# Long-Term File Activity Patterns in a UNIX Workstation Environment

**Timothy J. Gibson and Ethan L. Miller [1]**
Department of Computer Science and Electrical Engineering
University of Maryland - Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
{tgibso2, elm}@csee.umbc.edu
Tel: +1 410 455-3972
Fax: +1 410 455-3969

**Abstract:** As mass storage technology becomes more affordable for sites smaller than supercomputer centers, understanding their file access patterns becomes crucial for developing systems to store rarely used data on tertiary storage devices such as tapes and optical disks. This paper presents a new way to collect and analyze file system statistics for UNIX-based file systems. The collection system runs in user-space and requires no modification of the operating system kernel. The statistics package provides details about file system operations at the file level: creations, deletions, modifications, etc.

The paper analyzes four months of file system activity on a university file system. The results confirm previously published results gathered from supercomputer file systems, but differ in several important areas. Files in this study were considerably smaller than those at supercomputer centers, and they were accessed less frequently. Additionally, the long-term creation rate on workstation file systems is sufficiently low so that all data more than a day old could be cheaply saved on a mass storage device, allowing the integration of time travel into every file system.

## 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 storage system. This problem has long been an issue for supercomputer centers which have always had to manage huge quantities of data in large files, but the recent advances in CPU performance have brought traditional supercomputer power to the desktop. With greater power comes more and larger files. System designers must insure that workstation file systems can keep up with the increased bandwidth that this change requires.

During the last decade, CPU speed has increased approximately fifty-fold, and semiconductor memory capacities and speeds have also improved significantly. Typical workstation disk storage capacities have increased from tens of megabytes to several gigabytes in the same time period. However, disk access speeds have only improved by a factor of three [2]. The result is disks storing more and more data — being accessed more frequently by ever faster processors — but with disk access and transfer rates only slightly better than

they were ten years ago. Current computers incorporate several semi-conductor cache levels and use complicated caching policies to overcome this disk bottleneck [2, 3]. At the same time, file systems are being developed to make tape migration transparent to users [4], and near-line tape robots continue to drop in price. Soon, it will be possible for smaller computing centers[1] to use on-line and near-line tape systems — or some other high capacity medium — to store files which are seldom used. Consequently, long-term disk access patterns and storage requirements will affect more and more users, making the study of these patterns increasingly important.

To study disk activity, we developed a system which allows administrators to collect file system traces at the user-level without modifying the operating system kernel. Additionally, the collection system can be run without tracing activity appearing in the traces. This means that data can be gathered without the collection process directly influencing the experiment.

This paper is organized into four sections. We begin by reviewing previous work by other authors. Monitoring disk activity is not a new activity and was first done in the early 1980's. Next, we discuss the data collection tools. These tools differ significantly from previous tools used in earlier studies. The tools track file system activity at the inode and logical filename level, providing detailed activity logs of file creations, deletion, modifications, etc. in a manner different from earlier studies. After discussing the tools, we analyze several months worth of data the tools provided. This analysis provides some interesting conclusions that reinforce earlier conclusions about file activity, and provide new insights about file modifications. Finally, we discuss the direction of out future research, which is centered on using the trace data we are collecting to drive a long-term storage and migration simulator.

## 2. Previous Work

Researchers have looked at disk or tape storage systems in the past. However, the nature of computing changes continually, so storage systems need periodic, if not constant, study. In the 1980's, both Smith [5], and Ousterhout, *et. al.* [6] 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 so common on systems 15 years ago. Ousterhout's very detailed file traces were conducted over three to four day periods at a fairly low level. 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 [7] update Ousterhout's work, concentrate on physical disk activity, and have the same short trace periods as Ousterhout's traces.

Other studies, [8, 9, 10, 11] are directly applicable to supercomputing centers, but may not apply well to smaller commercial and academic computing centers. Both the size and number of files at supercomputing centers far exceeds "normal" computing activities; more importantly, supercomputing centers usually have large tape libraries with tape

robots providing near-line storage for hundreds of terabytes of data files. In contrast, most computing centers do not have tape robots and only use tape for archiving purposes.

Our work most closely resembles Strange's disk studies from 1992 [12]. We collect much of the same information as Strange, and in fact corroborate a good number of his findings. However, the traces used for this paper cover twice the time period as Strange's traces and provide additional information that Strange's did not.

We propose to study long-term file access and file usage patterns at both supercomputing facilities and on "normal" file systems — expanding the work done by previous authors. We begin our study of normal systems by examining workstation computing clusters and updating the state of research on disk storage systems in a networked environment.

## 3. Tools

The most basic part of the system is a modified GNU `find` utility we call `ftrace`. This program collects information on all the files in a file system or directory. The information includes the file's inode number, size, name, access time, inode create time, modification time, owner, and group. The file's name can be collected with or without the path, and either scrambled with MD-5 for privacy purposes or represented as plain-text. For example, `/JNsBhKWReJ4/9o9JCYa7VFY/2IzA6T74FOM` is an MD-5 hash of the plain-text `/ftrace/lib/Makefile`. Using MD-5 to scramble full pathnames allows us to protect privacy and anonymity on public systems while still studying file clustering and access patterns.

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. Because this information is collected on every file on the selected file system, a single trace can be quite large. For example, a trace of a file system with 10,000 files using full pathnames and MD-5 scrambling requires approximately one Mbyte of disk space to store the trace data. However, by compressing the data, we can store traces in approximately 25% of the original space until they are needed. The tracing program scans 200-300 files per second, tracing 210,000 files in twelve to fifteen minutes.

