IOStone: A Synthetic File System Benchmark

ARVIN PARK AND JEFFREY C. BECKER
Division of Computer Science, University of California, Davis, CA 95616, U.S.A.
netmail: park@iris.ucdavis.edu, tel: (916) 752-5183, fax: (916) 752-4767

RICHARD J. LIPTON
Department of Computer Science, Princeton University, Princeton, New Jersey 08544, U.S.A.

SUMMARY

This paper presents a portable benchmark called *IOStone* that measures file system performance on a string of file system requests that is representative of measured system loads. Instead of isolating a particular aspect of file system performance such as disk access speed, or channel bandwidth, *IOStone* measures performance of the entire file system which includes components of disk performance, CPU performance on file system tasks, and disk cache performance. *IOStone* provides a basis for comparing performance of different file system implementations. It can also guide system builders in matching processor performance with file system performance. Measurements made using the *IOStone* benchmark indicate that a good balance between processor performance and file system performance is rarely achieved. Version II of the C language version of *IOStone* is available through electronic mail from becker@iris.ucdavis.edu.

INTRODUCTION

This paper presents a benchmark called *IOStone* that measures file system performance on a workload that is derived from measurements of real system workloads. *IOStone* provides a basis for comparing the performance of different file system implementations. System vendors commonly quote bandwidth and latency numbers for disk drive hardware as indicators of file system performance. However, these numbers do not account for contributions from disk caches, file system structures, or CPU overhead. All of these factors play a major role in determining file system performance. Although no single metric can completely characterize file system performance, a metric that indicates file system performance under typical system loads, such as *IOStone*, can be used to compare the performance of different file systems. This is preferable to the more arbitrary comparisons that are currently made.

Similar benchmarks have been developed to measure processor performance. The

Whetstone benchmark[1] measures processor performance on a synthetic instruction mix that is representative of instruction streams of numerical scientific codes. The *Dhrystone* benchmark measures processor performance on a synthetic instruction mix that is representative of instruction streams of system codes[2]. This previous work on processor performance motivated development of the *IOStone* benchmark which measures file system performance on a synthetic workload that is representative of measured file system loads.

Other benchmarks measure file system performance. The *I/O Call* benchmark measures the time required to perform a large number of read and write operations to the same 500 byte file[3]. Although *I/O Call* is simple, portable, and easy to use, it only measures system call overhead which is just one factor contributing to file system performance. *IOBENCH*[4] is a more comprehensive measure of file system performance. However the workload it generates is not derived from traces of real file system workloads, so *IOBENCH* numbers may not be an accurate indicator of file system performance. The *IOBENCH* program also contains hundreds of lines of system-dependent code which makes it very hard to port to different machines. Except for *IOStone*, no presently existing benchmark measures file system performance on typical file system loads.

PREVIOUS STUDIES

Results of previous file system studies guide the construction of a synthetic file system load for the *IOStone* benchmark. Several studies[5,6,7,8] take dynamic measurements of file system activity that can be used to establish a representative workload. The BSD study[6] is currently the most comprehensive measurement of file system activity. It traces file references for three multi-user VAX 11/780 minicomputers running UNIX 4.2 BSD. The study gathers trace data by modifying file system calls to record file system requests. A database study[7] measures I/O system usage of the IMS database system on an IBM mainframe. A mainframe study[8] measures more general I/O system loads for IBM mainframes at Crocker Bank, Hughes Aircraft Company, and Stanford Linear Accelerator Center. This study gathers the physical addresses of large disk tracks which do not account for individual file accesses or page requests. A workstation[5] study gathers disk request data from a 68000 based workstation running the UNIX operating system. Like the mainframe study, this data is gathered at the disk driver level, so information on individual file accesses is not accounted for.

Although these studies were performed across a range of installations and systems, several characteristics of the workload are the same for all of them: Write operations account for about one third of all data transfer operations; the majority of file system

transfers tend to be short; file system accesses exhibit a large amount of spatial and temporal locality. Consequently, small disk caches significantly reduce file system traffic between main memory and secondary storage[6,8].

