# Towards Scalable Benchmarks for Mass Storage Systems

**Ethan L. Miller**
Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
1000 Hilltop Drive
Baltimore, MD 21228
elm@cs.umbc.edu
Tel: 410-455-3972
Fax: 410-455-3969

## Abstract

While mass storage systems have been used for several decades to store large quantities of scientific data, there has been little work on devising standard ways of measuring them. Each system is hand-tuned using parameters that seem to work best, but it is difficult to gauge the potential effect of similar changes on other systems. The proliferation of storage management software and policies has made it difficult for users to make the best choices for their own systems. The introduction of benchmarks will make it possible to gather standard performance measurements across disparate systems, allowing users to make intelligent choices of hardware, software, and algorithms for their mass storage system.

This paper presents guidelines for the design of a mass storage system benchmark suite, along with preliminary suggestions for programs to be included. The benchmarks will measure both peak and sustained performance of the system as well as predicting both short-term and long-term behavior. These benchmarks should be both portable and scalable so they may be used on storage systems from tens of gigabytes to petabytes or more. By developing a standard set of benchmarks that reflect real user workload, we hope to encourage system designers and users to publish performance figures that can be compared with those of other systems. This will allow users to choose the system that best meets their needs best and give designers a tool with which they can measure the performance effects of improvements to their systems.

## 1. Introduction

Mass storage systems are used by many data centers around the world to store and manage terabytes of data. These systems are composed of both hardware from many vendors and storage management software, often from a different vendor. Each data center builds its own system, and no two are alike. How can two different mass storage systems be compared? Additionally, how can users gauge performance of planned systems?

We believe the introduction of a standard benchmark suite for mass storage systems will enable storage users to plan their systems in the same way that the SPEC and Perfect benchmarks allow users to compare different computing systems. In such suites, one or

more of the benchmarks should sufficiently resemble a user's needs so that she can predict the performance on her own application. Similarly, data center personnel should be able to pick the metrics that most closely model their mass storage workloads, allowing some prediction of system performance without the need to experimentally configure multiple systems.

Mass storage system benchmarks must be portable, scalable, and reflective of real system workloads. Achieving portability will require limiting the scope of changes that must be made to the tests for different systems. Scalability is necessary because a mass storage system can hold from tens or hundreds of gigabytes to petabytes, and access patterns and file sizes will both vary greatly across this range of sizes. Finally, benchmarks must reflect real system workloads. Rather than rely on a single metric, a mass storage system benchmark suite must test both burst and sustained transfer rates and gauge the effectiveness of migration algorithms using several "typical workloads.

This paper proposes several candidate benchmarks for a scalable mass storage system benchmark suite. These programs are synthetic; they do not include code from actual user applications, but instead are based on access patterns observed on real mass storage systems. Some of the benchmarks generate access patterns similar to those of individual programs, typically requiring less than a day to run. Others model long-term access by many users to mass storage over periods of many days. Both types of benchmarks include programs that mimic "real world" access patterns as well as others that stress the system to find its limits, since both factors are important to mass storage system users.

While this paper contains concrete suggestions for mass storage systems benchmarks, there is still much work to be done. Using feedback from users of mass storage systems as well as vendors, it is our hope that this benchmark suite will become a standard that will ease the process of comparing many different options in storage system design.

## 2. Background

Research into mass storage system benchmarks builds on work in two different areas: the analysis of mass storage system usage patterns and the development of benchmark suites for different areas of computer systems. There are many papers that discuss benchmarks for areas ranging from processors to file systems to disks, providing a good foundation for deciding what a benchmark should (and shouldn't) do. However, there are relatively few quantitative papers on the usage patterns seen by mass storage systems; while many organizations study their systems to help plan for the future, these studies are rarely published.

Of the many papers that have been published about benchmarking, the most relevant to this research are those on file system benchmarks. These benchmarks fall into two broad categories: those that consist of a single program [1,2] and those that are built from many programs and are meant to model longer-term usage [3]. Additionally, many papers use ad hoc benchmarks to compare research file systems to already-existing file systems.