We currently run this collection system *nightly* on four active file systems in the Computer Science and Electrical Engineering Department at the University of Maryland, Baltimore County.[2] The four file systems have a capacity of 7.5 GB and are used by the department's faculty and students and include many World-Wide–Web pages. The typical user on the system compiles programs, writes papers, searches the World-Wide–Web, and sends electronic mail. Additionally, most users have their own World-Wide–Web pages, so the system is also accessed by some off-campus users.

These file systems are mounted on a SGI Crimson machine with 208 Mbytes of RAM and approximately 560 users. This central fileserver machine is accessed via the Network File System (NFS) by approximately 150 other workstations or personal computers within the department, as well as many dial-in connections. There are 17 other file systems mounted on this SGI Crimson, with an additional 28 GB of storage. However, we chose the four used in this paper as representative of the others when we began collecting the trace data.

The second part of the system takes two trace files sorted by inode number, compares them, and generates a file containing only the differences, along with new file creations and old file deletions. This program, called `fdiff`, dramatically reduces the tracing system's long-term storage requirements. If only three files out of 10,000 are used in a twenty-four hour period, the difference file only has three lines — not 10,000 — one for each file that has changed. Additionally, a line in the difference file does not necessarily contain the same information as the original trace file — in fact, only when a file is created or has its name changed does `fdiff` keep all the trace information. Table 1 shows the information kept by `fdiff` for the different types of file transactions. By keeping the first day's trace as a baseline along with the difference files, an accurate model of file system activity can be built as all the subsequent transactions are retained.

By way of illustration, we developed a third program, `rebuild`, which takes a trace file and a difference file as input and uses these to "rebuild" the second trace file. For example, if two trace files from period A and period B generate a difference file C, `rebuild` uses trace file A and difference file C to construct trace file B. `Rebuild` serves two purposes. First, it verifies that `fdiff` works correctly and saves all pertinent information. `Rebuild` also reduces the number of trace files that must be kept on the system, thereby reducing the total collection system's storage requirements.

| File Activity | Statistics Kept by `fdiff` |
|---|---|
| Access | inode number, access and create times, size |
| Create | inode number, last access time, inode creation time, last modification time, size, userid, group id, file name |
| Deletion | inode number, estimated deletion time, size, name |
| Modification | inode number, last access time, inode creation time, last modification time, new size, difference between old size and new size |
| Change in Owner | inode number, inode create time, new owner number |
| Change in Group | inode number, inode create time, new group number |
| Name Add or Name Delete (file move) | inode number, last access time, inode creation time, last modification time, size, userid, group id, file name, estimated deletion time (if applicable) |
| Change in inode creation time only | inode number and inode create time. Keeping this information is necessary because some programs, notably `tar`, can post-date files so their "birth date" gets older between traces. |

Table 1. File statistics retained by the `fdiff` program.

As mentioned, `fdiff` uses an "estimated deletion time." We estimate the file's deletion time because all we know is that the file was present during the first trace and absent on the second — obviously it was deleted between the two traces. Exactly when the file was deleted is unknown. To make the estimated deletion times more accurate, `fdiff` distinguishes between file deletions in which the inode is reused by another file and deletions where the inode is not reused. If the inode is reused the deletion time is a randomly generated time within sixty minutes prior to the inode being reused by another file [13, 14]. On

the other hand, if an inode was not reused, all we know is that the file was deleted between the traces. In this case, a time is randomly chosen within the last 24 hours as the estimated deletion time. As a variation, `fdiff` can also place 90% of the estimated deletions between 9 AM and 4 PM — normal working hours.

The last part of our collection and analysis package is the statistics program. This program programs use the difference files to generate the statistics shown in Table 2. In addition to

| File Activity | Statistics Collected |
|---|---|
| Accesses, Creates | Total number of files, total number of bytes, average size, standard deviation (size), maximum size, minimum size. 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 2. Statistics collected by the analysis program.

the four programs, we use two Perl scripts in the system. The first runs the file tracing program nightly as a `cron` routine and maintains an activity log; the second automates running the differencing program on trace files.

Our statistics collection and analysis package for file systems has both strengths and weaknesses. Ousterhout [6] 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, so do the temporary files they create. Our system traces are collected nightly at 10 PM when few users are active, so we miss most of these temporary files that are created and deleted during the day. 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 never be moved to long-term storage and do not contribute to long-term growth.

The differencing program system also misses two other types of transaction. First, while the system detects either a change in a file's name or a modification to a file, if a file has both its name changed *and* is modified between traces, the system counts it as a file deletion and a new file creation. This is because the modified file only retains the same inode, owner, and group; everything else has changed. Our traces show that modifications occur less frequently than many other transactions, and that name changes almost never happen. As a result, we believe the combination of name changes and modifications happens infrequently. Similarly, the system does not notice how many times a file is accessed or modified — only that an access or modification occurred and when the most recent one happened. However, 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.

The collection system is very powerful despite these minor shortcomings. First, the system does not require any operating system kernel modification. As a result, it can be run in user-space, although the tracing program is run as `root` so that all files can be traced. The ability to trace either file systems or directories means that while the entire disk can be traced, it also allows active file systems to be checked more often if necessary. Conversely, file systems or directories that receive little use or that are unimportant can be ignored. The current system allows administrators to answer questions that often provide daily headaches, such as "how many files are really being used?" or "are we using more disk space because of larger files or more files?"

