information on file activity is new and has not been collected or analyzed before by other authors.

3 Tools

Our tracing system was designed with one major goal: gather useful information without requiring operating system kernels to be recompiled. Our trace gathering tool is a modified version of the GNU *find* utility, which can be used to collect information on all the files in a file system or directory. The collected information includes the file's index node number (i-node), size, name, access time, i-node creation time, modification time, owner, and group. The tracing program can be run any time, and if the output is placed into a directory or file system that is not being studied, the tracing process is invisible to itself. Additionally, the file's name can be scrambled to ensure user privacy on public systems while preserving information about the relationships of files in the system.

The second component of our system sorts two trace files by i-node number, compares them, and generates a file containing only references to files which have changed, or have been created or deleted. This program reduces the long-term storage requirements and the amount of data which is analyzed later. Difference files generated by this tool are fed into the analysis program that generates statistics on the items shown in Table 1. An additional advantage of using file differences is the analysis tool runs faster because of the reduced amount of I/O.

Our statistics collection and analysis package for file systems has some weaknesses. Ousterhout [15] noted that 80% of all file creations have a lifetime of less than three minutes. Because the daemons, compilers and other programs that created these files during Ousterhout's work still exist, we miss many of the temporary files they create. However, our collection system is designed to gather information about long-term disk use and file system activity and growth; temporary files that exist for less than three minutes will not be moved to long-term storage and do not contribute to long-term growth.

Additionally, the differencing program cannot determine how many times a file is accessed or modified in a single day. It only notices that an access or modification occurred and when the most recent event happened. Fortunately, what matters from a long-term perspective is that the file was used on a certain day, not how many times it was used.

The only way to collect more detailed information is to either run the tracing system more often or to change the operating system kernel and generate a large detailed log file. Both of these options place a heavier load on the computer system and we deliberately decided against doing either. Additionally, while collecting data with kernel modifications may provide more data, it severely restricts the number and type of computing installations willing to collect traces. We had a difficult time convincing system administrators to allow us to run our current collection program; modifying multiple operating system kernels and convincing the different system managers to use the modified kernels (and to provide us the additional log file disk space) was not a feasible solution.

Finally, there are two ways to "look" at any file on a Unix file system. The normal way is with the file's path and name, and is exactly what people and most programs do when they open or close a file. However, the Unix operating system does not use file names to manage files. Instead, the operating system gives each file a unique number. We examine the activities of both these file system views, based upon the file names and the i-nodes, respectively. To distinguish between the two, we use the terms hierarchical name space and numeric index. The reason to examine these two views of the same file system activity is they often differ in significant ways. For example, the numeric index system view has a large number of file deletions and very few file modifications. This happens because most applications do not modify physical disk files, they simply erase the old version and replace it with a new one. As a result, although a person may edit the "same" file daily, in actuality a new file is created every time she saves the file. This is in contrast to the hierarchical name space view, based on file names, where the number of file modifications greatly exceeds file deletions.

In order to study the hierarchical name space view of the file system activity, we use a small conversion program which changes the file name in a trace file into a number based a MD-5 hash of the file name [16].

File Activity	Statistics Collected
Accesses, Creates	Total number of files and bytes. Produces histograms, grouped by file size, of the number of accesses or creations.
Deletions	Same as Access, with deletions where the i-nodes are reused tracked separately from deletions where the i-node is not reused.
Modifications	Same as Access with categories for files that are modified and increased in size, decreased in size, or remained the same.
Modification Differences (Deltas)	Same as Access with categories for files that are modified and increased in size or decreased in size. Tracks files by the amount of change. Produces a two dimensional histogram of file size versus the amount of the modification.
Change of Name, Owner, or Group	Separate categories for change of name (Unix mv) change of owner (chown), change of group (chgrp). Outputs the number of files only.
Long-Term Deletions	Separate data is kept for overall deletions, files that were accessed before deletion, files which were modified before deletions, how many times individual files were accessed or modified before deletion. Produces two dimensional histogram by file size and days the file 'lived' before it was deleted.
Inter-Access and Inter-Modification Period	Summary of accesses and modifications for the last 30 days. Two dimensional histogram by file size and how many days pass between file accesses (or modifications) on individual files.
Files on System at Analysis Completion	Summary of file system status at the end of the trace period. Produces two dimensional histogram by file size and number of times files have been accessed or modified. One dimensional histogram, by file size, of files that have never been used.

Table 1. Statistics collected by the analysis program.

4 Computing Environments Examined

We collected and analyzed daily traces for analysis on four different computing systems. Two are at the University of Maryland, Baltimore County (UMBC), the third is at the University of California at Santa Cruz (UCSC), and the fourth is at the U.S. Army's Aberdeen Proving Ground. Table 2 summarizes these systems. The UMBC Computer Science (CS) department's file system trace collection began in October 1996. We expanded our collection efforts on the CS systems in May 1997, hence the long and short collection periods. We also began collecting traces from the Aberdeen Proving Ground (APG) and the University Computing Services (UCS) file systems in May 1997.

The University of California traces were collected daily with a different tracing process and do not contain all long-term file system actions. Instead, they provide a listing of the file names, along with the file's size, access, modification, and creation times, which were modified during the past 24 hours. This affects the amount of analysis we can provide.

	Aberdeen Proving Ground (APG)	University Computing Services (UCS)	CS Dept. (Long Period)	CS Dept. (Short Period)	Univ of CA, Santa Cruz
Average Number of Files on System	72,000	1,320,000	230,000	690,000	2,300,000
Disk Capacity (avg. percent full)	3.6 GB (80%)	35 GB (70%)	11 GB (70%)	28 GB (50%)	84 GB
Type(s) of FS traced	User and System	User only	User only	User and System	User and System
Number of FS traced	7 user, 1 system	9 user	4 user	6 user, 2 system	49
Type of System	Administrative	General-purpose	Development	Development	Development
Number of Users	300 full-time, 2,000 email only	~15,000	~500	~500	~500
Trace Length	210 days	239 days	287 days	157 days	207 days

Table 2. Summary of Systems Examined

5 Basic File System Activity

This section analyzes daily file activity using the file system analyzer's simplest mode, which has no long-term memory about individual files. As a result, it provides very little information about individual file activity.

5.1 Distribution of File Sizes

The first file system attribute we examined was the distribution of file sizes across the different file systems, as shown in Figure 1. This figure shows both the percentage and cumulative percentage of files on each system by file size groupings. The bar chart shows the percentage of files in each size category, while the lines are the cumulative percentage.