IOSTONE BENCHMARK

There are two main goals for the *IOStone* benchmark. The benchmark should be simple, and it should accurately measure file system performance. These goals oppose each other. Making the benchmark as simple as possible entails ignoring details of file system performance, while making the benchmark as accurate as possible requires consideration of performance details that complicate the benchmark.

The *IOStone* code is a compromise between the conflicting goals of simplicity and accuracy. It consists of a few hundred lines of code that simulate a typical file system workload. The *IOStone* code first creates a synthetic file system. It then performs a sequence of read and write accesses to this system and the time required to perform these accesses is measured. Finally, the synthetic file system is erased.

The *IOStone* code first creates a synthetic file system that consists of a collection of files that vary in size. The distribution of these file sizes corresponds to the distribution of run lengths measured in the BSD study. To ensure that the file system is not clustered onto several adjacent disk cylinders, eight spacer files, each 512K bytes in size, are interspersed between the reference files. The spacer files are also used to flush the reference files from the system disk cache after the entire file system is created. This is accomplished by reading the spacer files after file creation is complete.

Ideally a synthetic file system should be comprised of thousands of files that occupy a significant portion of available disk space. However, the *IOStone*'s synthetic file system is limited to 404 files that occupy about five megabytes. This limitation is imposed to keep the file system portable, and to limit the *IOStone* program's running time. Many systems do not have more than five megabytes of free disk space that can be used for the synthetic file system. Furthermore, it can take hours, an unacceptably long time, to create more than several hundred files.

The *IOStone* code generates a stream of file system references that exhibits the same locality that is present in measured file system workloads. This allows *IOStone* measurements to account for performance gains that arise from system enhancements that exploit locality such as disk caches. The disk cache simulation of the BSD study was used as a guide in structuring a synthetic reference stream with a typical locality profile. The disk cache simulation uses file system traces to drive a simulator that measures disk cache

performance. The simulation shows that the cache miss rate drops from 100% to about 25% as the disk cache size is increased from 0 to 1M byte. As the cache size increases further, the miss ratio approaches 20%.

To achieve an approximation of the BSD locality profile, the reference files are divided into four groups labeled *A, B, C* and *D*. Each group occupies approximately 256K bytes of storage. The *IOStone* code accesses files in groups. When group *A* is accessed, all of the files in group *A* are accessed in a pre-computed randomized order. Since the measured ratio of reads to writes is about 2:1, each access is randomly selected to be a read or a write with probability 2/3 or 1/3 respectively.

Consider the following pattern of group accesses: *A, A, B, B, C, C, D, D, A, A, B, B, C, C, D, D, A, A, B, B, C, C, D, D*. If the disk cache is less than 256K bytes in size, it is too small to contain all of the files in a single group. Consequently the hit rate will be very low since files will not be present in the disk cache on consecutive accesses to the same file group (e.g., {*A,A* }). With a 256 K byte disk cache, the hit rate is about 50% because the file group references occur in pairs ({*A,A*}, {*B,B*}, {*C,C*}, {*D,D*}). When a group is referenced for the first time, none of its files are present in cache so all references will cause cache misses. When a group is referenced the second time, its files are present in cache. If the disk cache is 1M byte or larger, the hit rate is about 83% because misses are only generated by initial accesses to each group. These hit rates approximate hit rates from the disk cache simulations of the BSD study.

IOStone indicates file system performance more accurately than simple maximum bandwidth or random access time measurements. Even a linear combination of maximum bandwidth and random access time cannot accurately characterize a file system's performance because the bandwidth a file system delivers varies non-linearly with the data transfer size[9].

IOSTONE MEASUREMENTS

The *IOStone* benchmark has been run on a range of machines. These are ranked by *IOStone* performance in Table 1. Most machines have their own disks with the exception of one of the Sparcstations. File requests for this diskless system are channeled through a 10 megabit Ethernet to a remote file server. Although the *IOStone* performance of the diskless Sparcstation is less than the performance of the Sparcstation with a disk, it still has better *IOStone* performance than the Vaxstation II, Apollo DN3000, or Apple Macintosh IIx. Also note that Sun workstations running the Sun OS 4.0x operating system do not have a fixed size disk cache. Disk cache space is allocated dynamically from

available real memory. Since the *IOStone* program was the only user process running when the measurements were made, a large amount of memory was free to be allocated to the disk cache. This resulted in a high level of performance. The Apollo Domain 10.1 operating system running on the DN3000 workstation has a similar 'dynamic' caching scheme but does not perform as well.