The few file system benchmarks that do exist are designed to test workstation-class file systems, both standalone and in a networked environment. Several of the benchmarks consist of a single program sending many read and write requests to the file system; such programs include IOStone [1], iozone [4], and bonnie [5]. These benchmarks are designed to gauge the maximum file system or disk system performance available to a single application over a short period of time. However, constant improvements in memory size and disk performance require scalable benchmarks; Chen's scalable disk benchmark [2] addresses this problem by scaling the workload to the system used. Still, this scaling is restricted to a single program.

NFSstone and the Laddis benchmark used by SPEC, on the other hand, are designed to model the activity of several programs and their effects on the file server. Rather than present a workload from a single client, these benchmarks can mimic an entire network of workstations. These benchmarks may be scaled by increasing the file request rate or the file size or both. Unfortunately, they are very specific in that they test the ability of a file server to respond to NFS requests. While NFS is a commonly-used file system, it is not clear that good performance for an NFS workload is necessarily the hallmark of a high-performance file system.

An even more complex benchmark, the Andrew file system benchmark [3] tests the entire file system by including operations such as file creation and deletion. However, the Andrew benchmark is not directly scalable, and still runs for only a few minutes or less. Clearly, a mass storage system benchmark must measure performance over longer periods of time as well as gauging the burst rates that the system can attain.

Many researchers gauging the performance of their new file systems create their own "benchmarks" that involve reading and writing many files. While such ad hoc benchmarks can provide comparisons between different file systems, they require that the authors of such benchmarks run them on all of the systems being compared. This burden is not excessive for researchers because they often compare their research file system to one or two "real-world" systems that are already running at their site. However, this approach creates problems for "normal" users because most of them do not have access to the systems whose performance they wish to measure. While this approach is infeasible for standardized comparisons of many mass storage systems, the idea behind it is a good one: use a typical workload to measure performance. This method can be varied to find both performance under a normal load and the maximum load a system can handle.

Since synthetic benchmarks must mimic actual usage, knowing the access patterns exhibited by users of real systems is crucial. The system at the National Center for Atmospheric Research was studied in [6], and the National Center for Supercomputing Applications was studied in [7]. Both of these studies suggest that mass storage system performance must be measured over a period of days or weeks because that is the time scale over which file migration algorithms operate. Examining shorter periods is similar to running file system benchmarks that access a file that fits in memory — it provides a measure of peak bandwidth but does not give an indication of long-term performance.

These papers, along with other studies done for internal use at various organizations, provide a basis for designing benchmarks that test long-term mass storage system performance.

Short-term performance of large storage systems is also an important metric. Benchmarks that stress the file system as a single program would can model their I/O after the single program access patterns such as those reported in [8] and [9], which detail usage patterns exhibited by supercomputer and parallel computer applications, respectively. Short-term benchmarks might also include programs that stress the storage hierarchy, such as one that searches the relatively short file headers of hundreds of multi-megabyte files for a certain pattern.

## 3. Benchmark Characteristics

In order for a collection of mass storage system benchmarks to be useful, the benchmarks must have several features and characteristics. First and foremost, they must provide a good basis for comparing two systems that may be very different. They must also be portable and scalable, and should reflect real system workloads.

### 3.1 Suitability

Perhaps the most important quality for a benchmark suite is suitability. A benchmark must do two things to be useful. First, its results must bear some relation to the real use of a system. Typically, this is a predictive relation — the performance of a benchmark should be directly related to the performance of a real workload that the user will eventually run. Second, a benchmark suite should allow the comparison of two different systems in a manner more meaningful than "A is faster than B." While this is a good observation, it is almost always necessary to know *how much* faster A is relative to B.