The interesting thing to note in Figure 1 is that most files are relatively small, more than half are less than 8 KB. While the graph stops at 1 MB, this encompasses 99.5% of the files collected at the university and 96% of the files on the APG system. The APG users occasionally had a large word processing, email files, keeping the APG averages lower than the CS and UCS system averages. Surprisingly, there is also little difference in size distribution between the Computer Science user and system-level file systems.

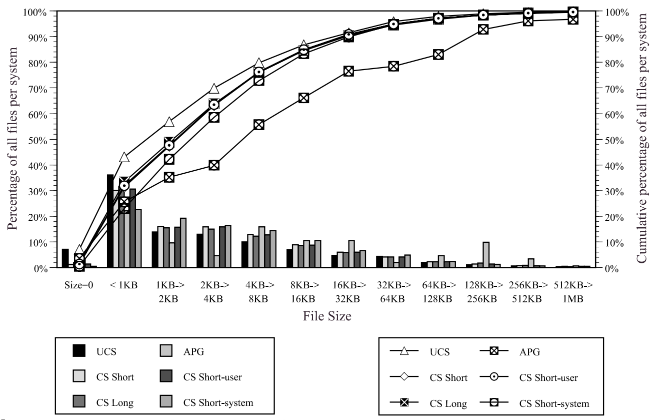

Figure 1. File Sizes, by size category and cumulative.

One of the most important facts to gather from Figure 1 is that 80% or more of the files are smaller than 32 KB. On the other hand, while only 25% are larger than 8 KB, this 25% contains the majority of the bytes used on the different systems, as shown in Figure 2. This figure depicts the total number of megabytes used by file size category on each system. The totals between the different systems vary because the disk capacities are different.

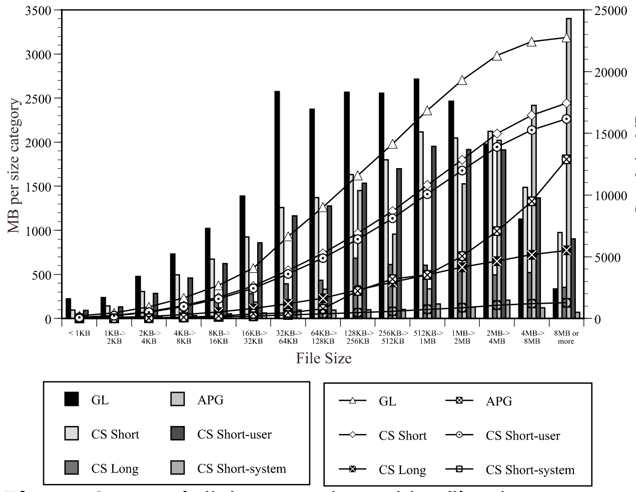

Figure 2. Total disk spaced used by file size category. The bar chart shows distribution by size category, the line chart shows the same data cumulatively.

5.2 Usage by Transaction Type

The software's simplest analysis mode provides a variety of information on the number of files and bytes used daily on each file system. Figure 3 and Figure 4 show the average daily file use per transaction type on the different systems. Figure 3 shows, as a percentage of daily file activity, the percentage of files accessed, created, deleted, or modified. In all cases, the greatest percentage is in the accessed category. Approximately one-half as many files are created and deleted compared to those accessed. This is logical because many files are only accessed (*e.g.*, program files, data files, and configuration files), rather than created or deleted. The number of file modifications lags far behind all other actions on the Computer Science systems. The reason there are so few modifications on the CS file systems is that many computer applications, particularly the applications programmers use, do not modify files, rather they create new files and rename or delete the original.

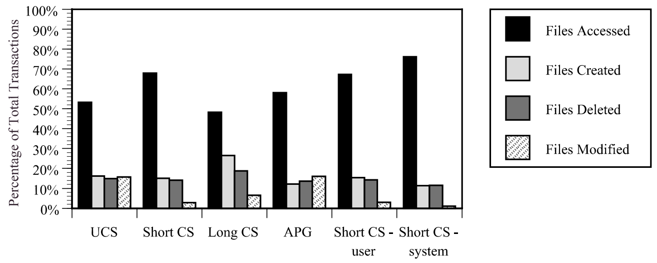

Figure 3. Activity by type, percentage of total transactions.

For instance, when a programmer changes a file with a text editor and saves the changes, the following steps occur:

- User gives command to open the file /home/profile
- Operating System finds the i-node (e.g., 34,717) associated with /home/profile
- Operating System gets the file's disk block numbers from the i-node
- Operating System reads the disk blocks and loads the file into memory (RAM)
- User makes changes and "saves" the file
- Operating System writes the modified file to a new set of disk blocks with a new i-node
- Operating System frees the old disk blocks and the old i-node
- Operating System updates the file name /home/profile to point to the new i-node.

As a result, from the viewpoint of the operating system's numeric index, saving the file with a text editor results in a file creation and a file deletion, not a file modification. Most programs, including word-processors, use the same technique as the text editor. An exception to this general rule is electronic mail (e-mail) packages, which actually modify the files contained in a user's in-box. Note: the in-box for electronic mail on Unix systems is one file, regardless of how many messages are contained within the in-box.

The result of this exception for e-mail files is seen in Figure 4. Figure 4 is similar to Figure 3, except that it displays bytes instead of files. On both the UCS and the APG systems, most users only use the computer system for e-mail. The result is that while only 15% of the files on these systems are modified daily (as shown in Figure 3), these modifications account for over 70% of the bytes used daily (in Figure 4) on the UCS and APG systems. The Figure 4 increase in modifications can also be seen in the Computer Science traces; the increase is not as noticeable in the

CS data because most CS users do more with the system than read e-mail.

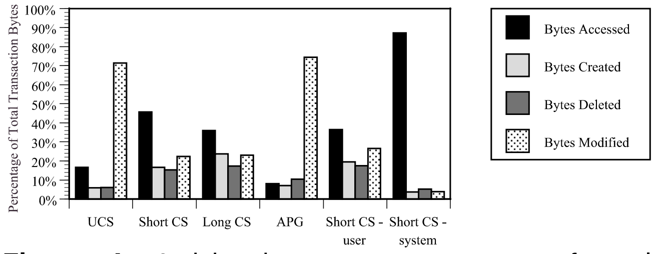

Figure 4. Activity by type, percentage of total transaction bytes.

While Figure 3 and Figure 4 provide interesting observations about the manner in which users and applications use and modify files, most importantly they illustrate the need to optimize file accesses.

