Flying with Instruments: Characterizing the NCAR MSS-III Workload

J.L. Sloan
Scientific Computing Division
National Center for Atmospheric Research (NCAR)
Boulder, Colorado

Abstract

The NCAR Mass Storage System, MSS-III, generates 10 megabytes a day of transaction log containing information about its workload. Traditional metrics, such as the average amount of data stored and retrieved per hour, are useful but omit information regarding temporality, locality, and burstiness. This information is critical to characterizing and understanding the MSS workload. NCAR has begun to use metrics usually applied to virtual memories, hardware caches, and network traffic to analyze the MSS-III transaction logs.

Current MSS-III workload characterization falls into three broad categories: parametric statistics (for example, mean and variance for various file and data metrics), trace-driven analysis (for example, working set size), and trace-driven simulation (for example, compulsory and capacity cache miss ratios). Results from all of these methods are presented.

Graphs of MSS-III transactions across a range of time scales show a self-similarity or "fractal burstiness" typical of network traffic. This suggests that measurements of self-similarity (for example, the Hurst parameter) may be useful. Also, the lack of normal distribution suggests that application of nonparametric statistics might be fruitful.

Introduction

NCAR's 38-terabyte mass storage system, MSS-III, moves an average of about 5 terabytes of data per month (8 terabytes peak) responding to user requests. An equal amount of data is moved within the storage hierarchy as part of cache management [1]. The MSS-III caches consist of a 1.2-terabyte robotic tape library (which we will refer to as the *tape cache*) and a 120-gigabyte disk farm (the *disk cache*). These caches are fed from a manual archive of 112,000 tape cartridges. Where files are cached is based on their size relative to an MSS-III file threshold parameter. Currently, files smaller than 30 megabytes go to the disk cache, while larger files go to the tape cache. Clients served by MSS-III include a Cray Y-MP8/864, a Cray Y-MP2/216, a Cray-3, a TMC CM-5, an IBM SP-1, and dozens of smaller file and compute servers. The primary method of connecting to the MSS is via HIPPI channels through a series of cascaded channel switches.

In the execution of its duties, MSS-III generates transaction logging records to the tune of about 10 megabytes a day. These transaction logs can be mined for information about what the MSS is doing, when it is doing it, and who it is doing it to and for. Analysis of this information falls into three broad categories: parametric statistics, trace-driven analysis, and trace-driven simulation. We will discuss these in some detail, and toward the end we will show some results from the application of these tools.

Parametric statistics

Typical parametric statistics might include metrics such as the average file size and associated variance, maximum number of files accessed per hour, or peak megabytes per second transferred. We use these types of statistics routinely and generate useful graphs like Figure 1, which shows the average transfer rates in kilobytes per second between MSS-III and different types of client systems for files of different sizes. However, we can make what we believe are compelling arguments that while these types of statistics are useful, they do not tell the entire story.

A statistic that is frequently bandied about is that the average size of a file accessed from MSS-III is about 28 megabytes. However, the frequency distribution for accesses by file size for the 1.2 million files in the MSS is not normal. The distribution is heavily biased toward small files: Given any MSS file access, there is nearly a 50-percent probability that that access is for a file smaller than 1 megabyte and almost an 80-percent probability that it is for a file smaller than 10 megabytes. Even so, given any byte accessed from the MSS, there is less than a 1-percent probability that it is from a file less than 1 megabyte in size. Highest probabilities are for files in the neighborhoods of 80, 145, and 180 megabytes. This is

1051-9173/94 $3.00

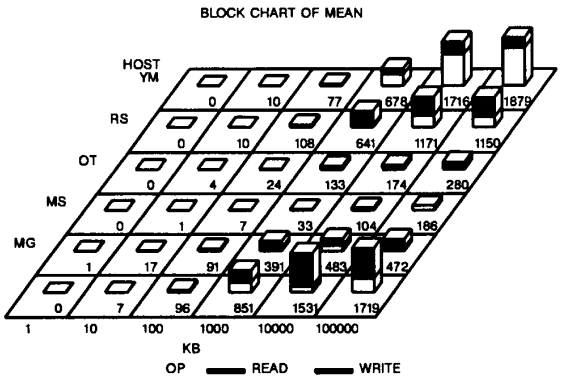

Figure 1. Transfer rates in kilobytes per second.

probably an artifact of the file usage patterns of the climate models run at NCAR. This points out the importance of determining frequency distributions for parametric statistics. Also, it suggests that nonparametric statistics (where there is no assumption of normal distribution) might be applicable.