The collection and statistics package allows us to study many long-term disk activities. We can catalog how many files on a system are used, and how often and how much these files are used. When a file is modified the package monitors how much the file changed in size. The package does have some shortcomings. Ignoring the temporary files observed by Ousterhout and the exact number of file accesses or modifications per file make simulations and measurements of short-term usage impossible. However, we feel the lower system overhead and the ability to run the system as a user process without kernel modification outweighs these two points, particularly for studies of long-term file system behavior. Since the file traces we collect will run a mass-storage simulator, we do not care about short-term access patterns; files that exist for seconds or have multiple accesses within minutes have little affect on movement of data between disk and tape. Temporary files will never be migrated to tape, hence we do not need to track of them. Similarly, the number of times a file is accessed per day is less important from a storage migration point-of-view — what matters in this case is whether the file was used at all.

## 4. Running the Collection System

The collection system has run without difficulty since we began using it in October 1996. We currently trace four file systems with approximately 210,000 files. Tracing the four file systems takes twelve to fifteen minutes, averaging 200-300 files per second; it takes longer

if the system has an unusually heavy work-load. The size of a trace file varies with the number of files on the file system. However, all four trace files, with 210,000 file entries, can be compressed to 5 MB of total disk space. These traces are stored on a separate file system that is not traced.

The work of generating a difference file with `fdiff` requires the trace file to be uncompressed, sorted, and compared with another trace file. How long this takes depends on the size of the trace files and the processing machine's speed and memory – it typically takes between 30 and 60 seconds to process one day's trace from each file system – with the vast majority of the time spent uncompressing and sorting the original trace file. Processing four months worth of traces from four file systems takes about 3 hours on our SGI Crimson in a multi-user environment.

Finally, the `stats` program uses only the `fdiff` difference files, which are very small compared to the daily trace files. How long the `stats` program takes to run depends on how active the file systems are, how long a period is being examined, and the statistics being generated.

## 5. Results

This section has three sub-sections. The first presents some of our findings about the overall system activity. The second sub-section shows daily file system activity during the trace period. Finally, the last sub-section display our results about file activity by file size, and file modifications by file size.

### 5.1. Overall System Results

Our initial analysis corroborates what earlier studies found, namely that most files are not used very often. Figure 1 clearly shows that on a typical day, only 5,000 files — less than 3% — are used on a system with 210,000 files.

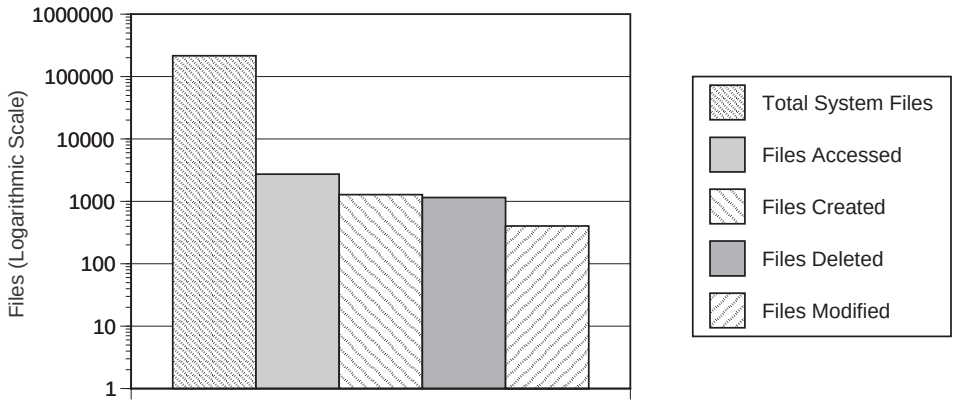


Figure 1. Average daily file usage by category (logarithmic scale).

Figure 2 shows the average daily number of files and bytes acted upon during the trace period. The left side of Figure 2 shows that more than twice as many files are accessed daily than are either created or deleted. It is interesting to note that file creations only slightly outpace file deletions. Finally, only 360 files were modified in any manner on the average day. The right half of Figure 2 shows the number of bytes used daily by the different file operations. Note that the relative ratio between files and bytes holds for file accesses, creations, and deletions. However, while the number of files modified is one-third the number of creations or deletions, the number of bytes modified exceeds the number of bytes for either of the other transactions.

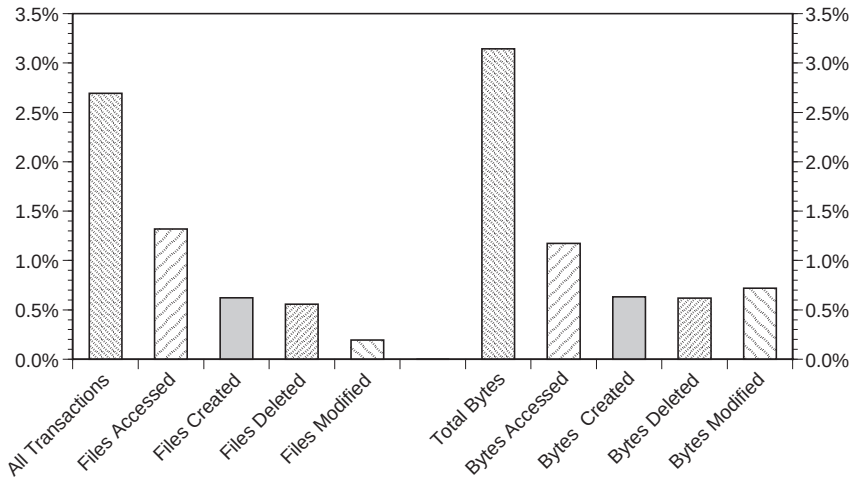


Figure 2. Average daily file system use.
Shown as the percentage of total files or bytes on the system. Note that the relative fractions of files and bytes in each category are comparable except for modifications.

