Characterizing Long Term Usage of a Mass Storage System At a Super Computer Site

Joshua C. Neil
Los Alamos National Laboratory
Los Alamos, NM 87545
jneil@lanl.gov

Abstract

One of the primary mass storage systems in use at the Los Alamos National Laboratory (LANL) is the Common File System, or CFS. CFS went into production in 1979, servicing supercomputer environments, and later was expanded for use with a broader networked workstation environment. It is now used by a very large user population at LANL. It can be used by any employee for storage purposes, and is used by all of the large supercomputers at LANL. CFS is being phased out for the supercomputing environment due to the need for a more scalable mass storage system design. To our benefit, records have been kept for the last seven years of all activity on CFS. A statistical analysis of these records has been performed, to understand how the mass storage system was used over a long period of time. Example usage statistics include maximum and average file sizes, data rates, and bytes moved for each month. Trends and observations about these usage statistics will be presented.

The paper will also present some study in the effects of environmental changes and their implications for CFS. An example of an environmental change question is: how does new media technology affect the usage or management of the system?

Study of the performance of the storage system over this long period of time will also be presented. Characterization of performance and how migration, environmental factors, and usage-affected data rate performance as well as time-to-first-byte performance is examined.

Some conclusions about usage and its effect on planning, design, and operation of mass storage systems will be discussed. It is hoped that the analysis of actual long run usage data of a mass storage system in a demanding supercomputing environment will provide interesting lessons that can be applied to the planning, design, and operation of future mass storage systems.

1 Introduction

At Los Alamos National Laboratory, one of the mass storage systems in use is the Common File System (CFS). CFS was developed in the late seventies to provide a centralized file server capability for a large heterogeneous computing network running a variety of operating systems (e.g. CTSS, NOS, MVS, VMS, UNIX) [1]. For the past seven years, a log has been kept on the usage of CFS. This paper presents a statistical analysis of these seven years. Important topics include statistics on the usage of the system, environmental change and its effect on usage or management, the effects of migration, environmental changes, and usage on performance, and some comments on future planning.

2 Usage

Usage is a very important aspect of this analysis. The performance of CFS is well known at least by the Los Alamos community, and therefore not the topic of this paper. But how CFS was actually used is less well known. Is CFS being used to its fullest extent, and if so, how often, and for what type of files?

The maximum data rate of CFS is 3.8 Mbytes/sec for a single file. This rate is rarely achieved, and appears to depend almost entirely on the client's performance limitations and usage models. In figure 1, it is clear that the average data rate is far below the maximum rate possible, and even the monthly maximum rate actually achieved is generally below the performance limit of CFS.