Distributions alone are not adequate. Consider these three cases:

(1) one 1-megabyte file is read 100 times,
(2) 100 1-megabyte files are each read one time, and
(3) one 100-megabyte file is read one time.

Cases (1) and (2) are identical in terms of number of files transferred, and all three cases are identical in terms of number of bytes transferred. Yet, it is easy to see that the impact that each of these cases could have on the MSS might be as much as two orders of magnitude different from that each of the other cases could have. For example, in Case (1), the file might be cached immediately, resulting in a single manual tape mount, while Case (2) would result in 100 manual tape mounts. Fixed overhead per file transfer would be incurred 100 times for Cases (1) and (2) but only once for Case (3). Case (1) takes up only 1 megabyte in a cache, while Cases (2) and (3) each take up 100 megabytes.

Important qualities of the MSS workload are lost in the tabulation of simple parametric statistics. To recover this information, we turn to trace-driven analysis.

Trace-driven analysis

Trace-driven analysis is an attempt to show how different aspects of the MSS workload change over time. The analysis is driven by time-stamped transaction log records. This method was inspired by earlier work characterizing the Unix 4.2 BSD file system [2] and supercomputer file access [3],[4],[5]. Figure 2 shows how the running average of the number of megabytes accessed per hour through the tape cache varies from December 1992 through November 1993.

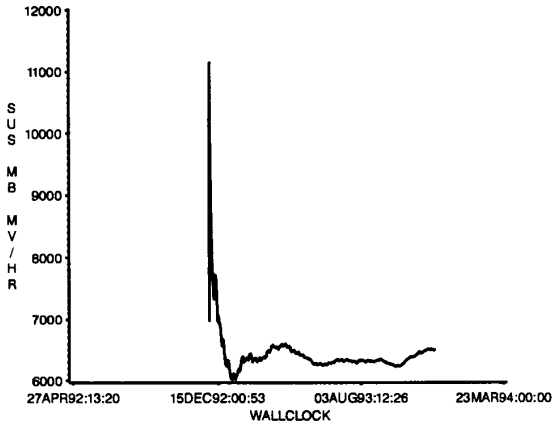

Figure 2. Accesses in megabytes per hour.

Using the running average for trend analysis is troublesome. The longer the span of time the analysis covers, the greater a change is required to perturb the graph. Over very long runs (a year or more), significant workload peaks visible to users may not be visible in the running average. Unfortunately, the burstiness evident in the raw data makes any trend analysis difficult. Linear regression usually results in confidence limits that do not inspire confidence. We will return to the bursty nature of the MSS-III workload below.

The fact that the MSS-III caches operate a very little bit like virtual memory suggests that analysis tools such as working sets [6] might be applicable. Given a desired file residency period, and by keeping a history of each file as we analyze the transaction logs, we can determine the working set of an MSS cache. The working set tells us how large a cache we would need to achieve the desired residency period, given a specific workload. Figure 3 shows how the 30-day working set of the MSS-III tape cache changes during the December-November time period.

In addition, trace-driven analysis permits us to measure the effectiveness of the MSS-III caches. Metrics of particular interest are the capacity miss ratio and the compulsory miss ratio. The former is an indication of the adequacy of the given cache size. The latter is a measure of the temporal locality of the workload and, hence, of how effective caching is likely to be. Figure 4 shows how the actual compulsory miss ratio for the 1.2-terabyte tape cache changes over the December-November time period.