Benchmark suites such as SPECint95, SPECfp95 and Perfect [10] are successful in large part because they use real programs (or fragments of them) to predict the performance of a computer system. A combination of several of the benchmark programs from these suites that closely mirrors the intended use of a system can usually be found, and the performance of the system on the real workload can be approximated by combining the system's scores on each individual benchmark program. Thus, benchmark reporting usually includes both a suite-wide average and a listing of the components' individual scores. The average is useful for gauging overall performance, while the individual listings allow the prediction of performance for a specific workload.

A relatively small suite of benchmarks works well for CPU benchmarks, but how will it work for mass storage systems? A benchmark suite may contain dozens of programs, but they are of no use if a user cannot assemble some of them into a workload that resembles her usage patterns. Fortunately, there are some general access patterns for mass storage

systems that may be generated by a benchmark suite. These patterns will be discussed in Section 4.

### 3.2 Portability

The portability of a benchmark suite is another major concern for mass storage system benchmarks. CPU benchmarks are often portable because the interface to the system is at a high-level — programs are simply written in a high-level language such as C or FORTRAN. Running the benchmark on a new system requires is largely dependent on the existence of a compiler for the appropriate language being available for the system being tested. While there are may be other requirements for a portable CPU benchmark such as environment or operating system dependencies, building portable benchmark suits for CPUs is relatively well understood.

Portability of mass storage system benchmarks is another matter altogether. While mass storage systems tend to have the same functionality, they often have very different interfaces. Some systems require a user to explicitly request transfers to and from tertiary storage, while others do so automatically. Worse, the commands to effect such transfers are often different from system to system. As a result, mass storage system benchmarks will likely need to be customized to run on each individual system. To preserve portability, this customization should be limited to a few small pieces of code so that porting the benchmarks to new systems is not a difficult task. Nonetheless, there may need to be large changes in the benchmarks between systems. While it is straightforward to make a small change to read and write files via system calls or ftp, it may be more difficult to adapt a benchmark that assumes explicit transfers of files between tertiary storage and disk to a system that uses implicit transfers. These tradeoffs will be discussed in Section 4.

A second difficulty with portability of mass storage system benchmarks is the existence of different features on various mass storage systems. This issue is not present in CPU benchmarks — while an individual processor may not have a vector coprocessor or floating point unit, it *can* emulate those features using other instructions, albeit at a loss of speed. However, mass storage systems may have features that are simply not present elsewhere and that greatly improve performance. For example, one mass storage system might have the ability to compress files before storing them tape, while another lacks this feature. Should the two systems be compared without compression? The use of compression is likely to slow down the system that uses it, but it will also free up additional space. The decision of whether to include such features will be left to the benchmarker; as long as the settings of such relevant features are reported, a user can choose the appropriate benchmark results.

### 3.3 Scalability

The second goal of the benchmark suite is scalability. The suite must permit comparisons of two systems of roughly equivalent size, regardless of whether their capacity is 50 GB or 500 TB. On the other hand, comparing the performance of two mass storage systems of very different sizes makes little sense since the two systems will almost certainly have different workloads — a 50 GB storage system would not experience many repeated reads and writes of 1 GB files, though a 50 TB system certainly might.

Scaling the benchmarks can be done by a combination of two methods: increasing the request rate, and making individual requests larger. Increasing the request size reflects the larger data sets that necessitate larger storage systems. However, more storage space can also correspond to a larger user community or faster computers, both of which can increase request rate as well as allowing larger data sets. The TPC database benchmarks [10] follow this model, increasing request rate as the capacity of the storage system increases while keeping request size relatively constant.

Not all of the benchmarks need be scalable in order to provide a scalable benchmark suite, though. Clearly, some of the benchmarks must place a higher demand on the system as it becomes larger, but individual benchmarks need not. For example, a benchmark that mimics the behavior of a single program requesting a single gigabyte file might not change from a 50 GB system to a 50 TB system. Since this benchmark measures peak transfer bandwidth and nothing else, it does not have to scale up as the system becomes larger. However, other benchmarks must measure the performance of the system as a whole instead of focusing on short-term performance issues such as peak transfer rate. It is these benchmarks that must take parameters governing their behavior to allow them to model various workload levels. A benchmark measuring a storage system's ability to service clients, for example, must take the number of users as an input. For small systems, this number might be three or four. For larger systems, though, it could be several hundred. Similarly, average request size and an individual user's request rate will be different for different systems; these parameters must be customizable between benchmarks.