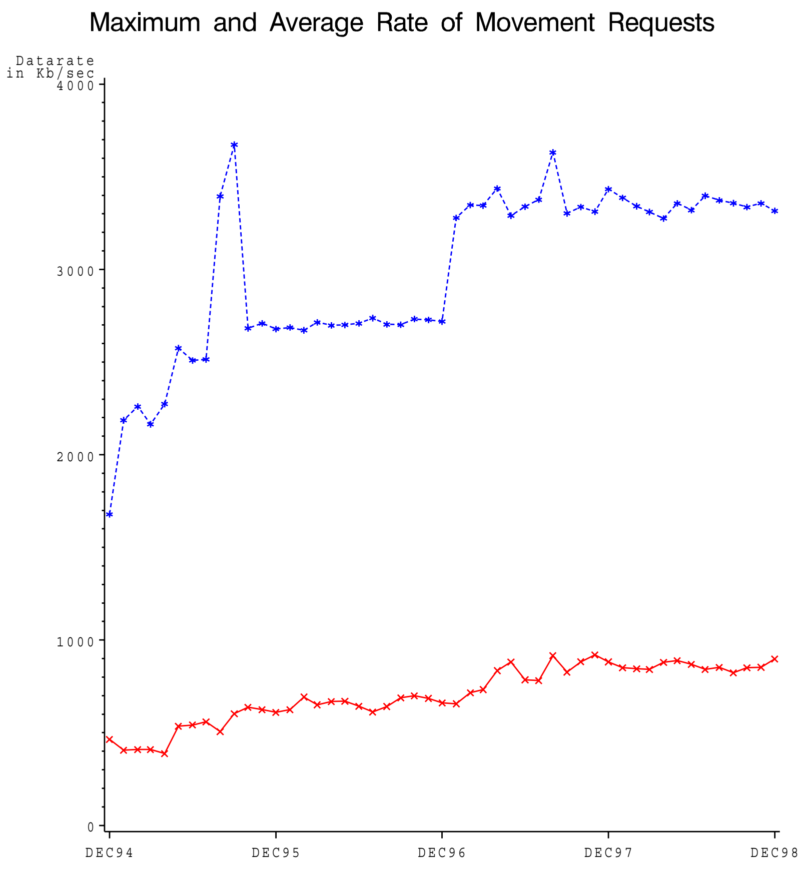

Figure 1

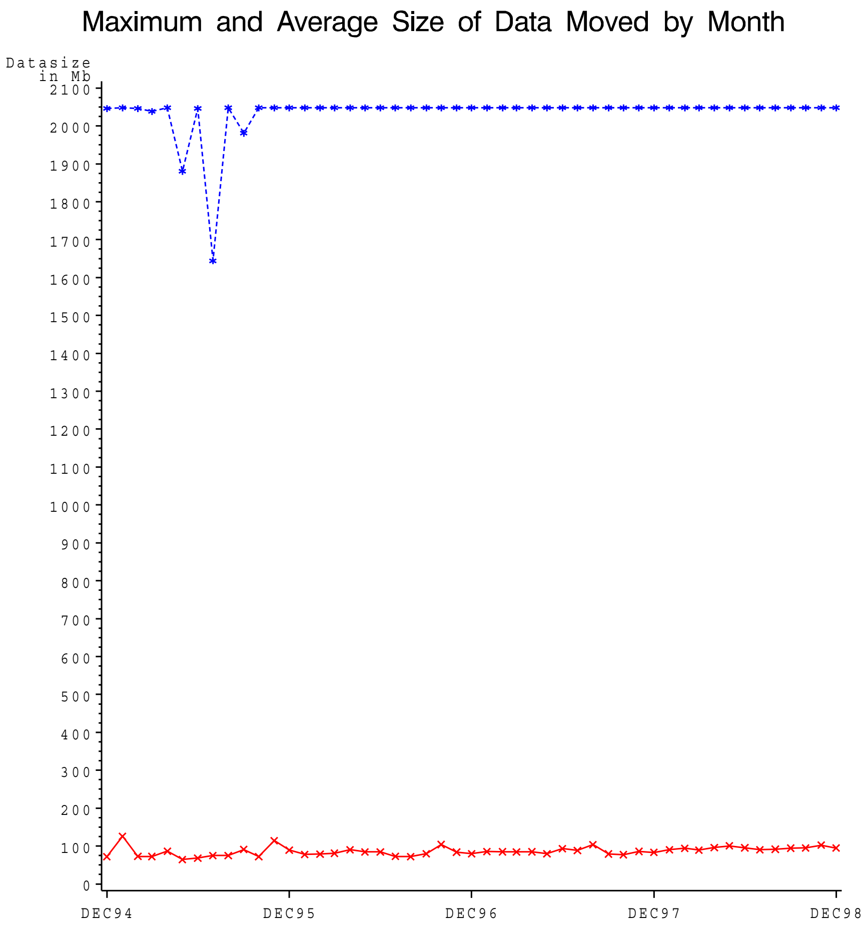

Figure 2

As seen in figure 2, almost all files are much smaller than the maximum file size (around 2.05 Gb). Generally, only large files (.5 to 2 Gb) achieve the maximum data rate due to file transfer startup, client efficiency, and network issues. With these two graphs in mind, it is clear that the system does not achieve its maximum data rates most of the time, since it is primarily being used for small files.

An interesting feature of data rates is the rates corresponding to different request types. As seen in figure 3, gets (data transfer out) achieved the highest data rates. Many clients when writing the file, run it through a pipe. For example,

```
tar -cvf - . | cfs store -:file
```

This is very slow, as it has to run through a pipe on the client's operating system. Many operating systems buffer at 512 bytes, so tar writes 512 bytes to a buffer, then a context switch occurs and CFS (the client program) writes the 512 bytes on the network destined for CFS, and then context switches back to the tar program and the cycle repeats. This is extremely inefficient. It is often done so that clients do not have to have enough disk space to first do the tar and then do the CFS store. This accounts for the slowness of saves/replaces. Gets are not generally piped in the same way.

We can also see that saves are generally performed at faster rates than replaces. This is due to the fact that many backup usage patterns exist where the files are tar-piped to CFS and replace the old backup file. On the other hand, many saves involve a code scheme that is saving supercomputing application dumps and saving time steps. These time step dumps are typically created as new files, whereas backups and backup-type usage patterns often replace files. The inefficiency of the tar-pipe technique is one reason the usage statistics show a faster rate for saves versus replaces.