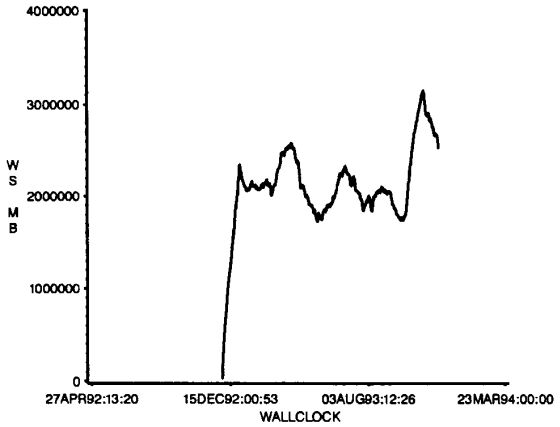

Figure 3. Working set in megabytes.

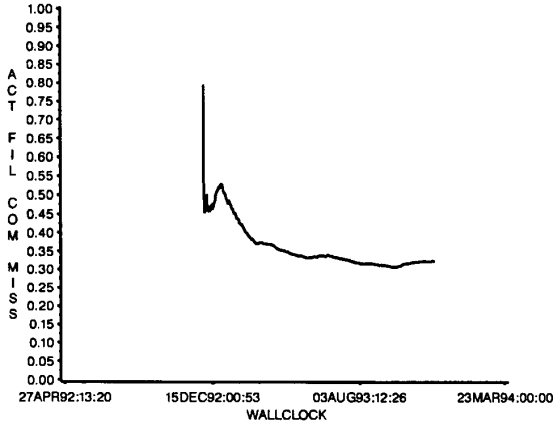

Figure 4. Actual compulsory miss ratio.

Although trace-driven analysis gives us many useful results, it does not provide any predictive capability to help us make informed decisions about hardware upgrades and changes in MSS-III strategies and parameters. For this, we turn to trace-driven simulation.

Trace-driven simulation

The design of MSS-III, using caches served by a backing store, suggests that methods for analyzing hardware caches might be applicable. Some changes are necessary; for example, cache lines in MSS-III are variable-length (the size of individual files), and when a cache line is dirtied, the entire cache line, rather than a single cache word, is replaced. The use of trace-driven simulation — feeding transaction log records to a cache simulator in which we can alter the cache size and the cache management strategy — potentially gives us useful predictive capabilities. Figure 5 shows how the predicted compulsory miss ratio for an upgraded 4.8-terabyte tape cache changes over the December-November time period.

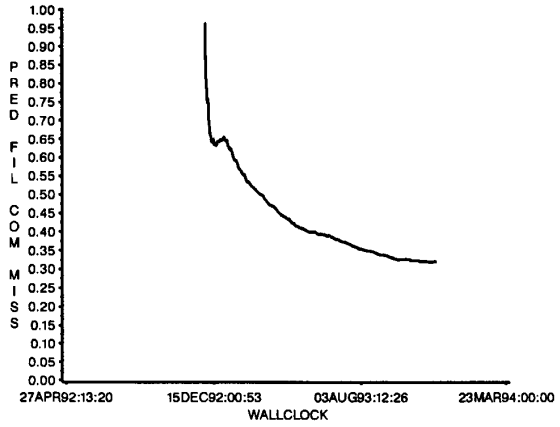

Figure 5. Predicted compulsory miss ratio.

There are a number of caveats that arise when one deals with trace-driven simulations. Many results are simply artifacts of the input workload. For example, our cache model never predicts more aggregate I/O bandwidth between the MSS and its clients than actually exists on the computer room floor. This is because the traces with the associated interarrival time of requests that drive the simulation are derived from the actual workload. Also, an enormous amount of work must be done to validate the simulation; otherwise, we would have no confidence in its predictions. We will return to model validation below. Other problems with trace-driven simulations can be found in the literature [7].

Implementation

MSS-III generates daily transaction log files that are about 10 megabytes in size. A month's worth of these files is preprocessed and saved in a monthly file. The preprocessing consists of selecting the records of interest, parsing the fields, and some minor translating. Both the daily and the monthly files are saved indefinitely in the MSS.

The workload statistics, trace-driven working set analysis, and trace-driven cache simulation were originally prototyped as separate programs written in Perl. They are now combined into a single program, *msscsim*,

written in about 12,500 lines of ANSI C. The program was originally run almost exclusively on NCAR's Cray Y-MP8/864 supercomputer but is now comfortably hosted on an IBM RS6000/980.

