[In Proc. 7th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VII), pp. 90-95. Rio Rico, AZ, March, 1999]

Brittle metrics in operating systems research

Jeffrey C. Mogul -- mogul@pa.dec.com
Compaq Computer Corporation Western Research Laboratory
250 University Avenue, Palo Alto, CA 94301

Abstract

Most operating systems research publications make claims about performance. We expect these performance claims to be both repeatable and relevant to important applications. We also expect them to be comparable to similar claims made in other papers. This implies the need for realistic and widely-used benchmarks.

Often, however, no such benchmark exists. The problem is especially acute in application areas with significant external latencies (such as Internet servers and file systems). Sometimes this leads researchers to measure only what is easily measured. Sometimes it leads to the naive use of unrealistic benchmarks, causing research to be diverted from solving actual problems.

A serious project to put operating systems research on a sound quantitative basis requires that we make an explicit effort to develop repeatable, comparable, and realistic metrics for evaluating operating systems research. We must also develop reliable techniques for using benchmark results to predict real-world performance.

1. Introduction

Computer systems research publications usually make quantitative claims about system performance. We call our field ‘‘computer science,’’ implying that we insist on sound scientific methods and principles, including the repeatability of measurements. We also realize that our field is an engineering discipline, and so we insist that such research must ultimately be relevant to the solution of some real-world problem.

Our demand for repeatable experimental measurements implies the need not only for proper scientific procedure; it also implies the need for common agreement on the metrics we use. If two researchers use different metrics to quantify their results, we cannot tell if one result is ‘‘better’’ than the other. Our demand for relevance additionally implies that the metrics that we use in research publications must be applicable to the ultimate real-world application domain.

Realistic and widely-used benchmarks, when applied properly, would satisfy these requirements. A benchmark is realistic in a strong sense if it predicts a system’s absolute performance in a production environment. A benchmark is realistic in a weaker sense if can only predict the relative real-life performance of two proposed solutions; weaker still if it only predicts their rank-order.

The mere fact that a benchmark gives quantified, repeatable, or statistically significant results does not necessarily prove that it has any value in predicting real-life performance. For example, one could measure with exquisite care and accuracy how fast several people can run a 100-meter footrace, but this would have very little value in predicting their success as professional basketball players. Basketball players do have to run quickly, but this is not their only necessary skill. Unfortunately, it isn’t always easy to determine whether a benchmark is realistic, or for what contexts.

If it’s not widely used, even a realistic benchmark fails to address the need for comparison between different research publications. Many things can prevent the widespread use of a benchmark: lack of access to the software, difficulty in its use, the need for elaborate test environments, or simple apathy.

In computer systems, as one ascends the levels of system complexity and generality, good benchmarks become increasingly difficult to define and construct. We have a solid ability to measure the cost of an inner loop in an array multiplication; we can do a pretty good job of measuring TCP bandwidth over a LAN; with a little more effort, we can measure the performance of simple database transactions.

In fact, what people call the ‘‘RISC revolution’’ was in reality more of a ‘‘quantitative methods in computer architecture’’ revolution. Led in part by John Hennessy and David Patterson [10], this represented a shift from seat-of-the-pants computer architecture design to an approach based on measuring what real programs required. Computer architecture publications typically rely upon a small set of widely-used benchmarks.

But when it comes to operating systems, things get ugly. We take systems intended to be used in complex, unpredictable environments, and we measure them in artificial and oversimplified ways. Worse, it’s hard to find two researchers who use the same approach. Our metrics are brittle: they appear firm but their predictions fail in unexpected ways.

I am not arguing that operating systems researchers are foolish or dishonest. Good operating systems benchmarks are indeed hard to define and construct, and most researchers understandably put much more intellectual effort into the systems under test, rather than the testing procedure. *If decent benchmarks were widely available, researchers would use them.*

My goal for this position paper is to analyze the problem of operating system benchmarks, and to encourage the development (and use) of realistic benchmarks. I will also stress the necessity of developing sound techniques for using benchmark results to predict real-world performance.

2. The state of the problem

My concerns about benchmark realism have been triggered by several observations. First, since I do research for a company that sells computer systems, I am often asked to predict how a proposed system will perform in real use, or to deduce why a installed system is performing badly. The gap between benchmark behavior and real-life behavior is often painfully obvious. For example, I know of no file system benchmark that can predict the excruciatingly long time sometimes required to rebuild the ''history'' database of a USENET server; random-access I/O to a file of this size (hundreds of Mbytes) just isn't reflected in existing benchmarks.

Second, I have often read (or reviewed) research papers that make claims based on inappropriate or incomparable benchmarks. That is, either the benchmark is insufficiently connected to full-scale real-life applications, or it's used only in one paper, and so does not allow comparison against other papers. (I've committed both of these errors myself.)