Figure 3 is a breakdown of the average number of files and bytes modified daily – whether the files increased, decreased, or remained the same size is also shown. This chart leads to the conclusion that most modified files either grow or remain the same size, few files get smaller, and those that do decrease in size are fairly small to begin with. You may also conclude that modified files which remain the same size are generally small to begin with. Finally, the majority of the bytes from modifications are in the category of 'modified and increased' in size.

Figure 4 charts the amount of activity on the four different file systems. The file activity is on the left of each file system category and the bytes are on the right. The two most active file systems are `faculty` and `grad3`. An examination of Figure 4 shows that for these two most active file systems, the ratio between file modifications and other actions changes when bytes are used instead of files. This leads to the conclusion that the average files being modified on `grad3` and `faculty` must be larger than on the other two systems. Finally, the `faculty3` file system is not very dynamic; it grew to 99% capacity during the trace period and a study of its daily logs shows a continual drop in all file activ-

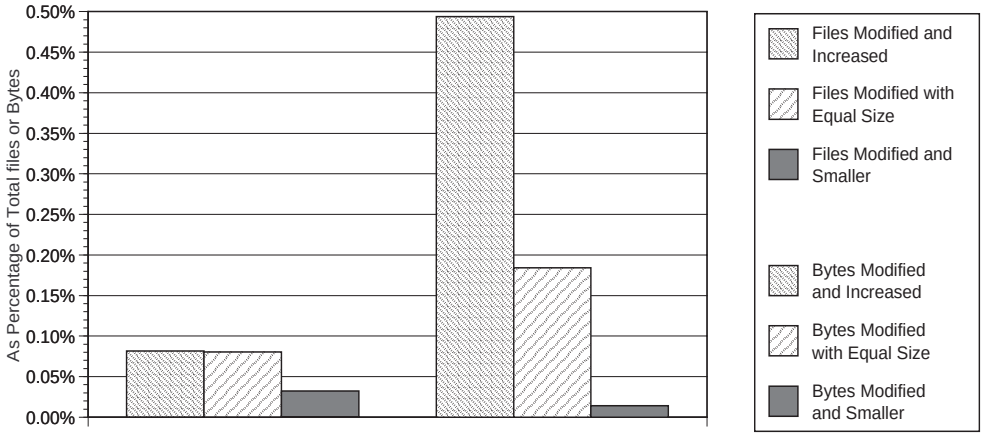


Figure 3. Average daily modifications, grouped by type of modification.
Shown as the percentage of total files or bytes on the system.

ity as it approaches capacity. This activity drop occurs because professors are pack-rats. When a faculty account fills up, the owner simply requests more space on other file systems. On a long-term storage system with automatic migration, these accounts could easily be moved to tape.

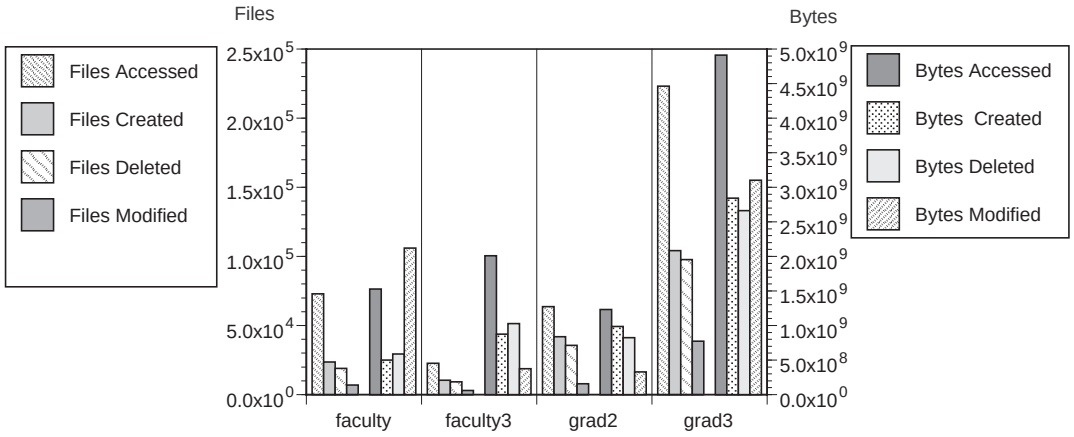


Figure 4. Total activity for the four file systems.

## 5.2. Daily Statistics

Daily statistics are also available, and some of these are shown in Figures 5 - 7. The time period for these three figures is a 141 day period from 23 October 1996 to 12 March 1997. Every Saturday is shown by a vertical bar in these figures and there is usually a corresponding drop in activity.

A plot of either the number of files or the number of bytes for accesses, creations, deletions, or modifications corroborates many of the conclusions from Figures 2-4. Specifically, that most file transactions are file accesses, that creations slightly outpace deletions, and that file modifications lag behind the other categories in the number of file transactions. This data is not plotted in this paper because the number of data points makes the graph very difficult to read.