A typical *msscsim* run on the RS6000 against a year's worth of preprocessed log data reads 1,800 megabytes of input, generates 16 megabytes of output, uses 64 megabytes of disk space for intermediate results, consumes 50 megabytes of memory, and runs for 2 to 4 hours. Optionally, intermediate results can be kept in memory instead of on disk, but this stretches the capacity of the host.

Postprocessing is done using a collection of statistical and graphical programs written in SAS. Output includes time step graphs; burstiness profiles; trend analysis; distributions by file size, interarrival time, and frequency of access; and basic statistics by cache, device, host, user, and project.

Applications

Below are some examples in which trace-driven analysis and cache simulation were put to use at NCAR.

Workload characterization and model validation

We used *msscsim* to determine the basic characteristics of the MSS-III tape cache workload, examining preprocessed log data from December 1992 through November 1993. These data represent actual MSS-III activity; they were not derived from the simulation portion of the program. Besides being useful for planning and design, these results were used to validate the simulation. In general, the simulated results matched closely with actual results, with two exceptions: file residency times and estimated manual tape mounts.

Residency times for files in the simulated cache are collected only when a file is removed from the cache. This means the simulation has to wait until the cache fills up and files begin to be aged out before meaningful residency times begin to be generated. For a large (1.2-terabyte) cache, this can take a couple of months of transactions. It has become clear in practice that as longer periods of time are simulated, the predicted results (including residency times) converge to the actual results.

The simulation predicted four times more manual tape mounts than we see in reality. This was initially puzzling, since although MSS-III uses a number of optimization strategies to minimize manual tape mounts, our intuition was that the ratio of tape operations to tape mounts was nearly 1:1. However, this ratio had never been measured over long periods of time. Instrumentation was added to *msscsim* to measure the operation/mount ratio, and we discovered that the optimization strategies were working much better than we expected. This accounted for the discrepancy between actual and predicted tape mounts. We now distinguish between *raw* (predicted) mounts and *cooked* (optimized) mounts.

During the period examined, the tape cache had an actual 57-percent file hit ratio, a 10-percent capacity file miss ratio, and a 32-percent compulsory file miss ratio. The compulsory miss ratio is especially troubling since it is a function of low temporal locality in the workload, and it cannot be addressed by improved caching. In general, distributions of interarrival times by file size generated by *msscsim* showed that the smaller the file, the greater its likelihood of reuse.

Cache size and diminishing returns

We used *msscsim* to simulate various tape cache sizes without changing any other MSS-III parameters (for example, the file size threshold that routes files to the disk cache versus the tape cache). Given the low temporal locality cited above for files destined to the tape cache, the results were what we expected: Increasing the tape cache without changing anything else yields diminishing returns in terms of increased cache hit ratios, decreased tape mounts, and increased file residency times. For example, doubling the size of the tape cache achieved only a 30-percent increase in average file residency time.

Client disk upgrade

NCAR's Cray Y-MP8/864 supercomputer underwent a two-phase upgrade that added about 36 gigabytes of local disk space to the system. Most of this disk space was made available to users for their home directories. We had hoped that besides making the users happy, this would decrease the small-file load on the MSS. Small files (under 1 megabyte) are less efficient to store but exhibit the highest temporal locality in the MSS workload.

We applied the trace analysis feature of *msscsim* to the 4 months before and after the disk upgrade. We discovered that the major users did make good use of their larger home directory space by storing frequently used small files, removing a significant amount of small-file MSS traffic. Because the total system is MSS-bound, the small-file workload was replaced with requests for larger files, increasing the average size of files requested from the MSS. Because larger files in the workload exhibit lower temporal locality, the compulsory miss rate for the MSS caches increased, as did the number of manual tape

mounts. Users saw better overall performance, since they had to make fewer requests to the MSS, but the requests that were still performed made less effective use of the MSS caches [8].