2.1. Categories of benchmarks

Benchmarks seem to fall into four categories:

1. **Official benchmarks**, where some attempt has been made to construct a useful, realistic benchmark that will be widely employed. These fall into two sub-categories:
   - **Vendor** benchmarks, intended to prevent atrocities in the marketplace.
   - **Research** benchmarks, intended to further the scientific process.
2. **Real users**, whose activities are either encapsulated in a trace, or studied in actual use of the system under test.
3. **Microbenchmarks**, constructed to focus on the performance of an isolated component of a system, rather than its complete performance on real problems.
4. **Run a bunch of programs**, an *ad hoc* approach in which the experimenters choose some programs that seem interesting.

Official benchmarks are sometimes the most realistic approach. However, vendor benchmarks are typically compromises between vendors with competing goals; they favor simplicity over realism, since complexity can lead to loopholes. Vendor benchmarks also favor the generation of a single figure of merit, rather than a multidimensioned vector (which might be more useful in making predictions).

Subjecting real users to a system under test (or vice versa) meets the realism test, if done carefully. Unfortunately, it tends to compromise repeatability. The use of traces can improves repeatability, but ignores the possibility that improved (or worsened) performance might cause users to change their behavior, invalidating the trace's realism. Sharing of traces can also run afoul of privacy concerns.

Microbenchmarks concisely identify the causes of performance problems, but they are not so good for predicting the performance of an entire system in actual use. Complex systems, especially, display performance interactions (such as cache conflicts) that microbenchmarks explicitly ignore [4]. Trying to fix this by using multiple microbenchmarks doesn't work, since it doesn't test their interactions.

The use of an *ad hoc* selection of programs to measure system performance is not inherently evil. For example, a paper about how an operating system fails to support a particular class of program should clearly report on the performance of such a program. But if a paper makes a more general point, then the use of *ad hoc* applications as benchmarks can be misleading, if these applications are not actually representative. It also makes comparison with other papers much harder.

2.2. Why existing benchmarks don't suffice

Most existing benchmarks, while usually reproducible, are not realistic. I will discuss a few of the most widely used benchmarks, although space constraints lead me to oversimplify. Endo *et al.* [9] provide a similar discussion, with different emphasis.

2.2.1. Vendor benchmarks

Vendor benchmarks that measure operating system performance include AIM (from AIM Technology), and SYSmark (from the Business Applications Performance Corp.), and SPECweb96 and SFS97 (LADDIS) from the Standard Performance Evaluation Corporation (SPEC).

AIM measures various aspects of UNIX and NT system performance, while SYSmark measures performance on DOS/Windows productivity applications. It's not clear whether SYSmark is intended to measure operating system performance, in part because it runs on the operating systems of exactly one vendor, but it does measure real applications. AIM, in contrast, runs on lots of operating systems, but measures a workload comprising a mixture of microbenchmarks, not real applications.

SPECweb96 measures full-system performance with a real HTTP server application, but in an unrealistic LAN environment. While SPECweb96 was designed to scale the ‘‘working set’’ of data files as the operation rate grows, in an attempt to include a dependency file system performance, all vendors simply use enough RAM to cache the entire working set. Gaurav Banga and Peter Druschel have shown that standard HTTP benchmark clients (such as SPECweb96) behave unrealistically compared to actual Internet loads [2], and Gaurav and I have shown that this is especially problematic in WAN environments [3].

SFS97 (LADDIS) measures NFS server performance, by generating NFS client RPCs. It therefore cannot be used to compare NFS against any other protocol (or even the two common versions of NFS against each other), nor can it be used to test client-side caching strategies.

The database community uses the TPC benchmarks (from the Transaction Processing-performance Council). These have some useful properties; for example, the load-up-on-RAM approach used for SPECweb96 trials is discouraged in TPC, because the results include the costs of ownership. But this is primarily an application benchmark, since high-performance database software mostly avoids the operating system.

2.2.2. Research benchmarks

Researchers and experimenters have developed a number of benchmarks, for various topic areas, including:

- **Kernel microbenchmark suites:** the ‘‘Ousterhout benchmarks’’ [17], lmbench [15], hbench:OS [5].
- **File systems:** the Modified Andrew Benchmark (MAB) [17], Chen and Patterson’s self-scaling benchmark [8], Bonnie, IObench, IOZone.
- **Network throughput and latency:** netperf [14], ttcp.

Microbenchmarks are widely employed because they are easy to use, they provide simple numeric results (seldom requiring statistical analysis), and because their results are easily correlated with implementation features. Brown and Seltzer have pointed out, however, that creating a reproducible and statistically valid microbenchmark is not as trivial as it seems [5]. Perhaps more important, I know of no systematic work (except for prelimary work by Seltzer *et al.* [19]) that has measured the correlation between an operating system’s microbenchmark results and that system’s performance for complex applications. In other words, the predictive value of microbenchmarks is questionable.