Table I. IOStone performance of various machines

Machine	Operating System	Disk Drive	Disk Controller	Block/Frag Size (bytes)	Disk Cache Size (bytes)	IOStones /second
Sparcstation 1	Sun OS 4.0.3c	Quantum	Sun 4 SCSI	8K/1K	variable	72383
Sun 3/80	Sun OS 4.0.3c	Quantum	Sun 3 SCSI	8K/1K	variable	64268
Decstation 3100	Ultrix 3.0	DEC RZ55	DEC SCSI	8K/1K	1.6 M	11151
Next N1000	Next Mach 1.0	Maxtor xt8760	Next	8K/1K	120 K	7589
Sun 2/120	Sun UNIX 4.0	Fujitsu M2321	Xylogics	8K/1K	358 K	5012
Sparcstation 1	Sun OS 4.0.3c	diskless	N/A	8K/1K	variable	4304
Vaxstation II	Ultrix-32 3.0	Micropolis 1325	DEC	8K/1K	1023 K	3857
Apollo DN3000	Apollo Domain 10.1	Micropolis 1325	Apollo st506	1K/1K	variable	2799
Macintosh IIx	Mac OS 6.03	Apple SCSI	Apple SCSI	8K/1K	0	1492

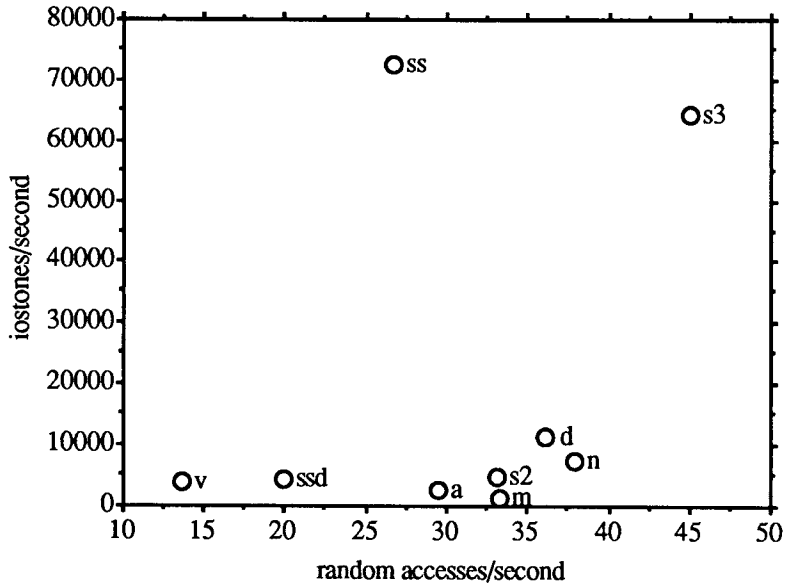

Figure 1. IOStone performance vs. random access speed. Legend: a-Apollo DN3000, d-Decstation 3100, m-Apple Macintosh IIx, n-Next N1000, s2-Sun 2/120, s3-Sun 3/80, ss-Sparcstation 1, ssd-diskless Sparcstation 1, v-Vaxstation II

A comparison of *IOStone* performance with random access speed appears in Figure 1. It shows that there is very little correlation between random access speed and *IOStone*

performance. This fact is surprising because a large number of small block transfers are performed by the *IOStone* code, and one would expect these small block transfers to approximate random accesses. This lack of correlation highlights the fact that random access speed is not a good indicator of overall file system performance.