5.3 Growth of Individual Files from Modifications

The basic analysis also provides information about the ratio between modifications that make a file larger, modifications that leave the file the same size, and those that make the file smaller. On the CS systems, all these categories of modification were approximately the same. However, on the UCS and APG systems, the number of files modified that increased in size was double the other two categories added together. While 46% of the modifications on the CS systems increased a file by 512 bytes or less, 67% of the modifications on the UCS system were in the 1 KB to 16 KB range. Again, we attribute this to the e-mail files, because receiving a few e-mail messages daily causes this type of increase.

On the UCS system, e-mail activity also appears in how files decrease in size. This occurs because many users keep their e-mail messages for a long time, with their single in-box file slowly growing in size. However, users eventually delete many of the old messages, usually at one time. On the UCS system, 48% of file modifications occurred when a file lost between 4 KB and 32 KB. The APG system behavior is similar. In contrast, 69% of the CS file modifications reduce files less than 512 bytes, with only 16% falling into the 4 KB to 32 KB range. While the UCS and the APG activity is in keeping with how users handle e-mail files, the activity on the CS systems is consistent with development activity. Both these findings and observations are in accord with the different system environments.

This file growth rate has direct implications for operating system design. When a file needs to increase in size, some operating systems allocate additional space based on the file's original size—the larger the file the more space is allocated. In order to recover unused disk space, the operating system must keep track of the files that received additional space, and recover any unused disk space at a later time. However, as we have shown, files do not increase in direct proportion to their original size. Because relatively few Unix files increase in size at all, it may be more efficient to allocate disk blocks differently. For example, disk blocks can even be allocated one block at a time — writing some files will take longer, but the additional CPU cycles to find unused disk blocks are eliminated.

5.4 File System Growth

The previous sub-sections showed that:

- Few numeric index files are modified;
- At most, half of all numeric index modifications result in a file size increase; and
- Most numeric index file size increases from modifications are less than 32 KB.

A logical question to ask after discovering these facts about file modifications is 'what causes file systems to grow?' Figures 5 and 6 show byte creep on the CS and UCS systems. Byte creep is the cumulative number of bytes added to the system by creations and modification increases, with the number of bytes deleted or lost from modification decreases subtracted. Both the UCS and the CS file systems became larger during the trace collection, however, the reasons for these increases were different. For example, the UCS system increased primarily because existing files became larger from file modifications. The number of bytes gained or lost from file creations and deletions remained relatively constant. Note: The spike in the UCS trace on day 55 and day 56 is caused by a disk drive failure and a tape restore over a weekend.

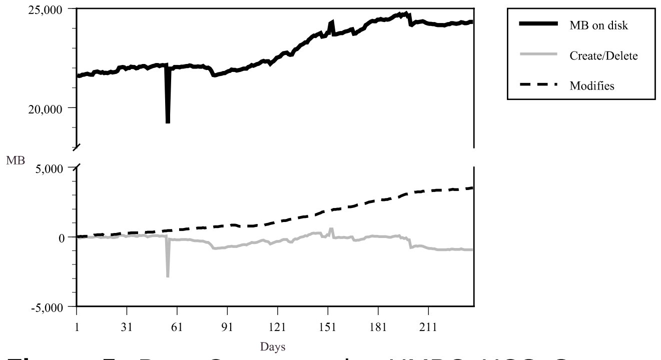

Figure 5. Byte Creep on the UMBC UCS Systems (9 file systems, 239 days, May 1997—January 1998).

The CS file system, on the other hand, exhibits markedly different behavior. Nearly all file system growth on the CS system came from new files, while file system growth or loss from modifications remained relatively constant. We believe the cause of this disparity is the different user populations and applications on the two systems. While the overall file system growth is the same in both cases, the cause is different. The information in Figures 5 and 6 is of direct interest to system administrators for whom the source of file system growth is a constant question.

5.5 Percentage of Files Used Daily

Both Figure 5 and Figure 6 require a break in the graph's y-axis to allow detail in the graph's lower section. This is because the proportion of bytes used to total bytes on the system is relatively low. In fact, very few files, or bytes, on a file system are used daily. Figure 7 shows the average percentage of files and bytes used daily on the different systems.

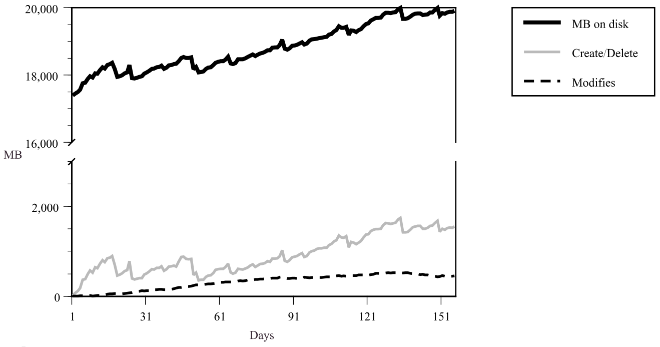

Figure 6. Byte Creep on the UMBC Comp. Science System, 8 file systems, 157 days, Aug 1997-Jan 1998.

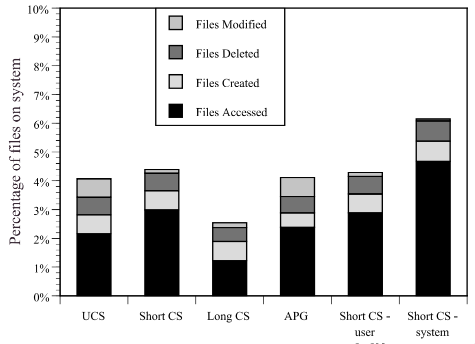

Figure 7a. Average percentage of files used daily.

What is clearly seen in Figure 7a is the fact that relatively few files are used on any one day — normally less than 5%. Similarly, the number of bytes used daily is also low, as Figure 7b shows. The percentage of bytes used on the UCS and the APG systems is higher because the smaller disk space quotas make users delete unnecessary files. However, in no instance does the number of bytes used daily exceed 25%. Otherwise, Figures 7a and 7b are in keeping with earlier findings in this chapter where we showed that most file activity is caused by file accesses, and that byte activity is either caused by creations and deletions, or modifications, depending on the type of user. The fact that so little of the data on the file system is used is important because it provides support for using an integrated tertiary storage system (or compressing unused files) for the vast majority of files that are not regularly used.