By adding together the byte increases from file creations and file modification increases and comparing these numbers with the byte decreases from file deletions and file modification decreases, you can visually see the “byte creep” on the four file systems. Figure 5 shows this long term file system growth. The cumulative number of bytes on the system in Figure 5 appears to start out below zero because approximately 260 MB of data was deleted on the first two days that traces were collected. Nonetheless, file creations overcome this deficit within a month; by the time two months had passed, the system picked up nearly 500 MB from where it was on the second day. A drop occurs in early February, the beginning of a new semester. This February drop is from students and faculty deleting the previous semester’s files and doing general ‘house-cleaning.’

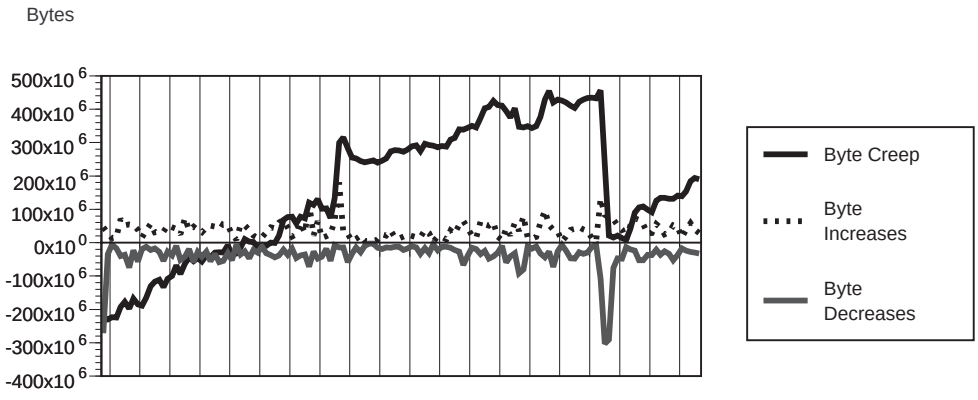


Figure 5. Cumulative daily file system increases.
Plot for a 141 day period from 23 October 1996 to 12 March 1997—vertical bar denotes Saturday.

Although not charted, a comparison of the number of bytes from file creations and deletions with the byte amounts added and removed by file modifications reveals that modifications make very little change in the amount of space on the file systems. Our traces show that modifications usually account for less than 5% of a file system’s growth or contraction. The vast majority of change in both the number of files and the number of bytes on the file system comes from file creations and deletions.

Finally, Figure 6 shows the increase in the overall number of bytes on the system. The lines in Figure 6 are fairly flat because the number of bytes added is small compared to the number of bytes already on the system. The large spikes up in week 8 and down in week 18 do appear as small “bumps” in the ‘Total’ line, because many of the bytes added and deleted during the spikes were larger than normal. A plot of the number of files on the system shows less change because most of the files on the system were never used.

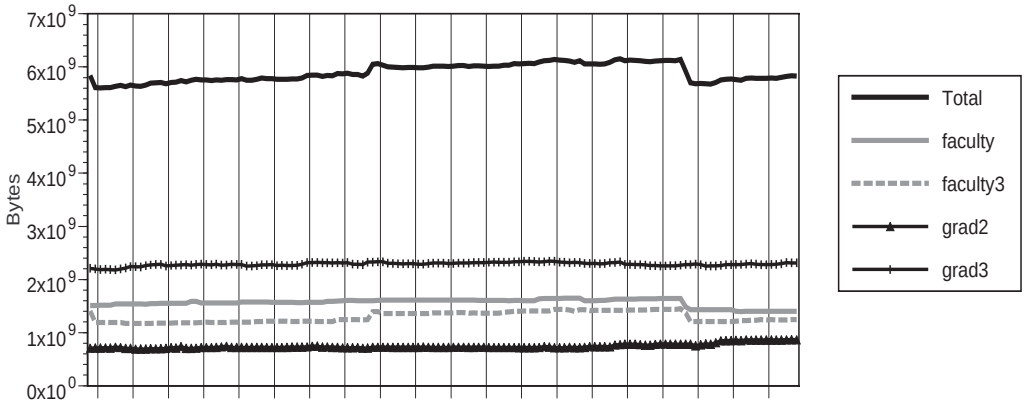


Figure 6. Total bytes on the system during the observation period.
Plot for a 141 day period from 23 October 1996 to 12 March 1997—vertical bar denotes Saturday.

However, plotting the total amount of activity on each file system provides more variance. Figure 7 is a plot of all transactions on the file systems without regard to size. The `grad3` file system is the clearly the most active — proving that graduate students do most of the work at a university. The most active days are in the middle of the week, and weekends show a significant drop in activity.