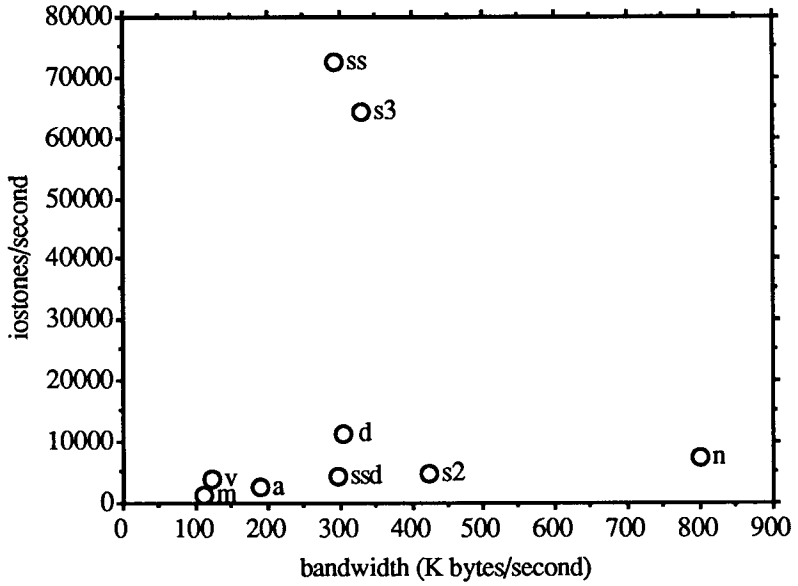

Figure 2. IOStone performance vs bandwidth. Legend: a-Apollo DN3000, d-Decstation 3100, m-Apple Macintosh IIx, n-Next N1000, s2-Sun 2/120, s3-Sun 3/80, ss-Sparcstation 1, ssd-diskless Sparcstation 1, v-Vaxstation II

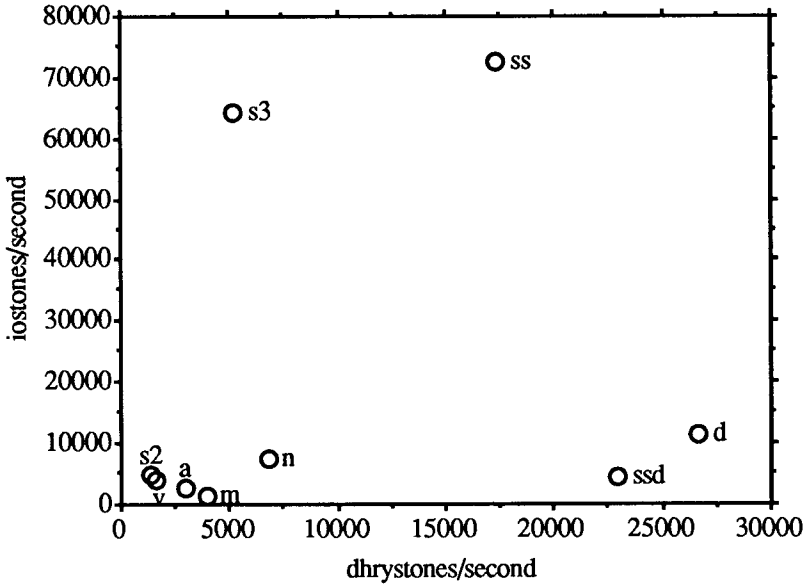

Figure 3. IOStone performance vs Dhrystone performance, legend: a-Apollo DN3000, d-Decstation 3100, m-Apple Macintosh IIx, n-Next N1000, s2-Sun 2/120, s3-Sun 3/80, ss-Sparcstation 1, ssd-diskless Sparcstation 1, v-Vaxstation II

A comparison of *IOStone* performance with maximum bandwidth on read operations is presented in Figure 2. As with the comparison with random access speed, there is very little correlation between *IOStone* performance and maximum read bandwidth. This highlights the fact that maximum read bandwidth is not a good indicator of overall file system performance either.

A comparison of *IOStone* performance against *Dhrystone* performance appears in Figure 3. Recall that the *Dhrystone* benchmark is a comprehensive measure of CPU performance on system tasks[2]. As can be seen from this graph, the balance between CPU performance and file system performance varies greatly from system to system. If there exists an optimal balance of file system performance to CPU performance, most of these systems do not achieve it.