5.6 Summary of Daily Statistics

The last few sub-sections presented the information and data analysis capabilities from the simplest analysis mode. The information available in this mode consists primarily of the number of files and bytes used daily by the different file transaction types. Information about the system's overall activity, file size distributions, the amount of file increases and decreases from modifications, and the source of file system growth can all be analyzed. However, this information is based upon what transpired on an individual day. Information in not kept about individual files, limiting this mode's analysis capabilities.

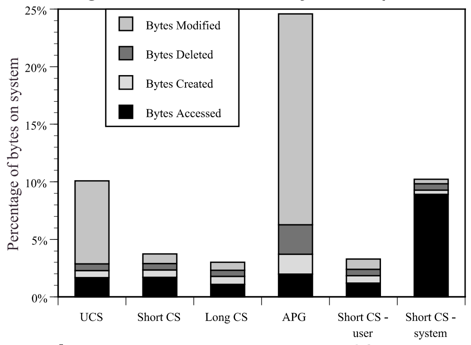

Figure 7b. Average percentage of bytes used daily.

6 Long-term File System Activity

While Figures 5 and 6 showed that only a small proportion of files are used each day, we have not yet shown individual file usage patterns. Tracking individual activity patterns is important; if less than 5% of the files are used daily—as Figure 7 shows—but each day that 5% represents different files, in less than a month all the files on the file system will be used. If this is true, it will be very difficult to predict which files will be used daily, rendering integrated tertiary storage impractical. However, if the 5% represents many of the same files every day, integrated tertiary storage or compression may be viable.

In contrast to the simple analysis mode's daily usage patterns, a more advanced mode provides the ability to monitor individual file activity patterns. Figure 8 shows that fewer than 30% of the files remaining on any file system at trace completion were used during the trace period (the trace periods were all longer than 150 days), with one exception. The exception is the CS system-level files where a higher usage rate is seen, as expected, because many of the files are commonly used programs and libraries. Overall, fewer than 50% of all non-system–level bytes were used on any system. Figure 8 shows that most of the files and bytes remaining on the file systems were never used in any way during the trace collection period. Consequently, if the files which were never used resided on tertiary storage instead of disk, their absence from the disk would have no effect at all.

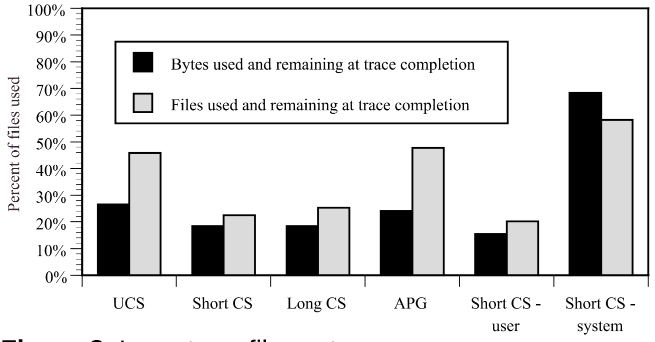

Figure 8. Long-term file system usage.

One of the file system activity traits which can be inferred from Figure 8, particularly when compared to Figure 5 and Figure 6, is that some files are used regularly. If this was not the case, and all files on the file systems were used intermittently, the daily usage percentages could be the same, but the end-of-trace percentages would probably be higher. However, this is not what happens, as we will show in the next three sub-sections.

6.1 Long-term Repeated Usage

Another analytic tool we provide is the inter-access and inter-modification period statistics shown in Figure 9 and Figure 10. Both these figures show the number of days between consecutive accesses or modifications. For example, a file that is accessed on both Monday and Wednesday has a two day inter-access period. Both Figure 9 and Figure 10 show that most accesses and modifications occur on consecutive days. Generally, if a file is accessed one day and not the following day, the file has less than a 10% chance of being accessed on the third day. In fact, the file has between a 15% and 45% chance of ever being accessed again, depending on the computer system. There is one anomaly in Figure 9 access patterns; the CS system-level files occasionally have small spikes. They are probably caused by some type of system routine (or user program) which periodically accesses many system-level files that are otherwise rarely used.

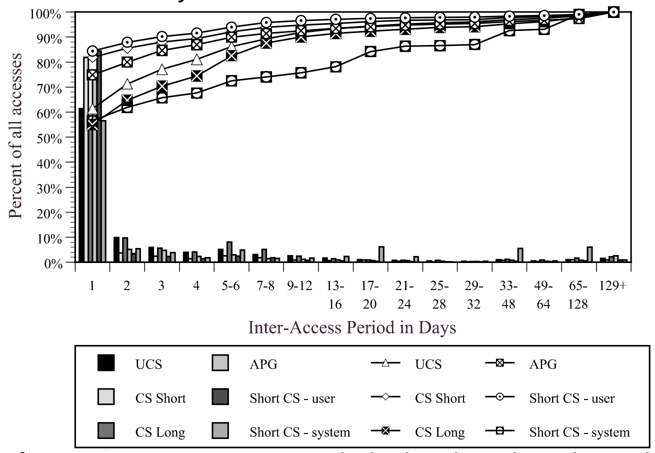

Figure 9. Inter-Access period showing the elapsed time between consecutive accesses. The bar chart shows distribution by size category and the line chart shows the same data cumulatively.

Figure 10 does not have any modification spikes or other anomalies. However, the percentage of files modified with one day intervals is lower than for files accessed. Similarly, the percentage of files modified per time interval falls off more slowly than for files accessed—many users checking their individual e-mail files once every several days can explain this behavior.

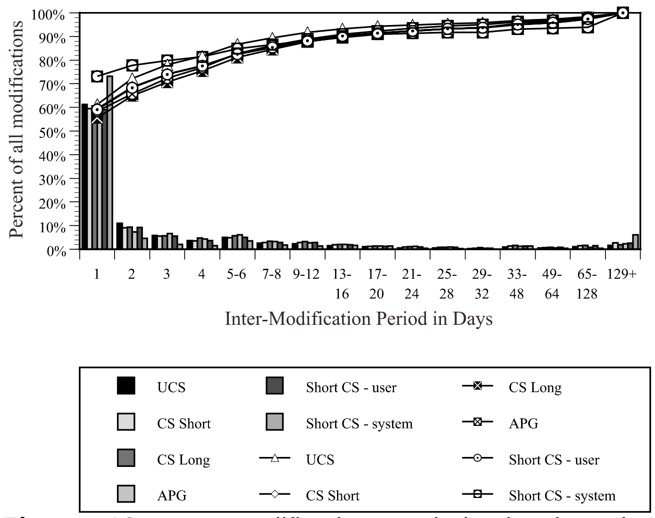

Figure 10. Inter-Modification period showing the elapsed time between consecutive modifications. The bar chart shows distribution by size category, the line chart shows the same data cumulatively.