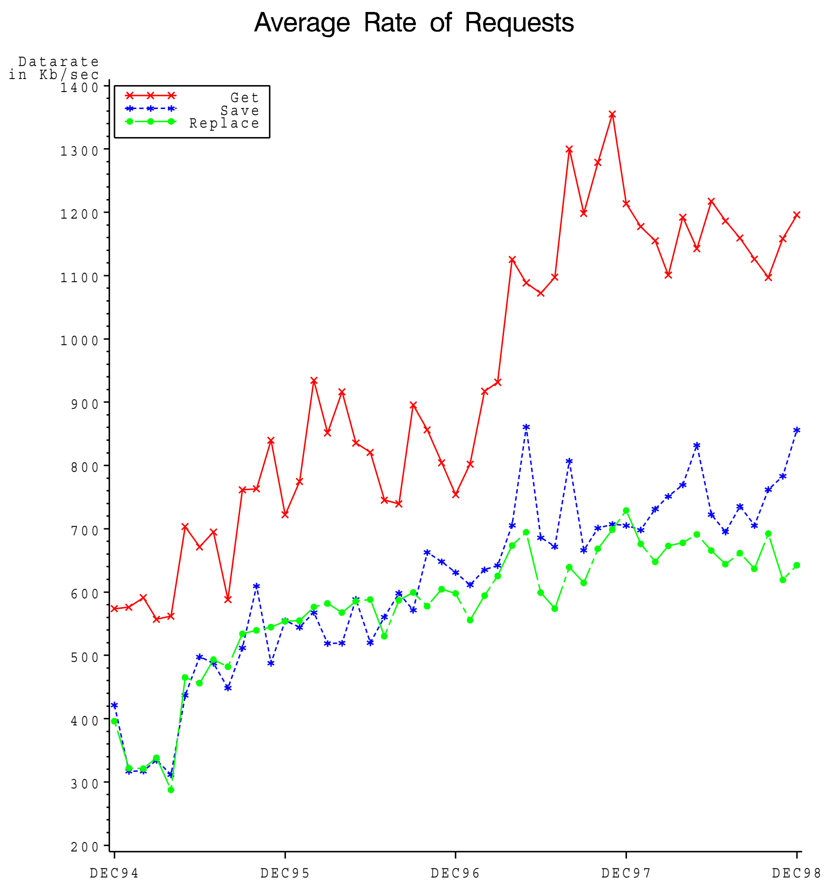

Figure 3

Another interesting aspect of usage is bytes moved. From figure 4, it is clear that most of the time the amount of data moved into the system was more than the amount of data moved out of, or moved internally within the system, except in those times when technology change necessitated the need for large amounts of internal movement. These internal movements are known as migrates. Also, it is clear that more data was saved than was retrieved. Users tended to save everything, and retrieve only a small percentage. This is in line with the idea that CFS is a basically infinite storage, and saving everything is better than the risk of losing something needed in the future.