LIMITATIONS

There are several limitations of the *IOStone* code. *IOStone* measures file system performance on a typical system workload that is determined by a set of performance measurements from business, research, and development sites. Large scientific applications exhibit different workload characteristics, so *IOStone* may not be an accurate indicator of file system performance on these tasks.

IOStone also does not measure paging performance. Paging activity contends for mass storage bandwidth with the file system accesses. However, measurements taken in the BSD study suggest that paging activity accounts for only a small fraction of total I/O system load. Additionally, few empirical studies measure paging activity[10], so there exists very little data on which to base a representative paging workload. For these reasons, *IOStone* concentrates exclusively on measuring file system performance.

IOStone accesses a synthetic file system that is limited to five megabytes in size to maintain an acceptable level of portability. One might argue that a system with an appropriately large disk cache may realize unrealistic performance gains from this restricted locality. However, the BSD study notes that file system requests tend to exhibit a great amount of locality. Thus, even a modest sized buffer cache greatly reduces the number of data transfers between mass storage and main memory. Furthermore, *IOStone* approximates the locality profile of file system references measured in the BSD study. If a computing system can exploit this locality profile, *IOStone* measurements should reflect this fact.

File creation and deletion operations are not performed in the benchmark. These operations do not occur during read operations which constitute the majority of transfers.

The BSD study indicates that only a small fraction of the write requests entail file creation or deletion operations, so neglecting these operations does not significantly reduce the accuracy of *IOStone*.

The *IOStone* code only writes to existing file blocks. This differs from file extension write operations that require allocation of new file blocks. The performance consequences of this omission are small, however, since the list of free blocks is frequently accessed and generally resides in the system disk cache. Therefore, allocating a new block typically entails only a simple main memory lookup.

CONCLUSIONS

The *IOStone* benchmark measures file system performance on a workload that is derived from measurements of real system workloads. Measurements made with *IOStone* indicate that file system performance of most machines is rarely well matched with processor performance. As processor performance continues to increase faster than file system performance, file system performance will become the major constraint on total system performance. Metrics that accurately assess file system performance, such as *IOStone,* will therefore assume increasing importance.

REFERENCES

1. H. J. Curnow and B. A. Wichman, 'A Synthetic Benchmark'", *Computer Journal*, **19**, (1), 43-49 (1976).
2. R. P. Weicker, 'Dhrystone: A Synthetic Systems Programming Benchmark', *CACM*, **27**, (10), 1013-1030 (1984).
3. J. Stubbs, NCR Corporation Advanced Development, San Diego, CA, netmail correspondence, November 1986.
4. B. L. Wolman and T. M. Olson, 'IOBENCH: A System Independent IO Benchmark', *Computer Architecture News*, **17**, (5), 55-70 (1989).
5. I. Hu, 'Measuring File Access Patterns in UNIX', ACM *SIGMETRICS Performance Evaluation Review*, **14**, (2), 15-20 (1986).
6. J. K. Ousterhout , H. D. Costa, D. Harrison, J. A. Kunze, M. Kupfer and J. G. Thompson, 'A Trace Driven Analysis of the UNIX 4.2 BSD File System', *Proceedings of the Tenth Symposium on Operating Systems Principles*, 15-24 (1985).
7. A. J. Smith, 'Sequentiality and Prefetching in Database Systems', *ACM Transactions on Database Systems*, **3**, (3), 223-247 (1978).
8. A. J. Smith, 'Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms', *IEEE Transactions on Software Engineering*, **SE-7**, (4), 403-417 (1981).
9. A. Park and R. J. Lipton, 'Models and Measurements of File System Performance', *Proceedings of the 18th Annual Pittsburgh Conference on Modeling and Simulation*, 867-876 (1987).
10. H. Garcia-Molina, A. Park, and L. R. Rogers, 'Performance Through Memory' *Proceedings of the 1987 ACM-SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 122-131 (1987).