When the information in Figure 9 and Figure 10 is used in conjunction with data presented earlier in this section on long-term statistics, there is growing evidence that files which are used—accessed or modified—are used on a regular basis. Conversely, files that are not used, tend to remain unused. Again, files that are not regularly used could be either compressed or moved to tertiary storage without affecting system performance. Figure 9 and Figure 10 both reinforce the proposition that a small percentage of files are used regularly, and a large percentage of files are used infrequently, or not at all.

Figure 11 illustrates the activity of files that are used (accessed or modified) and proves that relatively few files are used regularly throughout the trace period. The x-axis of Figure 11 is a series of categories showing the number of days a file was accessed or modified (or both). The categories extend from one day to more than 129 days. Because all the traces were collected for at least 157 days (usually more than 200 days), all the categories are valid for all the file systems. The y-axis percentage for each category is based upon the total number of files created during the trace period for each file system.

Figure 11 shows that 5–10% of all files created are only used on one day, depending on the system. On the other hand, approximately 0.8% of the files are used more than 129 times—essentially every day. These files which are used more than 129 times account for less than 1% of all files created and approximately 10% of all the files which were accessed or modified. Obviously, for a tertiary storage system to work efficiently, these files must be kept on disk at all times.

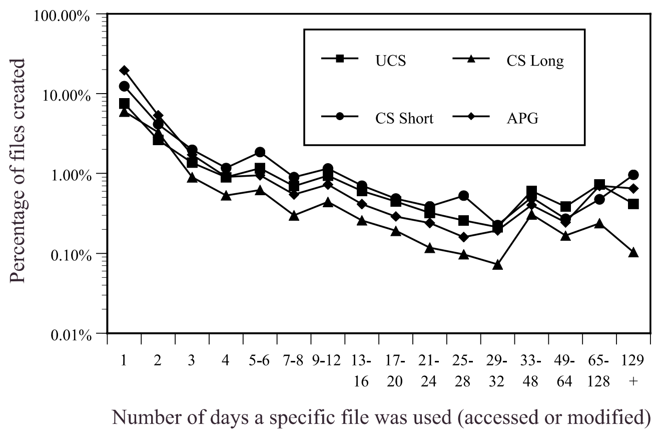

Figure 11. Number of days a specific file was either accessed or modified after it was created. Shown as a percentage of all files created during the trace period, *logarithmic scale*.

6.2 Reference Locality

The fact that a very small number of files are used nearly every day increases the likelihood that most files which are used exhibit reference locality. Locality of reference is a term normally used with RAM and cache memories. When data is moved into the cache memory, it is very likely to be used again. Indeed, this property is what makes cache memory systems worthwhile. Caching stratagems try to ensure that data needed in the immediate future is kept in the cache. Because the file system can be viewed as a super-set of the RAM and cache data, it is possible that the files on secondary storage also have the locality of reference property. In fact, this is true, as Figure 12 shows. One of the statistics the software collects is the number of files accessed and modified during the last X of Y days. The data shown is from the 7 day interval. The y-axis in Figure 12 is based on the average number of all files that were accessed or modified daily per system. For example, on an average day, 36,994 files were accessed daily on the UCS system. Likewise, an average of 2,380 files were used every day in the last seven days. Thus, an average of 6.48% of the files accessed had been accessed daily for the last week.

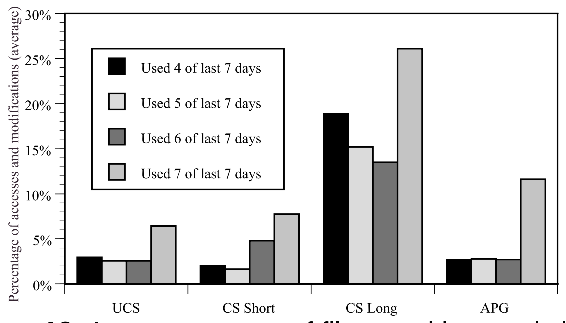

Figure 12. Average percent of files used in exactly last X of 7 days. Shown as a percentage of the average number of daily accesses and modifications.

This data allows us to verify the presence of reference locality for file accesses and file modifications. One note of interest about Figure 12 is the fact that on every system more files are used every day of the week than for any other single sub-period.

The only anomaly found in Figure 12 is the high level of activity for the long UMBC Computer Science traces. This is explained by the fact that the faculty3 file system in the long CS traces became full and inactive during the trace collection period, and two others (faculty and grad2) became less active as the trace period progressed. For example, the faculty3 file system averaged less than 220 file transactions per day, compared to 3,311 transactions per day on the most active file system from the same traces. As a result, the combined activity level for these traces was lower than any other trace (see Figure 7 and Figure 11). However, while these three file systems were less active than other file systems, the files which were active on these three tended to be more active than on other file systems. This is because some users were running out of disk space and had to keep unused files at a minimum. The result is seen in Figure 12. Regardless, the overall activity level for the long Computer Science traces as a percentage of total files—Figure 12 is a percentage of total accesses and modifications—is relatively low, as shown in Figure 7 and Figure 11.

6.3 File Lifetimes

One reference locality characteristic that cache data has is that the cache data is eventually replaced. In the case of file systems, this translates to the files either lapsing out of use or being deleted. Figure 8 shows that most of the files on a file system are never used. By comparing Figure 8 with Figure 7, it can be inferred that the number of files on the system which were used at least once increases over time. (In fact, the simulator shows this directly, but this graph is not shown for brevity.) The proposition that most files tend to lapse out of use can be supported by the facts that most files were not used during the trace periods, and that most files which were used tended to only be used a few times.

In keeping with the theory of reference locality, the deletion rate for used files should initially be fairly low, and increase as the files get older. This slowly increasing deletion rate for used files is shown in Figure 13. The deletions are shown as a percentage of all files deleted with the same lifetime. For example, the UCS system has a value of 11.02% for the two day lifetime. This value is computed by dividing the number of files which were used and then deleted two days after being created (15,373) by the total number of files deleted two days after being created (139,491). Note: A file that is created one day and deleted before the next trace lives ‘zero’ days.