### 3.4 Feasibility

While mass storage system benchmarks share many characteristics with CPU and disk benchmarks, they also have limitations not suffered by CPU and file system benchmarks. CPU benchmarks usually have running times of a few hours or less, with many needing only a few hundred seconds to complete. Disk benchmarks may take longer, but still complete in well less a day for even the longest benchmarks. These time scales are too short for mass storage system benchmarks, however. Individual programs using a mass storage system may complete in a few hours or less, but long-term performance is just as important, and much more difficult to measure. The effects of a poorly-chosen file migration algorithm may not be apparent until several weeks have passed because the storage system's disk is not filled until then. Worse, policies governing file placement on tape may have little effect on overall performance until files are migrated from disk to

tape and back, a process which could take several months before a significant number of files have taken the path.

Additionally, long-running benchmarks are difficult to use for tuning purposes. Seeing the effect of a faster CPU on a benchmark suite requires only an hour or two, while adding one more tape drive may not show performance improvement on a benchmark suite for days. This lack of responsiveness makes it likely that mass storage system benchmarks will be run on simulators rather than on real equipment at least some of the time; this requires the development of good simulators that model software systems and their quirks as well as hardware.

A second issue for mass storage system benchmarks is the existence of a system on which the benchmarks can be run. This is typically a simple matter for CPU benchmarks because the manufacturer usually has a system on which the benchmarks can be run. For expensive supercomputer systems, the manufacturer need only run the suite as part of the development process or even during the testing period for a new system. Since the benchmark suites take less than a day, the equipment cost is minimal. For mass storage systems, however, equipment cost is not as low. A system is usually built from components from several vendors, and the installation of the software to manage the storage is hardly trivial. The difficulty of assembling a storage system for benchmarks is another factor that makes it likely that a benchmark suite used for its predictive ability will be run on simulated rather than real hardware.

## 4. Proposed Benchmark Programs

Based on the analyses presented in several studies of mass storage systems [6,7] and the behavior of individual programs [8,9], we propose a collection of mass storage system benchmark programs. To assure their portability, the benchmarks use few file system features beyond reading, writing, file creation, file deletion and directory listings. Rather, they focus on the ability of the mass storage system to supply and store data. They are not restricted to reading and writing whole files, however; some of the benchmarks perform operations that model workstation file usage of large scientific files including partial file reads. Although such operations may not be supported efficiently by many mass storage systems today, our experience has shown that users viewing large data files often do not view the entire file.

The benchmarks in this collection fall into two broad categories: short-running benchmarks that highly stress the system to gauge its maximum performance, and long-running benchmarks that model long-term user behavior, allowing the testing of file migration algorithms and other long-term processes that cannot be measured by a single program that only runs for a few hours. It is our expectation that the long-running benchmarks will be primarily run on a simulation model of the mass storage system rather than on an actual system because of the time and expense involved in dedicating a storage system to a benchmark suite for more than a month.

### 4.1 Short-Running Benchmarks

One aim of the benchmark suite is to measure short-term performance of mass storage systems. Since these systems consist of both disks and tertiary storage devices such as tapes and optical disks, any benchmark suite must be capable of measuring the sustained performance of each of these parts of the system. Measuring the peak performance of the disk is straightforward, but measurements of tertiary storage device performance may be more difficult, particularly in systems that do not require explicit commands to move files between disk and tertiary storage.