Many of the file systems benchmarks are also, in effect, sets of microbenchmarks. Some concentrate more on the underlying storage system rather than the operating system. One common technique is to use a file large enough to defeat the operating system’s file cache, which ensures that such a benchmark won’t reflect how a system’s caching mechanisms work in real-life applications. Chen and Patterson analyze the drawbacks in many I/O benchmarks, pointing out that many ‘‘do not help in understanding system performance’’ [7].

Smith and Seltzer show that the conditions under which file system benchmarks are run can have significant effects; in particular, use of a newly-initialized file system often makes the results look a lot better than they would be on a real-life system [21]. And I know of no benchmarks that attempt to account for scaling in the number of directory entries, even though many applications (USENET, email, proxy caches) create large directories full of small files.

Network performance benchmarks typically measure either throughput or latency, or both. They are usually executed in a LAN environment, in part because it’s hard to get repeatable results in a WAN. Also, while the interesting problems in network protocol research involve wide-area and/or low-bandwidth networks, the use of LANs in distributed systems (DSM, RPC, etc.) causes most operating systems researchers to focus on LANs. There is nothing inherently wrong with this, and there is a strong correlation between message latency and (for example) DSM performance [11]. However, those operating systems researchers who make claims regarding WAN applications (such as HTTP servers) seem to ignore the special characteristics of WANs, especially the long connection durations.

2.3. Emerging realistic benchmarks

I do not want to give the impression that nobody is working on improved benchmark realism. In fact, several researcher groups have addressed this problem.

Chen and Patterson presented a self-scaling I/O benchmark [8], and explicitly addressed the problem of estimating system performance on workloads other than the benchmark. Their benchmark focuses more on the I/O subsystem than on file system operations (such as directory manipulation).

Wireless mobile networks present an especially difficult benchmarking problem, because in the real world, the terminals move around in complex physical environments, and in order to repeat a measurement, one would apparently have to faithfully follow the same path. Noble *et al.* rescue mobile-networking researchers from this tedious task by introducing *trace modulation* [16]. This technique starts by recording a set of traces while an instrumented host traverses one or more paths, then extracts parameters that characterize these traces. The parameters are then used to perturb a LAN-based network, to simulate the observed wireless network.

While not as glamorous as HTTP, USENET news serves millions of users. ISPs report serious performance problems with USENET software. Scaling USENET server performance is, in some ways, much harder than scaling HTTP server performance because the client software expects to see a compact sequence of *article IDs*, which are assigned locally at each server, and thus represent a serious synchronization task. There does not appear to be a widely available USENET benchmark, but Yasushi Saito, working with Ben Verghese and myself, has produced a first cut at a realistic USENET benchmark [18]. His *newsbench* program, based on traces made of a moderately busy server, attempts to replicate the server's actual client load.

As noted earlier, Gaurav Banga and Peter Druschel addressed the lack of realism in HTTP server benchmarks, and also developed an improved benchmark client [2]. Similarly, Jussara Almeida and Pei Cao have been developing the *Wisconsin Proxy Benchmark* for measuring Web proxy server performance [6]. Proxy cache benchmarking is more difficult than simple server benchmarking, because one must emulate realistic server-access delays and realistic reference stream locality.

2.4. Statistics from recent conferences

What do operating systems publications actually use for benchmarks? I examined the proceedings for two recent and prestigious conferences, SOSP-16 (1997) and OSDI-2 (1996). (Disclaimer: as a PC member for both conferences, I might have influenced the results, but I do not recall any particular biases regarding benchmarks.) I put each paper into a loose taxonomy of topic areas, and then tried to discern what benchmarks were actually used[1]. This is a tedious and error-prone process, since authors are inconsistent in describing benchmark sets. I did a similar analysis for ASPLOS-8 (1998), to test the hypothesis that computer architects are more rigorous.

Figures 3-1 and 3-2 summarize the results for ‘‘operating systems’’ papers and ‘‘computer architecture’’ papers, respectively. Almost all of these papers do make specific performance claims. All 19 OSDI-2 papers and 28 ASPLOS-8 papers do so, as well as 20 of the 23 SOSP-16 papers.

Papers in areas with widely available benchmarks seem to use them. For example, most papers about uniprocessor architecture and memory hierarchies use versions of the SPEC CPU benchmarks. Papers about multiprocessor systems and parallel programming mostly use a small set of parallel benchmark suites (SPLASH-2, Nasa7), although not as religiously.