Total Data Moved by Movement Type

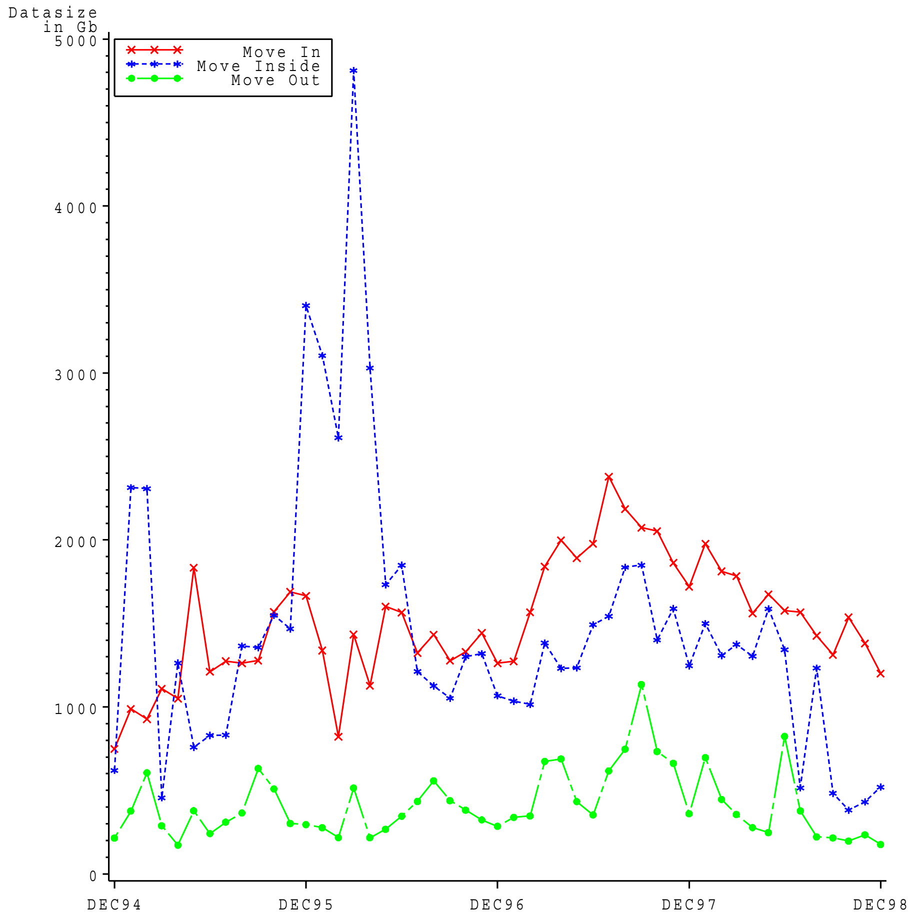

Figure 4

Figure 5 indicates the growth of CFS. The system grew around 1.5 Tb/month. Figure 6 shows the total bytes moved each month. This includes migrates, gets, saves and replaces. Contrary to expectation, the system did not trend toward more bytes moved as time progressed, but instead fluctuated fairly drastically. It is probable that there was a much more significant trend toward more data movement previous to the time covered by this study. The fluctuation certainly is in part due to various supercomputing projects that took place at different times during the seven-year period. Another possible reason for fluctuation is the effects of recharge (charges imposed on the client for space utilization). When recharge rates are changed, users either used the system less or more depending on the nature of the change. The important fact here is that it did fluctuate, and future designs should not assume linear growth. In addition, CFS is a mature system, and does not display the same growth behavior as a storage system in the beginning of its life might.