Figure 13 shows that files which are used tend to live longer in comparison with files which are not used. For example, files which are used account for less than

20% of the files deleted in the first week after they are created. Thereafter, the percentage of files which were used and deleted rises. This is true for all systems except the APG system, where the lifetime of used files varies widely. We believe the APG data is a result of both the user behavior and strict disk quotas. However, even with the APG data, Figure 13 clearly reinforces the theory of locality of reference by showing that files which are used are deleted more slowly than files which are not used. This is particularly true on all three university systems where few active files are deleted until 20 to 30 days have passed.

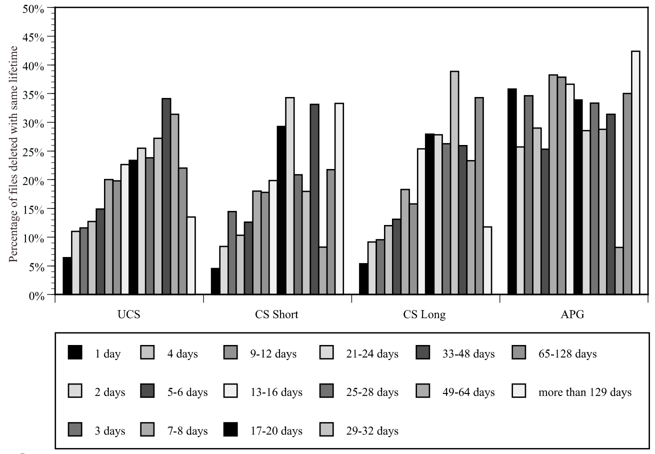

Figure 13. Lifetimes of used files as a percentage of all files deleted with the same lifetime.

To put Figure 13 in perspective and to give the reader a grasp of both the file deletion rate and the relationship between deletions for used versus unused files, we provide Figure 14. This figure shows the cumulative rate of deletions for files that were used (accessed or modified) and the cumulative rate of deletions for files that were created and never used. These rates are shown as a percentage of all the files that were deleted during the entire trace period. For example, approximately 30% of all deletions are deletions of files that were on the system for only one day (present at one trace and gone the next). The figure also shows that 85% of all deletions are upon files that were not used, except for the files on the APG file system. Similarly, for the first two weeks, unused files at the university are deleted more quickly than files that are used. At the two week point, unused files are deleted at approximately the same rate as files that were used. Interestingly, the deletion rates for the different file types are virtually identical on all three university systems.

The APG data is different from the university data. Unused files are deleted more slowly, while the used files are deleted more quickly. The deletion rate on the APG system increases for both types of file at the two month point, possibly from users conducting periodic housecleaning.

6.4 Summary of Long-term File System Activity

The data and analysis presented in the preceding three sub-sections are extremely important for understanding file system activity, and to our knowledge the extensive data collection and analysis is unique. To summarize our findings:

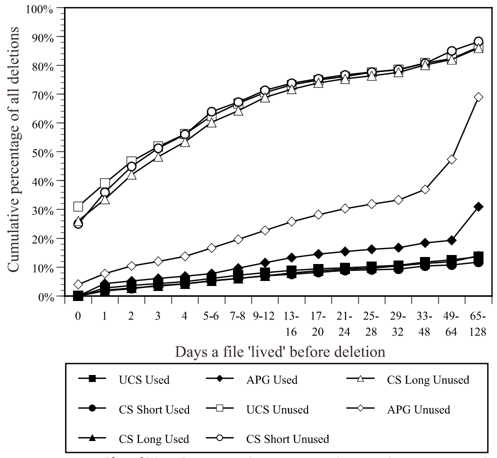

Figure 14. File lifetimes for used and unused files as a percentage of total deletions.

- Files which are used have short inter-access and inter-modification times;
- An extremely small number of files are used every day (normally less than 1% of all files on the file system);
- Files which are being used exhibit reference locality, where a file that is used is likely to be used again, and vice versa;
- Files which are not used for a several days are unlikely to be used again; and
- Files which are not used are deleted more quickly than files that have been used.

7. Hierarchical Name Space Activity

In the preceding two sections we discussed and analyzed disk activity as seen from the viewpoint of the operating system's numeric index. As pointed out earlier, the disk activity shown by the numeric index is not necessarily the same activity the user observes using the hierarchical name space viewpoint. This section shows the key file system activity similarities and differences observed when data is collected using the hierarchical name space viewpoint instead of the numeric index.

The most important thing to remember when comparing the file system's activity using the hierarchical name space (i.e., the hierarchical paths and file names) with the numeric index is that they are merely two different views of the same activity. The traces collected information regarding the activity of the physical files on the file system; any changes made to the files during the trace period were made on both the hierarchical name space view and the numeric index view of the file system. Any differences between these two views can be ascribed to what the view concentrates on: the hierarchical view focuses on the file's path and name, while the numeric index focuses on the operating system's underlying implementation and numeric index.

7.1 Key Differences Between Hierarchical Name Space and Numeric Index Activity

The only real difference between the hierarchical name space view of the file system's activity and the numeric index view is in the area of file modifications and file deletions. The physical file blocks referenced by the i-node's numeric index are rarely modified, except in the case of e-mail files. In contrast, the hierarchical file names in the hierarchical name space are regularly modified. The result is that the number of deletions in the hierarchical name space drops off immensely while the number of modifications are much greater.

This relationship is shown in Figure 15. Because the number of files in the hierarchical view and the numeric index view are not the same—a single i-node can have multiple file names—Figure 15 is normalized. A value of 1.0 in Figure 15 denotes equivalence between the numeric index view and the hierarchical name space view of the file system's activity. A value greater than 1.0 shows that hierarchical name space view had proportionately more transactions than its numeric index counterpart. A value less than 1.0 shows proportionately less transactions in the hierarchical name space.

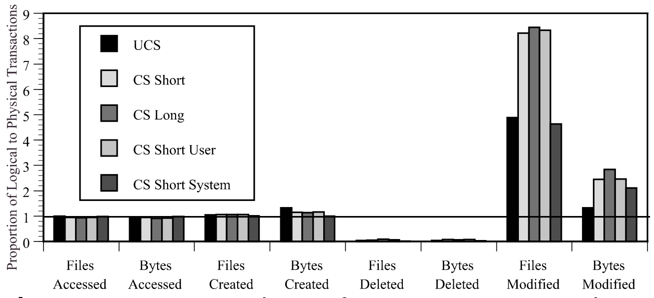

Figure 15. Proportion of average transactions, hierarchical name space to numeric index. A value of 1.0 denotes equivalence, greater than 1.0 shows more transactions in the hierarchical name space.