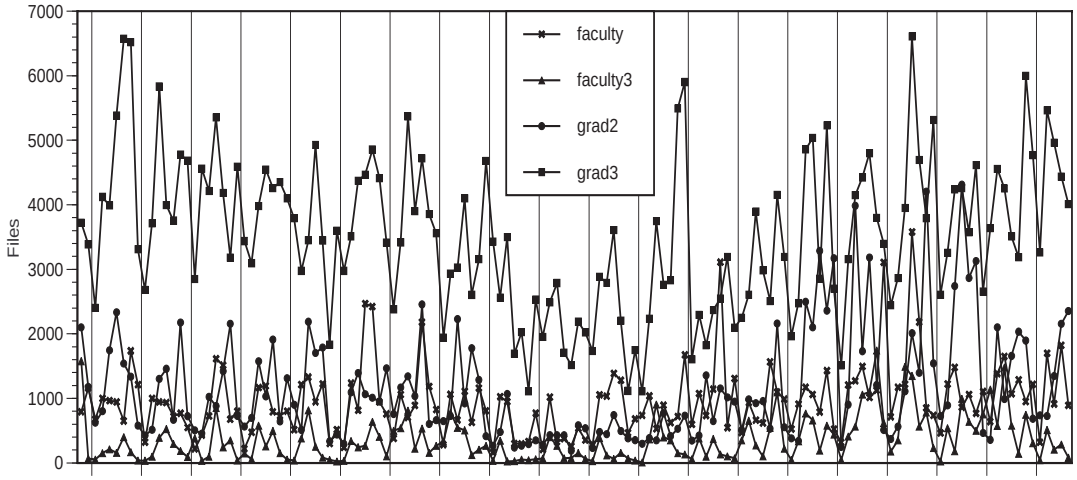


Figure 7. Total system activity — all categories.
Plot for a 141 day period from 23 October 1996 to 12 March 1997—vertical bar denotes Saturday.

### 5.3. Activity by file size

The statistics package also provides information on the file size the operations take place upon. For example, Figure 8 shows the breakdown by file size of all the files on the four file systems at the beginning of the trace collection process. This figure illustrates that on our file systems, the preponderance of the files are small, generally less than 8 KB, with the “less than 1 KB” category having the largest number of files.

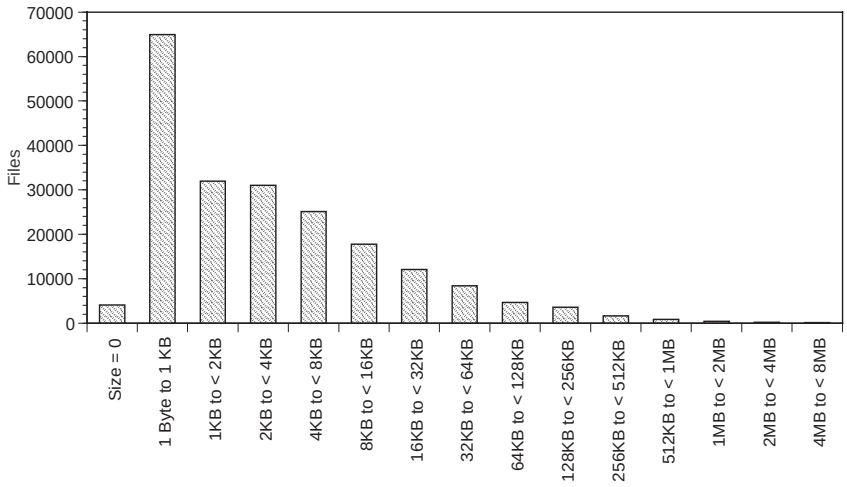


Figure 8. Total number of files on the system by size at the beginning of the trace collection

Figure 9 shows the cumulative percentage of transactions by file size. Again, most transactions occur on fairly small files of less than 8 KB. In most cases, 90% of all transactions are done on files of less than 32 KB. The only transaction category that lags behind the others is modifications, where 25% of the modifications occur on files that are larger than 64 KB.

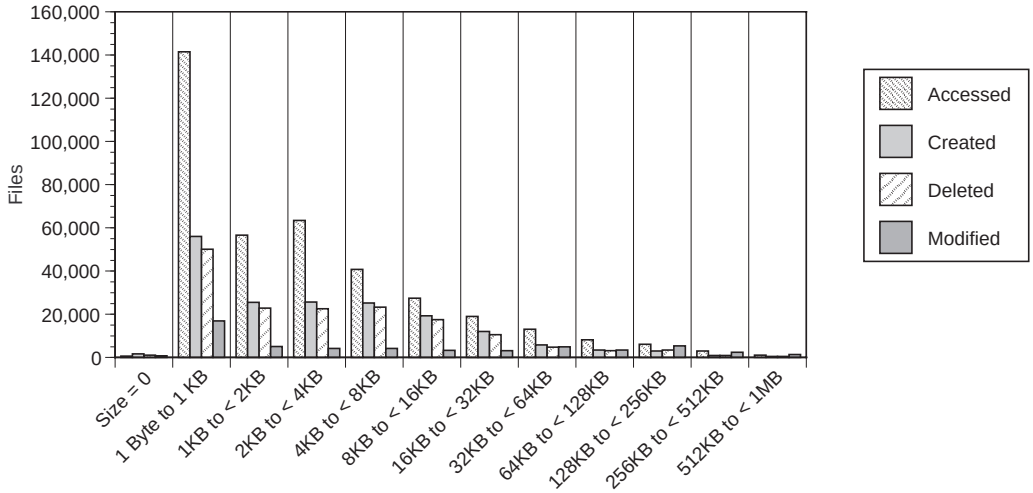


Figure 9. Total Transactions by file size.

Finally, one of the more interesting results comes from comparing the amount a file increases when it is modified with the file's size. Figure 10 shows this information. As expected, most of the files modified are small files. However, the most striking item about this figure is that the amount a file increases as it grows is very small, regardless of the

file's actual size. Most files increase in size by no more than 1 KB; in fact, file increases of more than 8 KB are rare, regardless of the file's overall size. A similar chart can be made from plotting the file decreases with the file's size. The only difference is that the majority of the files that are modified and decrease in size is even smaller than those that increase.