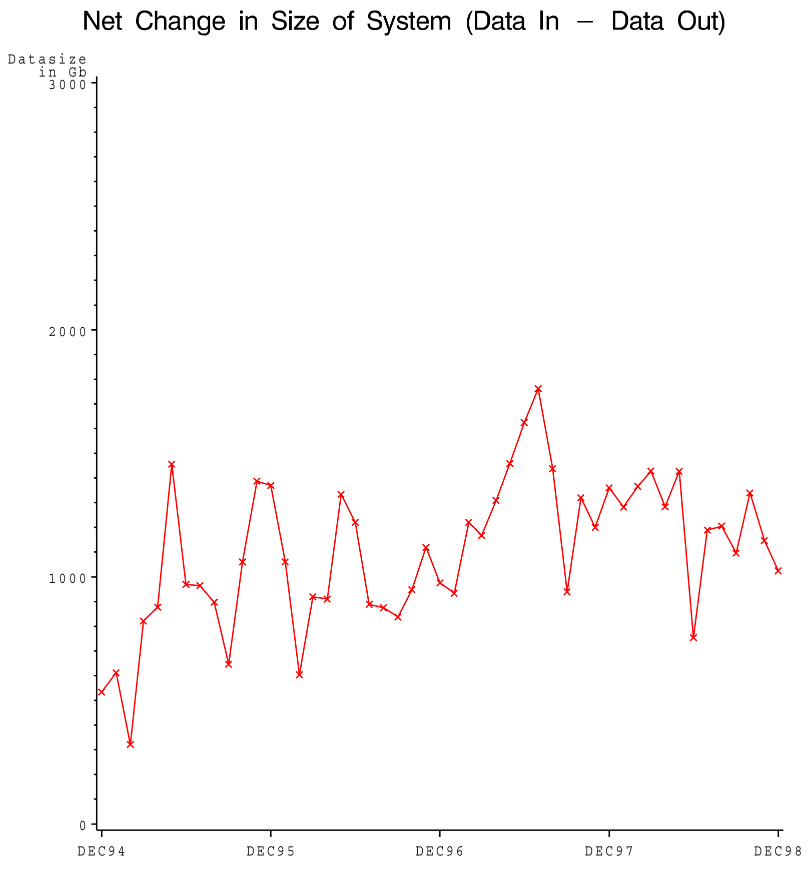

Figure 5

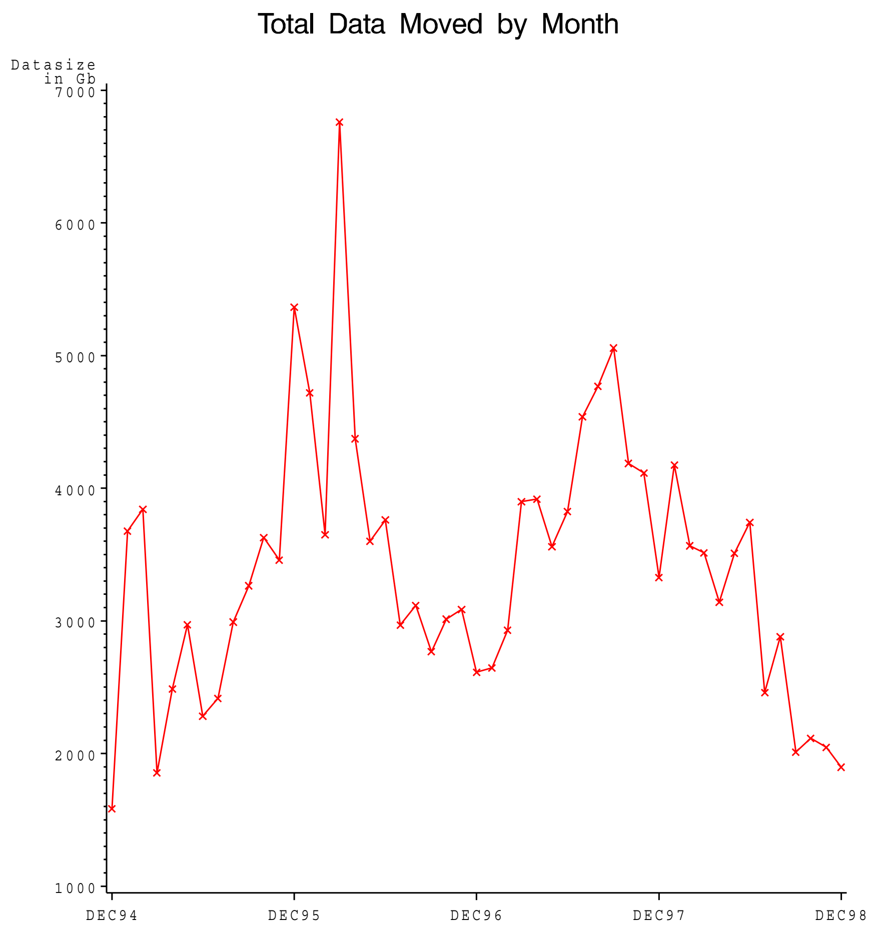

Figure 6

3 Time-to-First-Byte Performance

Wait time is another area that we examined. Wait time is the difference between request time and time of first byte transferal. In order to discuss wait time, it is necessary to describe how CFS works in a little more detail. CFS is comprised of tapes and hard disks. The tapes are stored in StorageTek robotic tape silos, which have used 1/2 inch 18 and 36 track and serpentine tapes for active files and serpentine and helical scan technology for large inactive files. When a request for a certain tape is received, the system instructs the robot to find the tape and load it into the drive. This procedure accounts for much of the wait time seen. A problem can occur when more than one user requests the same tape. In this case, user requests are queued, and wait time starts adding up. Wait time for hard drives is only due to the maximum number of tasks that CFS allows for the system or user. Needless to say, wait times for files stored on tape are much larger than for those stored on disk. Within tapes, however, some surprising results were obtained. In figure 7, we see one such result, namely, that wait time for gets was comparable to wait times for saves.