The first program in the benchmark suite merely writes several large files and then reads them back. The number of files to be written and the size of the files is configurable, allowing users to scale up the benchmark to larger systems. This benchmark only tests peak sequential read and write performance; it does not attempt to gather any other file system metrics. Nonetheless, the peak performance of a file system on large sequential reads is of great interest to many users, necessitating the inclusion of such a benchmark.

A similar program can be used to measure the ability of a mass storage system to create and delete small files. As with the first program, the number and size of files are specified as parameters. Rather than merely create all of the files, though, this benchmark creates the files, lists the directory in which they were created, reads them in, and then deletes them. These operations stress other aspects of the mass storage system software, showing its performance on small file operations.

Another variation on the first program creates several large files and then reads only the first few blocks of each file, "searching" for a particular piece of data. This benchmark is similar to the access pattern exhibited by a user when she is looking through the headers of large data files.

The remaining "micro-benchmarks" model two types of real user behavior: workstation users accessing the mass storage system, and scientific programs using the storage system for input and output. Human users typically read a group of files over the span of several hours, perhaps performing a few writes during that time. While some files are read in their entirety, many are partially read as users look at slices through their data. Since this program is designed for measuring short-term performance, it only models a user's access to a single set of data over a relatively short period of time. Longer-term benchmarks that model user behavior are mentioned in Section 4.2. While this program only generates the workload for a single user, it is possible to run multiple copies of the program, generating a workload resembling that from multiple users.

Batch jobs using the storage system behave quite differently from human users. They almost always read entire files and perform more and larger writes than do humans [6], stressing the storage system in a different way. Programs such as out-of-core matrix decomposition and global climate modeling make excellent benchmarks because their I/O access patterns can easily be captured without the need to actually perform the computations called for in the programs [12]. Rather than actually factor a large matrix, the

benchmark simply reads and writes the files in the same pattern as the real application . Similarly, the benchmark simulating global climate modeling does not do any actual modeling, but rather follows the same access pattern as the real program. This allows the benchmarking of a high-performance storage system without the need for a high-powered CPU to run applications. This is particularly important for planning purposes, since there may not yet be a computer that can run the program sufficiently fast — given the rate with which computers increase in processing power, a storage system that will become operational in eighteen months must deal with programs twice as fast as those running today.

The benchmarks listed in this section are generally useful for determining peak performance for bandwidth, request rate, or both. Combining the various benchmarks and running several copies of each allows users to customize the benchmark to their needs, matching the presented workload to what their installation will eventually support. However, these benchmarks are only good for measuring peak rates such as maximum bandwidth for reading files from tertiary storage or disk or the maximum rate at which a user may create small files. They do not measure any long-term statistics such as the efficiency of the file migration algorithms or the efficacy of tertiary storage media allocation.

### 4.2 Long-Running Benchmarks

A second class of benchmarks are those that generate multi-week workloads. Unlike CPUs and disks, mass storage systems exhibit activity with cycles considerably longer than a day. To measure the effects of file migration and differing sizes of disk cache for tertiary storage, benchmarks must run sufficiently long so that the disk fills up. Merely filling the disks is not sufficient, though, since the benchmark must also exhibit other user behaviors such as occasional file reuse after a long period of inactivity.

Fortunately, long-term benchmarks can be built from the short-term benchmarks mentioned in Section 4.1. Rather than running the benchmark programs alone or in small groups, though, long-term benchmarks run hundreds or thousands of instances of the same programs, possibly supplying different parameters for each run. This is done by a "master" program that controls the execution of all of the micro-benchmarks.

In addition to the usual issues of setting parameters appropriately, the master program may also need to throttle the execution of the benchmark suite. For example, a batch job that normally takes 200 minutes might take only 180 minutes because of improvements in the mass storage system. Rather than leave the machine idle for that period of time, the master benchmark coordinator should run the next benchmark "job." Of course, not all benchmarks need such throttling — it is unlikely that a human being will want to come to work earlier just because she finished a few minutes early the night before. Thus, the benchmark coordinator only throttles batch jobs, leaving the programs modeling human behavior unaffected. While this may not accurately reflect reality (people may

actually *do* more work with a more responsive system), the question of gauging the changes in human response time are beyond the scope of this work.