Figure 15 shows that the ratio between numeric index activity and hierarchical name space activity remained relatively constant for file accesses and file creations. On the other hand, file deletions are almost non-existent in the hierarchical name space view of the file system, in return for a corresponding abundance of modifications. Essentially, many numeric index deletions become hierarchical name space modifications. Because the number of modifications are greater in the hierarchical name space view, the distribution of file and byte transactions is also different from the numeric index view, as shown in Figure 16. (Figure 16 is the same data plotted in Figure 7, except Figure 16 uses the hierarchical name space view instead of Figure 7's numeric index view.) While these differences are significant, they are essentially the only distinctions between the hierarchical name space and the numeric index views of the same file system activity.

7.2 Similarities between the Hierarchical Name Space and the Numeric Index

Every other characteristic of file system activity discussed in Section 5 and Section 6 from the viewpoint of the numeric index is almost identical to the hierarchical name space perspective. The percentage of files used daily and during the entire trace are very close using either scheme. Similarly, the size distribution of files on the system, the size distribution of files modified, and the amount files increased or decreased when modified using the numeric index observations are nearly identical to the corresponding hierarchical name space observations.

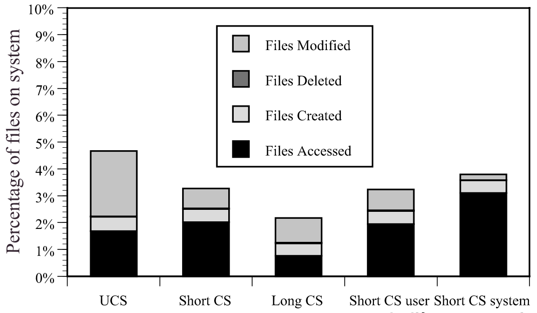

Figure 16a. Average percentage of files used daily. Hierarchical name space view only.

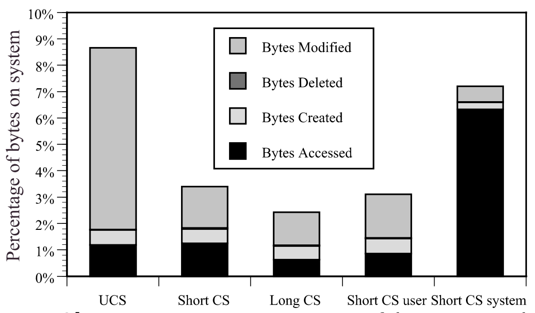

Figure 16b. Average percentage of bytes used daily. Hierarchical name space view only.

The periodic usage and reference locality observed in the numeric index is also present in the hierarchical name space, although the activity drop-off rate is slightly slower in the hierarchical name space view.

An excellent example of reference locality in the hierarchical name space is provided by the traces from the University of California system. These traces only collected the names of files (i.e., the hierarchical name space) which had been modified within the last 24 hours. As a result, their usefulness is limited. However, they can show how many files in the hierarchical name space are modified on consecutive days, as Figure 17 does. Obviously, an average of 100% of the files were modified 'today,' shown as Day 0 in the chart. However, only 42% were modified two days in a row. By the time a week has passed, the percentage of files modified on every one of the previous seven days is less than 20%. The number of files modified on consecutive days continues to drop, until only 3.8% of the files modified daily were modified every day of the last 30 days.

Reference locality can be assumed to be a general file system characteristic because:

- It has been shown to exist on both the numeric index and hierarchical name space views of the traces from UMBC and APG;
- It has been shown to exist in the University of California traces; and
- It has been shown to exist by earlier researchers (Jensen and Reed [7], Miller [11, 12, 13])

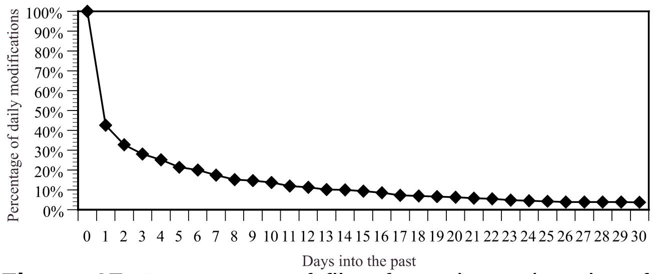

Figure 17. Percentage of files from the University of California traces, modified for N consecutive days in a row.

8 Summary

This paper has provided a comprehensive look at long-term file system activity. This type of data is useful to both file system designers and system administrators because it provides an insight to daily file system activity. While this type of analysis has been done before, it has not been done recently. The most recent research of this detail is Satyanarayanan's work in 1981 [18]. Clearly, the size and complexity of file systems has changed significantly since 1981, as well as the types of applications being used, so a more recent file system survey was needed.

The data provided by the long-term mode is of less interest to system administrators, but is very useful to file system designers and migration algorithm researchers. For example, the data on deletion rates and reference locality can be directly applied to file system design. A case in point is in deciding when to clean file segments in a Log-structured File System (LFS). Studies of long-term file system activity can significantly assist in this area.

To our knowledge, the differences between the hierarchical name space and the numeric index views of file system activity have never been studied. Previous researchers (Baker [2], Burns [3], Kumar [8], Miller [11, 12, 13], Mummert [14], Ousterhout [15], Shirriff [19], and Strange [24]) examined file system activity at the hierarchical name space level, despite the fact that many of them also collected the numeric index information. Occasionally, researchers have used existing data and focused on the physical components. For example, Gribble, et. al, [5, 6] used Ousterhout's [15] and Baker's [2] existing file system traces to check for self-similarity in file system activity at the read/write level. Yet, a specific examination comparing and contrasting numeric index and hierarchical name space activity has not been done before.

We have also presented evidence that shows hierarchical name space and numeric index activity are similar in many respects. For example, both possess locality of reference, and the same basic activity patterns. These basic activity patterns include:

- Deleting files which are used at a lower rate than files which are unused;
- Periodic file usage; and
- A high proportion of 'used' files either lapsing into disuse or being deleted.

While hierarchical name space files share these characteristics with the numeric index view of the same activity, the hierarchical name space view has a much lower deletion rate and a higher modification rate. This has an impact on file migration algorithms. The majority of the numeric index activity is file accesses, creations, and deletions. Because a file migration algorithm has no control over file creations and file deletions, a file migration algorithm working with the numeric indexes has to contend primarily with file accesses. In contrast, migration algorithms being used with the hierarchical name space (*i.e.*, the hierarchical file names) must deal with the same file accesses and many more file modifications.