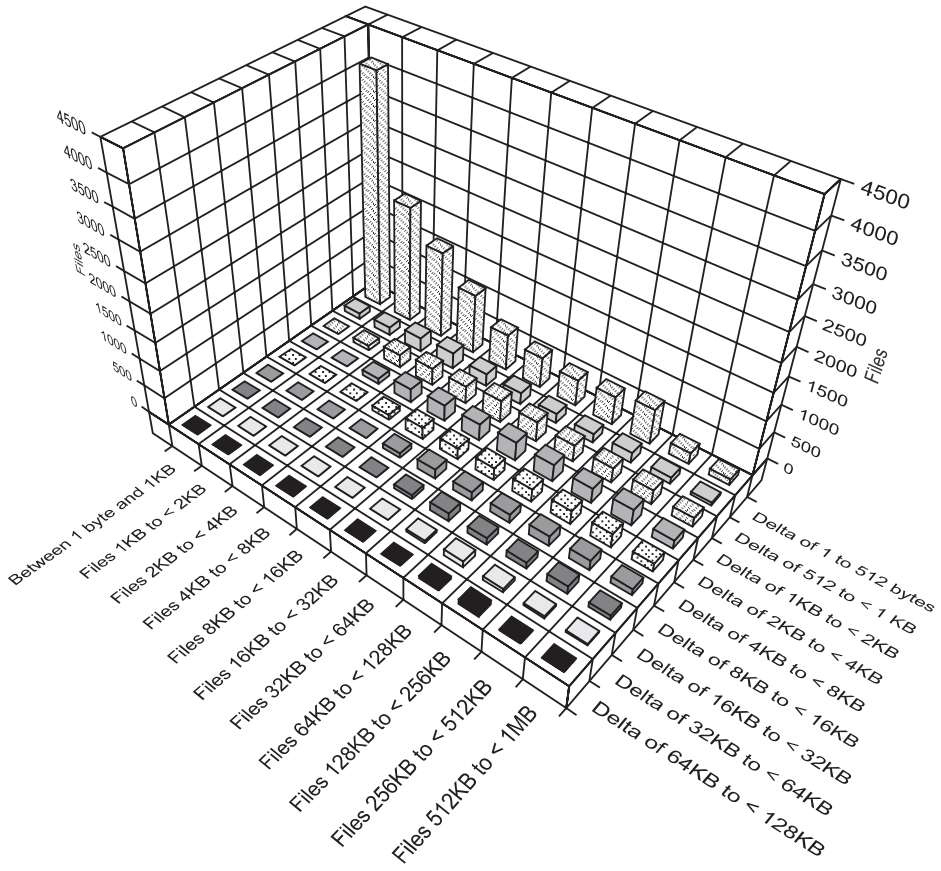


Figure 10. Number of file modification increases cross-referenced with the modified file's size

By way of illustration, Figure 11 is a graph of the file modification increases for large files only, shown as a cumulative percentage. There are six file size categories between 128 KB and 8 MB. Regardless of the category, 90% to 95% of all file modifications add less than 64 KB to a file, 80% add less than 32 KB.

## 6. Observations

Our initial work allows us to make some observations about file system growth and activity. Some of these observations corroborate earlier research, while others are new. First, file accesses occur more often than any other type of file transaction. An obvious implication is file accesses should work efficiently. For a mass storage system with nearline tape storage, this means that files which are likely to be accessed should be on the disk system,

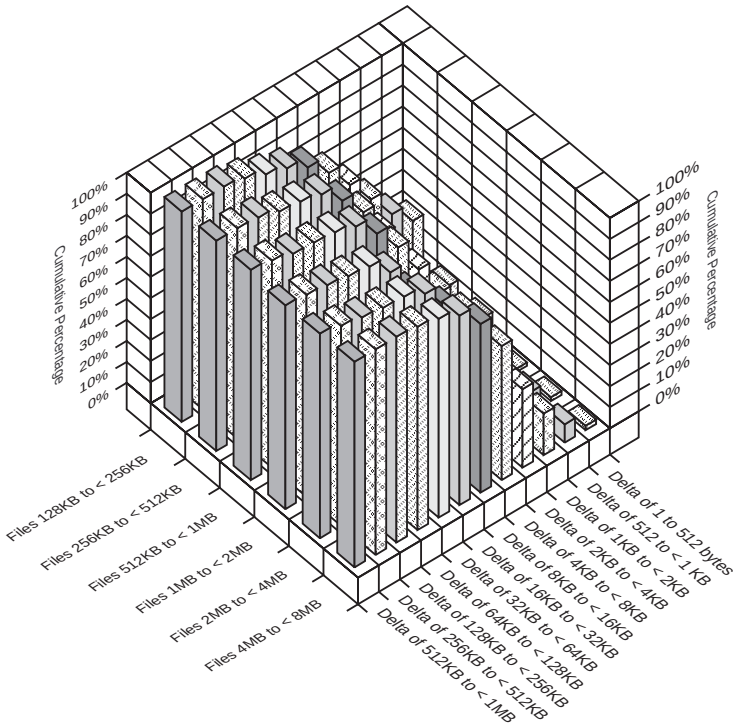


Figure 11. File modification increases for large files (shown as cumulative percentages).

and if a file is on the nearline tape it should still be quickly accessible. Another observation in keeping with previous work is that most user files on a system are small, generally less than 4 KB.