File size threshold

We used the *msscsim* cache simulation feature to predict the effect of expanding our tape cache to 4.8 terabytes and moving the file size threshold so that more smaller files would be saved there rather than in the disk cache. This would free up space in the disk cache for more very small (and higher locality) files. File size thresholds of 30 megabytes (the current threshold), 20 megabytes, 10 megabytes, and 5 megabytes were simulated.

Overall, the simulation predicted that expanding the tape cache and lowering the threshold would be a win, increasing the average residency time for cached files. An obvious downside for MSS users was that a significant portion of the workload would be moved from a low-latency (disk) cache to a high-latency (tape) cache.

An unexpected result suggested by the simulation was that as the file threshold was decreased, the average number of manual tape mounts decreased (because of better use of the expanded tape cache), but the peak number of mounts per hour increased during bursty periods. We found that the increase in the simulated peak mounts was due to migrating old files from the disk cache back to the manual archive. As we lowered the file threshold, the distribution of file sizes in the disk cache grew narrower. We could no longer remove a larger and less frequently used file from the cache and save several smaller and more frequently used files in its place. The ratio of files removed to files added approached 1:1, effectively resulting in the loss of a "free lunch." (In practice, we do not expect to see any increase in manual mounts due to the mount optimization strategies discussed above.)

Fractal burstiness

We used *msscsim* to search for self-similarity (fractal burstiness) in the workload of MSS-III. Several runs were made with different time step intervals, adjusting the amount of log data consumed so that 50 time steps were generated by each run. Time step intervals of 15 minutes, 30 minutes, 1 hour, 4 hours, 8 hours, 12 hours, 1 day, and 7 days were used. A high degree of similarity was observed among the graphs, suggesting that techniques for characterizing self-similar burstiness in other contexts, such as the Hurst parameter [9], might be applicable.

Cache management algorithms

Msscsim makes it easy to simulate different cache management algorithms and select which algorithm to use for a particular run from the command line. We used *msscsim* to search for cache management strategies that might be more effective than those currently in production. We eventually found a strategy that was objectively superior to the production algorithm. By not caching larger (100 megabytes or more) files, which incidentally exhibit lower probability of reuse and are only referenced by only a handful of our major (best funded) supercomputer users, and saving that cache space for caching more smaller files (with higher probability of reuse) for longer periods, better overall system throughput was predicted. This strategy was informally termed "the career-limiting algorithm." Although *msscsim* predicted higher cache hit ratios and fewer tape mounts for MSS-III, we predicted unemployment for ourselves. The search for better and more politically correct cache strategies continues.

Acknowledgments

The author would like to express his appreciation for the ongoing support of his management for this project. Specifically, thanks go to Gene Harano, Bernie O'Lear, and Bill Buzbee, all of the National Center for Atmospheric Research (NCAR).

NCAR is operated by the University Corporation for Atmospheric Research under the sponsorship of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author and do not necessarily represent the views of the National Science Foundation.

References

[1] E. Harano, "Data Handling in the NCAR MSS," Nat'l Center for Atmospheric Research, July 1993.

[2] J. Ousterhout et al., "A Trace-Driven Analysis of the UNIX 4.2 BSD File System," *Proc. 10th ACM Symp. Operating Systems Principles*, Dec. 1985.

[3] E. Miller, "Input/Output Behavior of Supercomputer Applications," UCB/CSD 91/616, Univ. of California, Jan. 1991.

[4] E. Miller et al., "Analyzing the I/O Behavior of Supercomputer Applications," *Proc. 11th IEEE Symp. Mass Storage Systems*, Monterey, Calif., 1991.

[5] E. Miller et al., "An Analysis of File Migration in a Unix Supercomputing Environment," extended abstract, Univ. of California, Berkeley, Calif., 1992.

[6] P. Denning, "The Working Set Model for Program Behavior," *Proc. ACM Symp. Operating Systems Principles*, Oct. 1967.

[7] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, New York, N.Y., 1991.

[8] J. Sloan et al., "Impact of the Shavano Disk Upgrade on the MSS," Nat'l Center for Atmospheric Research, Nov. 1993.

[9] W. Leland et al., "On the Self-Similar Nature of Ethernet Traffic," *Proc. ACM SIGCOMM '93*, San Francisco, Calif., 1993.