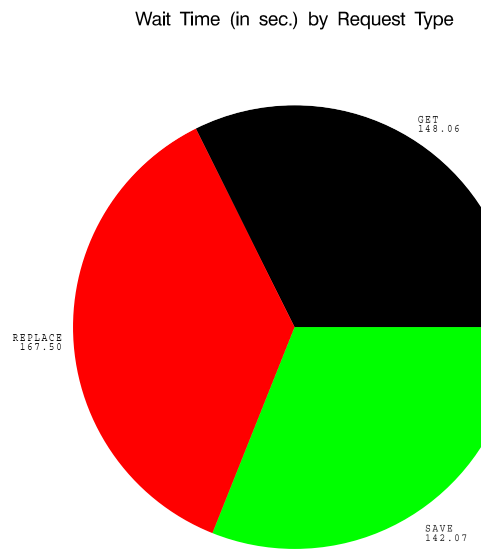

Figure 7

The way CFS handles replaces is to simply write to a new location in the system, and then update the metadata on the directory accordingly. Wait time on saves and replaces occurs because either the drives are busy when the system receives the request, or the directory metadata is busy being updated. There is a trade-off here. On the one hand, a system might only allow new files to be transferred to disk, in which case migrates will be necessary to move large files onto tape. The advantage is that the size of the file can be quickly determined, and tape usage is minimized. In the case of CFS, however, writing directly to tape is allowed. In fact, tape is the default media if the user does not specify file size. The system then runs the risk that the file was small, and a migrate is required to move the data to disk. The wait time for saves and replaces might be reduced by reserving drives for save/replace tasks. In this case, save/replace requests would not have to wait for a drive in use by the system.

4 Migration

One of the more surprising results of this study is the data produced on migrate requests. Migrates are the most prevalent request in CFS. This can be seen in figure 8.

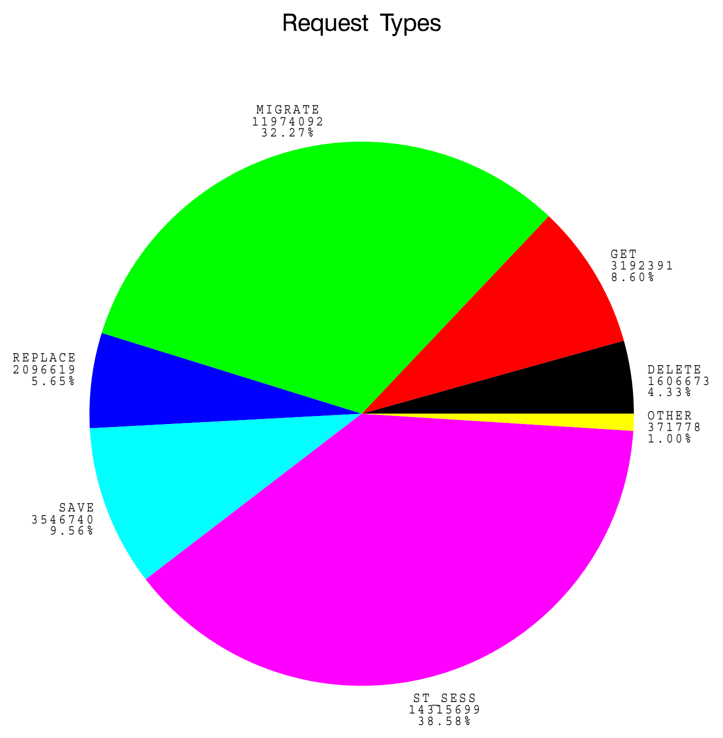

Figure 8

ST_SESS is the code for the start of any session. It will be recorded for any transaction in CFS.

Many situations require migrate requests. For example, if the user does not specify a file size when saving to CFS, the system assumes largest file size, and places the file on a large tape. After transfer is complete, if the size is small enough, the system will migrate the data to disk.

Another example of migration is the upgrading of media technology. If better, cheaper, or just different disk or tape technology is implemented, then it is necessary to migrate data from older technologies to the newer one. This can be seen in Figure 9. The spikes occurring correspond to the following upgrades in technology:
May 1995 -- Partial upgrade in disk technology
December 1995 -- Finish of upgrade in disk technology
April 1998 -- Upgrade in tape technology
November 1999 -- Upgrade in disk technology