Next, a great number of files are created and deleted every day. At least some of these creations and deletions come from the fact that many text and image editors make modifications by erasing the old file and creating a new one with the same name. In essence, compilers do the same thing. However, this should not be counted as a file modification. From the file system's viewpoint, it truly is a file creation and a file deletion, and not a file modification. At first glance, it may seem wiser to treat these transactions as a file modifications from a long-term storage point-of-view. Unfortunately, the long-term storage tape may not have room to let a file grow by 4 KB.

A more telling observation is that comparatively few files are modified, and if they are modified, the amount added to the file is very small — regardless of the file's overall size. This is a new concept for file system designers. It is generally assumed that if a large file needs to grow, the amount of new disk space allocated for growth should be proportionate to the original file's size. However, this does not appear to be true. Our initial work shows that most modifications result in a very small file increase, regardless of the file's original size.

Given these observations, it is reasonable to assume that a file is more likely to be deleted than modified; and if it is modified, a file will only increase by a small amount. Most

importantly, means that file accesses should be the most efficient operations. The efficiency for file creations and file deletions follow behind file accesses. Finally, file modifications do not have to be extremely fast or efficient operations because they rarely occur.

In keeping with the observation that file modifications do not need to be extremely efficient, a file system that allocates more than one disk block at a time to a growing file only makes work for itself. For example, assume the following: A file system with 512 byte blocks has a file larger than 256 KB that needs to increase in size. This type of modification happens fairy rarely; in our traces of 838,000 total file transactions there are only 23,619 modifications resulting in a file increase. Of these, only 3,278 – 0.39% of all transactions and 13.87% of all modifications with increases – fall into the category of a file larger than 256 KB needing to grow. Of these 3,278 modifications, 92.3% increase by less than 32 KB. In fact, more than half of the files that are larger than 256 KB increase by less than 4 KB, or just eight 512 byte blocks. in order to meet the file's size increase, the file system will need usually only need to allocate eight 512 byte blocks. If a file system automatically allocates more than eight blocks based on the file's original size, these excess blocks have to be reclaimed later by the operating system. Automatically allocating disk blocks or file extents based on file size becomes even more problematic with larger files — 85% of all files larger than 1 MB increase by 32 KB or less.

Files that grow in size happen rarely enough that allocating blocks as needed will not usually make much difference to users. However, it makes file system design easier because it reduces fragmentation and 'garbage collection.'

## 7. Future Work

We are continuing to work on the file tracing system and are expanding both the number and types of file systems we collect trace data upon. An important note for system administrators is that a computer system does not need to run `ftrace` to use either the differencing or the statistics programs. Data can come from other tracing programs or from system logs, as long as the data is in the correct format. We are actively working with a super-computing project that keeps detailed logs of the file accesses, there we will only need to modify the format of the log file.

The next step of the project is to develop a mass storage system simulator which we will then use to model different file migration strategies. The first part of the simulator will generate better statistical information than we currently have. For example, we want to be able to accurately display inter-reference periods and the number of times and ways a file is used.

## 8. Conclusions

We have developed a comprehensive set of tools that allow system administrators to monitor long-term file system activity and growth. We do this by collecting periodic traces, as unobtrusively as possible. The traces can then be analyzed for different types of activity. The file differencing program dramatically reduces the amount of work the analysis programs must do. While the differencing program must sort every trace file, the resulting dif-

ference file is *much* smaller than the parent trace file. This is particularly important when long-term statistics are being stored.

In this paper, we have shown that most long-term file system activity is caused by file accesses, with file creations and file deletions lagging behind. File modifications account for little of the file system's activity.

Finally, we have illustrated long-term growth on the file system, and accounted for the source of this growth. File systems grow in size because of file creations, not file modifications. File modifications do not normally increase the size of the file by more than 32 KB. The fact that file modifications do not increase (or decrease) a file's size by a large amount may allow file system designers to change the way file blocks are allocated.

[1] A computing center has 30 or more workstations, with 30 GB or more of disk storage. Obviously, the larger the computing center, the more cost effective storing seldom used files on tape becomes. Similarly, what people using the computing center do also influences this decision. For example, a business could easily move old financial accounts to tape, knowing they were accessible within a few minutes.

[2] This was true when this paper written (May 1997). As of October, 1997, we collect data from the UMBC Computer Science Department (8 file systems with 28 GB of storage), the UMBC Computing Services cluster (9 file systems with 35 GB), a DoD installation (8 file systems with 4 GB), and a supercomputing complex more than fifty terabytes.

---

**References**


[1] This work was supported by the NASA Ames Research Center under grant NAG 2-1094.

[2] David A. Patterson and John L. Hennessy, *Computer Architecture: A Quantitative Approach* (Morgan-Kaufman Publishers, San Francisco, CA, 1996)

[3] R. Hugo Patterson, et al, "Informed Prefetching and Caching," Operating System Review 29(5), *Proceedings of the 15th ACM Symposium on Operating Systems Principles* (1995) pp. 79-95.

[4] 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." *USENIX Winter 1989 Conference* (San Diego, CA, January 1989) pp. 65-84.

[5] 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.

[6] 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.

[7] 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.

[8] 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 (1991)

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

[10] 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.

[11] 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.

[12] 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).

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

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