Most programs access the underlying file system through the Unix operating system's hierarchical file name interface. As a result, the hierarchical file system provides researchers a better idea of user and program activity. However, the numeric index view of the file system is better for examining what the operating system does internally, how it manages physical devices, and how it stores data.

Modeling with the hierarchical file names is more intuitive because people usually think of files as names, not disk blocks and numeric indexes. On the other hand, the operating system manages the file system with the numeric indexes found in the i-nodes, and not with the file names.

9. Future Research

Using the findings from this research, we have been able to develop a tertiary storage migration algorithm that "ages" files when they are not used [4, 22]. This technique works an order of magnitude better than the current "best" algorithm, space-time [20, 21, 24].

Our future research includes: applying this software and data collection technique to Windows NT, in order to provide a comparison between NT file system activity and Unix; and developing an automatic system with the new technique to either compress unused files or move these files to integrated tertiary storage.

References

[1] Maurice J. Bach*, The Design of the Unix Operating System*, Prentice Hall, Englewood Cliffs, NJ 1990.

[2] Mary G. Baker, John H. Hartmon, Michael Kupfer, Ken W, Shirriff, and John K. Ousterhout, “Measurement of a Distributed File System,” Operating System Review 25(5), *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, 1991, pp. 198-212.

[3] Randal C. Burns, Darrell D. E. Long, “Efficient Distributed Backup with Delta Compression,” *Proceedings of the Fifth Workshop on I/O in Parallel and Distributed Systems*, ACM: San Jose, Nov. 1997, pp. 26-36.

[4] Timothy J. Gibson, *Long-term File System Activity and the Efficacy of Automatic File Migration*, Doctoral Dissertation, University of Maryland, Baltimore County, May 1998.

[5] Steven D. Gribble, Gurmeet Singh Manku, Eric A. Brewer, Timothy J. Gibson and Ethan L. Miller, “Self-Similarity in File-System Traffic,” *Proceedings of ACM SIGMETRICS '98*, Madison, Wisconsin, June 1998, pp. 141-150.

[6] Steven D. Gribble, Gurmeet Singh Manku, and Eric A. Brewer, “Self-similarity in File-systems: Measurements and Applications,” Unpublished Paper, Department of Computer Science, University of California, Berkeley, 1996.

[7] David W. Jensen and Daniel A. Reed, “File Archive Activity in a Supercomputer Environment.” *Technical Report UIUCDCS-R-91-1672*, Department of Computer Science, University of Illinois, Urbana, IL.

[8] Puneet Kumar and M. Satyanarayanan, “Log-based directory resolution in the Coda file system,” *Proceedings of the Second International Conference on Parallel and Distributed Computing*, January 1993, pp. 202–213.

[9] Samuel J. Leffler, *et al.*, *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison Wesley, Reading Massachusetts, 1990.

[10] John Merrill and Eric Thanhardt, “Early Experience with Mass Storage on a UNIX-Based Supercomputer,” *Tenth IEEE Symposium on Mass Storage Systems*, Monterey, CA 1990, pp. 117-121.

[11] Ethan L. Miller and Randy H. Katz, “An Analysis of File Migration in a UNIX Supercomputing Environment,” *USENIX Winter 1993 Conference*, San Diego, CA, January 1993, pp. 421-434.

[12] Ethan L. Miller and Randy H. Katz, “Analyzing the I/O Behavior of Supercomputer Applications,” *Eleventh IEEE Symposium on Mass Storage Systems*, Monterey, CA 1991, pp. 51-55.

[13] Ethan L. Miller and Randy H. Katz, “Input/Output Behavior of Supercomputing Applications,” *Proceedings of Supercomputing ‘91*, Albuquerque, NM, 1991, pp. 567-577.

[14] L. Mummert and M. Satyanarayanan, “Long-term Distributed File Reference Tracing: Implementation and Experience,” *Software–Practice and Experience, Volume 26(6)*, June 1996, pp. 705-736.

[15] John K. Ousterhout, Herve Da Costa, David Harrison, John Kunze, Mike Kupfer, and James Thompson, “A Trace-Driven Analysis of the UNIX 4.2 BSD File System.” *Operating System Review 19(5), Proceedings of the 10th ACM Symposium on Operating Systems Principles*, 1985, pp. 15-24

[16] R. L. Rivest, “The MD5 Message Digest Algorithm,” RFC 1320, April 1992.

[17] Mendel Rosenblum and John K. Ousterhout, “The Design and Implementation of a Log-Structured File System,” *Operating System Review 25(5), Proceedings of the 13th ACM Symposium on Operating Systems Principles*, 1991, pp. 1-15.

[18] M. Satyanarayanan, “A Study of File sizes and Functional Lifetimes,” *Proceedings of the 8th Symposium on Operating systems Principles, Association of Computing Machinery*, 1981, pp. 96-108.

[19] Ken W. Shirriff and John K. Ousterhout, “A Trace-Driven Analysis of Name and Attribute Caching in a Distributed System,” *USENIX Winter 1992 Conference*, San Francisco, CA, January 1993, pp. 315–332.

[20] Alan Jay Smith, “Analysis of long term file reference patterns for application to file migration algorithms.” *IEEE Transactions on Software Engineering SE-7(4)*, 1981, pp. 403-417.

[21] Alan Jay Smith, “Long term file migration: development and evaluation of algorithms.” *Communications of the ACM 24(8)*, 1981, pp. 521-532.

[22] Keith A. Smith and Margo I. Seltzer, “File System Aging—Increasing Relevance of File System Benchmarks,” *Proceedings of the 1997 SIGMETRICS Conference*, June 1997, Seattle, WA, pp. 203-213.

[23] Mirjana Spasojevic and M. Satyanarayanan, “An Empirical Study of a Wide-Area Distributed File System,” *ACM Transactions on Computer Systems 14(2)*, May 1996, pp. 200-222.

[24] Stephen Strange, “Analysis of Long-Term UNIX File Access Patterns for Application to Automatic File Migration Strategies,” *Technical Report UCB/CSD-92-700*, Computer Science Division (EECS), University of California, Berkeley, California, 1992.

[25] Trevor Blackwell, Jeffrey Harris, and Margo Seltzer, “Heuristic Cleaning Algorithms in Log-Structured File Systems,” *Proceedings of the 1995 USENIX Technical Conference*, Berkeley, CA, Jan 1995, pp. 277–287.