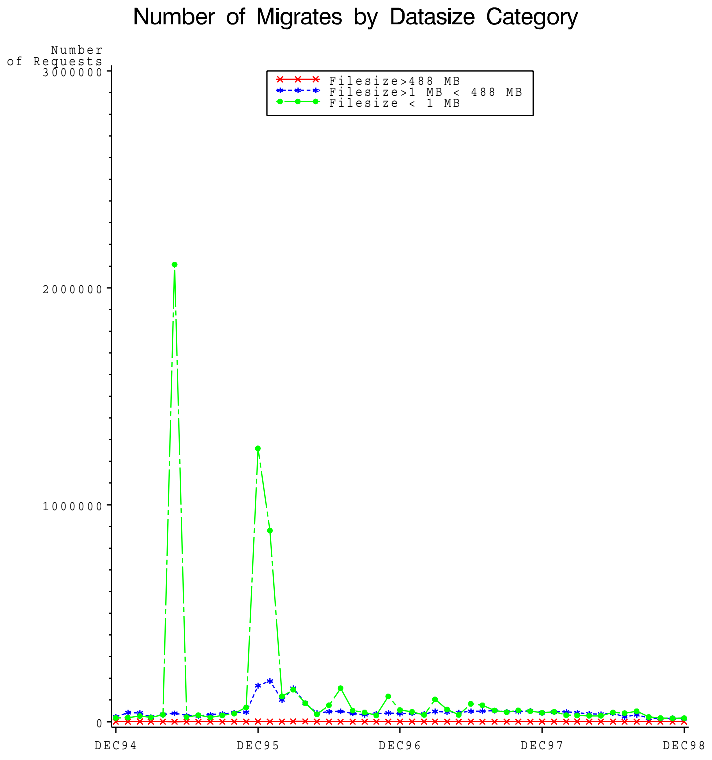

Figure 9

Another interesting feature of this graph is the file size. All of the spikes correspond to very small file sizes. One feature prevalent in our results was the fact that the majority of the system's resources were used in migrating very small files. This is possibly the most important observation made during this study. Even though this system is oriented towards supercomputer storage, these small files have a huge effect. It is their sheer *number* that causes so much migration and maintenance to be necessary. Unfortunately, even in the supercomputing world, these small files are prevalent, and unavoidable. Future designers of large storage systems should necessarily take them into account. As previously noted, the system only attains its maximum data rates for very large files. It is clear, then, that much of the system's potential for a fast data rate is not used frequently, but only when very large files are being worked with. The old problem of capability versus capacity rears its ugly head once again.

In figure 10 we see the type of migrates which took place. From figures 9 and 10 combined, one can see the type of media the system uses for different sizes, i.e. small files are on disk and large ones on tape.

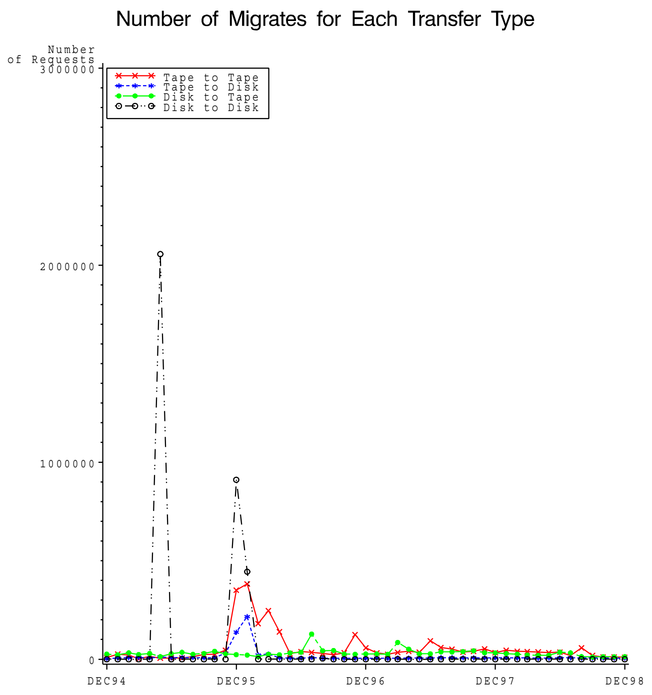

Figure 10

The spikes correspond to the disk and tape technology upgrades. If you ignore these spikes, then disk-to-tape migrates become the prevalent form of migration. These are movements of small, generally seldom used files to tape to keep the disks clear. Even with a system like CFS, which allows direct tape storage, there is still plenty of disk-to-tape movement.

5 Effects of Media Upgrade on Management

Media upgrades definitely require attention in the management of such a storage system. Media upgrades lead to large spikes in migration, as well as many months, if not years of background migration to catch up to the new technology without affecting user performance. Background migration can in part be attributed to tape technology upgrades, which are generally implemented at a slower rate. The large spikes generally come from disk upgrades, since it is important to move the data quickly onto new disks due to space and power constraints.

These spikes must be carefully monitored to ensure that real users are not affected. If the spikes result in lower performance, a rethinking of the management of the system might be required, so as to maintain performance for real user requests. Certainly migration should not affect the users. Referring to figure 11, however, we can see that these spikes did not adversely affect CFS performance.