Because of the length of time necessary to run a long-term benchmark and the expense of setting up and maintaining a system for the weeks necessary to complete its run, it is likely that most long-term benchmarks will be run on simulations of a mass storage system rather than on real hardware, as will be discussed in Section 4.3.

### 4.3 Running the Benchmarks

A major concern with a benchmark suite is the method used to run it. CPU benchmarks are simply run as programs, either individually in or in a group. The results of running the benchmark are the time taken to complete it and the amount of work the benchmark program did. A similar principle applies to file system and disk benchmarks because their behavior can be encapsulated in either one or a small group of programs.

Mass storage system benchmarks follow the same general guidelines but on a different time scale. Certainly, some benchmarks will consist of a single program or a small group of programs that finishes within a few hours. Since these benchmarks will model individual programs, they must intersperse "computation" with requests for file data. This presents a problem, however — a mass storage system's performance should not be dependent on the computation speed of its clients. To address this problem, benchmarks will avoid computation as much as possible, focusing on file I/O. Benchmarks will thus often be of the form "transfer all of this data, and then do nothing with it." While this format removes the effect of a slower CPU, it allows the file system to perform "optimizations" by not actually fetching or storing the requested data. This can be prevented by writing files with pseudo-randomly generated data, reading them back in, and checking the results by either using the same generator or computing the digital signature for the file and ensuring that it matches that computed for the original.

Workload generators that may run for many days present a different set of problems. If a system crashes in the middle of a one hour benchmark, the program can just be rerun from the start. This is not practical for benchmarks that may run for more than a month (though it may encourage mass storage system software vendors to improve the quality of their code...). Instead, the workload generator may be restarted so it begins with the next request after the last one that completed successfully. Of course, such an outage will adversely affect overall performance, since the time spent fixing the system counts towards the total time necessary to run the benchmark.

### 4.4 Benchmark Customization

Running the benchmark programs may require customization in the form of providing the appropriate calls to open, read, write, close, and perform other operations on files and directories. To facilitate customization, the benchmark suite uses a standard library across

all programs to access the mass storage system. This library can contain real calls to a storage manager, as would be required for short-running benchmarks, or it can contain calls to a model of the storage system that returns appropriate delays. Since the interface to the storage system is localized to a single file, the benchmark suite can easily be ported to new architectures by modifying that library file.

Localizing the interface to a single file allows benchmarks to be widely distributed, and lessens the ability of manufacturers to "cheat" on the benchmarks by reducing the changes they may make to the benchmarks. It also facilitates the development of new benchmarks, since the programs may call a standard interface rather than requiring a custom interface for each system. It also encourages the development of a standard set of capabilities for mass storage systems because "special" functions are not exercised by the benchmarks and will not improve their performance. While this may sound restrictive, it will actually benefit users by ensuring that they will not need to modify their programs to run efficiently on different mass storage systems.

## 5. Evaluating the Benchmark Programs

The true test of benchmarks is their ability to predict system behavior; thus, we plan to test our benchmark suite on several systems to gauge how well its results match the actual performance of working systems. Because the designs presented in this paper are very preliminary, we expect that several rounds of benchmark tuning will be necessary before the suite is ready for wider distribution.

The basic testing method is similar to that of benchmarks in other areas: obtain performance measures from the benchmark by running it on several systems, and compare the results with the actual performance of the systems. This exercise is not as simple as it may seem, however, because no two mass storage systems have the same workload pattern. For a fair test, it will be necessary to select the most appropriate benchmark mix for a system without knowing in advance what performance to expect. Thus, our final test will be to run the benchmark on one or more systems before measuring performance and looking for correlation between predicted performance and real performance.