Maximum and Average Wait Time by Month
Vs Number of Requests (in light gray)

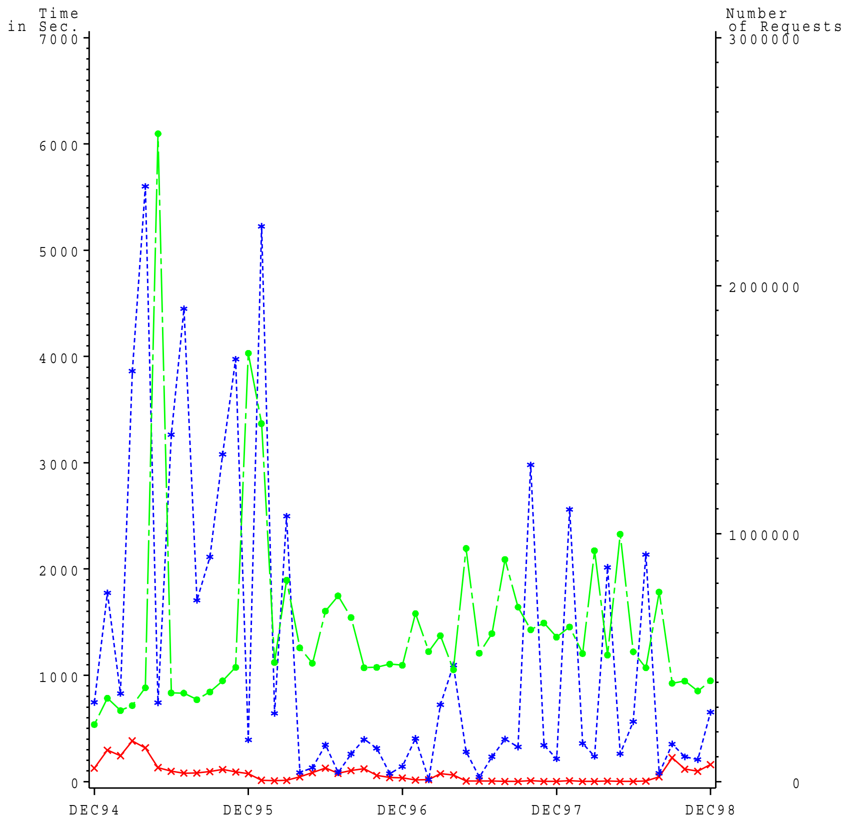

Figure 11

The request numbers seen in figure 11 reflect the large migrations occurring at that time, but wait time appears to be truly independent of request numbers. This indicates that the system was properly managed, and users saw no ill effects.

6 Future Work

Further analysis of CFS may yield many more important results. Simply describing the system and its usage is only the first step. One interesting area is the effect of technology on the system. A finer look at peak load times versus average rates for the system would be worthwhile, in order to get a better idea of the range of usage one can expect. Further investigation into the wait time associated with saves and replaces could be fruitful as well. Is this wait time due to a lack of enough drives, or is there some other cause for this delay? Another interesting avenue for research is in prediction. Is there an analogous situation to Moore's Law for large-scale storage? Our data so far has not behaved according to Moore's Law. What is the explanation for this? What other predictors, if any, exist? A more detailed study of usage is also of interest. Is it necessary to design a storage system that can achieve Gbyte/sec data rates when users only achieve Mbyte/sec due to client constraints, usage patterns, and the overwhelming effect of small files? Is all this migration really necessary, and exactly how much of an effect does it have on the overall performance?

7 Conclusions

The data suggest that the system was mainly used in a way the designers expected. Maximum data rates were only achieved for large files. Most files were fairly small, so the system wasn't used to its fullest potential most of the time from a data-rate perspective. A few surprises were encountered, however. The most important surprise encountered was the preponderance of migrates. The system performed migrates three times more than it handled user requests. The main bulk of these migrates was performed on small files. Future designers should note this. It is very important, in view of this finding, that small files need to be handled in a very efficient way to maintain the overall efficiency of the system.

Future designers might want to note that the majority of files on a system such as CFS are in the 80 Mbytes range. The large majority of users were workstations, which limited the transfer rate. The only users able to take full advantage of the systems performance were supercomputers, which comprised a small fraction of the overall usage of the system. It might be far cheaper to design a system that handles lower demand clients separately, especially since the trend to date has been to increase the maximum file size. As the gap grows between the biggest files and the smallest ones, the contention between capability and capacity will increase, and some solution must be found.

References

[1] M. W. Collins. and C. W. Mexal. The Los Alamos Common File System. *Tutorial Notes, Ninth IEEE Symposium on Mass Storage Systems*, IEEE, Oct. 1988.