## 6. Future Work

The work on mass storage system benchmarks presented in this paper is still in its very early stages. By presenting these ideas to the mass storage system community at this point, we hope to get valuable feedback and direction for the construction of this benchmark suite. In particular, we hope that mass storage system users will contribute representative codes to be added to the collection.

Our first goal is to produce source code for several of the benchmarks mentioned in the paper and run them on different storage systems including workstation file servers as well as multi-terabyte tertiary storage-backed storage systems. Using the results, we plan to

refine our benchmarks, producing a set of a dozen or fewer programs that generate workloads representative of those seen in production mass storage systems.

We are also building a simulation model of mass storage systems to allow the running of long-term benchmarks. When this model is complete, we will be able to examine long-term effects such as the tradeoffs between different file migration algorithms and performance gains from different sizes of disk cache for tertiary storage. Using the benchmark suite rather than a particular workload will allow us to come up with general guidelines for building mass storage systems rather than the site-specific advice common in the field today.

## 7. Conclusions

This paper presented design principles for building a benchmark suite for a mass storage systems with capacities ranging from tens of gigabytes to petabytes. The benchmark programs will be synthetic; while they will be based on access patterns observed on real mass storage systems, they will not include real code from actual user. Some of the benchmarks will generate access patterns similar to those of individual programs, typically requiring less than a day to run. Others will model long-term access by many users to mass storage over periods of many days. Both types of benchmarks will include programs that mimic "real world" access patterns as well as others that stress the system to find its limits, since both factors are important to mass storage system users. Using feedback from users of mass storage systems as well as vendors, it is our hope that this benchmark suite will become a standard that will ease the process of comparing many different options in storage system design.

## References


1. A. Park and J. C. Becker, "IOStone: A synthetic file system benchmark," *Computer Architecture News* 18(2), June 1990, pages 45-52.

2. P. M. Chen, and D. A. Patterson, "A New Approach to I/O Performance Evaluation - Self-Scaling I/O Benchmarks, Predicted I/O Performance," *Proceedings of the 1993 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Santa Clara, CA, May 1993, pages 1-12.

3. J. H. Howard, M. L. Kazar, S. G. Menes, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham and M. J. West, "Scale and Performance in a Distributed File System," *ACM Transactions on Computer Systems* **6**(1), February 1988, pages 51-81.

4. W. Norcott, IOzone benchmark source code, version 2.01, posted to comp.sources.misc, October 30, 1994. Available from `ftp://www.cs.umbc.edu/pub/elm/iobenchmarks/iozone01`.

5. T. Bray, Bonnie benchmark source code, 1990. Available from `ftp://www.cs.umbc.edu/pub/elm/iobenchmarks/bonnie.sh`.

6. E. L. Miller and R. H. Katz, "An Analysis of File Migration in a UNIX Supercomputing Environment," *USENIX - Winter 1993*, San Diego, CA, January 1993, pages 421-434.

7. D. W. Jensen and D. A. Reed, "File Archive Activity in a Supercomputer Environment," Technical Report UIUCDCS-R-91-1672, University of Illinois at Urbana-Champaign, April 1991.

8. E. L. Miller and R. H. Katz, "Input/Output Behavior of Supercomputer Applications," *Proceedings of Supercomputing `91*, Albuquerque, NM, November 1991, pages 567-576.

9. D. Kotz and N. Nieuwejaar, "File-System Workload on a Scientific Multiprocessor," *IEEE Parallel and Distributed Technology* **3**(1), Spring 1995, pages 51-60.

10. M. Berry, et. al, "The PERFECT Club Benchmarks: Effective Performance Evaluation of Supercomputers," *International Journal of Supercomputer Applications* **3**(3), 1989, pages 5 - 40..

11. Transaction Processing Perf. Council. Details available from ***http://www.tpc.org***

12. E. L. Miller, *Storage Hierarchy Management for Scientific Computing*, Ph.D. dissertation, University of California at Berkeley, April 1995. Also available as technical report UCB/CSD 95/872.