Papers in mainstream topics of operating systems research are fuzzier. The problem is most obvious for file systems papers, where the eight papers share not one benchmark in common. Further, the ‘‘benchmarks’’ used in these papers, for the most part, are not very realistic. The Modified Andrew Benchmark is about the best of the bunch, but it's about a decade old and has a tiny working set. The Connectathon benchmark is effectively a collection of microbenchmarks.

Similarly, the five papers on operating system structures do not share a single benchmark[2], nor do the three papers on extensible kernels (except that two of these use vaguely similar microbenchmarks). The inconsistency in benchmark choice does not directly affect predictive power, but it certainly makes comparison harder.

Four papers assert improved HTTP server performance, but all four measure this performance in LAN environments, with simplistic benchmarks; as I argued above, this is not at all realistic. The authors of OSDI-2 and SOSP-16 papers might not have been aware of that realism problem, but why did not of them use an existing standard benchmark (e.g., SPECWeb96 or WebStone).

Five papers on scheduling use overlapping benchmarks in a few cases[3]. Measurement of scheduling performance, and that of interactive user interfaces, is inherently subjective: one can choose or ignore numerous dimensions. One of the OSDI papers directly tackles the problem of interactive applications, and perhaps we will see increasing maturity in this area.

3. Recommendations

I believe that the key challenge in quantitative operating systems research is to improve our ability to predict real-life system performance. We can make statistically rigorous and repeatable measurements of our systems, but we lack the ability to base useful predictions on these measurements.

This is a hard problem. It goes against our preference for simple, neat, objective results. But without solving this problem, our field risks irrelevance. Recent operating systems research has been plagued by the extreme dif-

[1] Small *et al.* made a similar study of a broader set of conferences [20].

[2] One could quibble that hbench:OS is simply an improved version of lmbench, although hbench:OS was created, in large part, because lmbench is not sufficiently reproducible.

[3] It is not clear if Dhrystone is a useful benchmark for scheduler performance, since it is no longer viewed as being useful for anything else.

ficulty of converting our novel ideas into systems deployed outside computer science departments. Perhaps it is time to admit this and increase our emphasis on the prediction problem, which has clear utility even in a world that resists innovation.

We can start by encouraging graduate students, and other researchers, to work in this area[4]. If it's a hard problem, then solutions should be worthy of recognition via doctorates and refereed publications. It might also be a good way to shake loose some corporate grant money. Given that computer company profits come increasingly from system integration charges, rather than selling hardware, the ability to accurately predict performance translates directly into higher profits.

We also need to work to define a shared set of interesting problems, to motivate and channel such research. For example, virtually all published measurements of Web servers assume the use of static pages (but see [13]), but many high-volume sites (especially electronic commerce) use mostly dynamic content, which is far less efficient. The ability to predict the performance of servers generating dynamic content would be a major victory, but today it's not even a well-defined problem. Can we characterize the performance requirements of a ''typical'' commerce site?

There is often a tension between realism and reproducibility; real-world effects can make it hard to repeat experiments exactly. We should strive for reproducibility in realistic benchmarks; otherwise, they would have little predictive value. But we must also concede that realistic benchmarks might not generate exactly repeatable results as often as synthetic benchmarks can.

Given the difficulty of getting access to high-volume real-life sites, traces will probably remain the most feasible method for sharing research results. We need to build up a much larger and broader set of public trace archives. The Internet Traffic Archive [12], which contains several sets of packet traces and HTTP server logs, shows how this might be set up, but it needs larger and more up-to-date traces. We also need modern file-system traces to replace the now-ancient Sprite traces [1]. We need to establish guidelines for trace formats, and tools to anonymize traces without removing the useful information.

More and better traces, however, cannot substitute for better benchmark design. Traces can increase the realism of a request stream, but they do not affect the realism of the execution environment. Without both kinds of realism, one cannot rely on a benchmark to predict real-world performance.

[4]Margo Seltzer has students working on this [19], although I believe their approach discounts the need for environmental realism.

Topic area	# of papers	Benchmarks used
Database	1	TPC-B like, TPC-C like
Dist. systems	3	BXMH, simple HTTP (2), Web server traces
DSM	2	SPLASH2, TPC-D, microbenchmarks, pieces of various parallel suites
Extensible kernels	3	UDP latency, X11server (adhoc), microbenchmarks (2), microbenchmarks of packet filters
File systems	8	Connectathon, MAB, NAS, TPC-B like, UCB user traces, browser traces, convolution, cscope, datacube, dinero, glimpse, image proc, ld, modified grep, postgres-join, postgres-select, read bw, sort, Sprite traces, SQL, write bandwidth, xds
Mobile code	1	BIT, Hanoi, JHLZip, JavaCup, Jess, TestDES
Mobility	2	actual users, browser, speech recognizer, video player
Networks	4	simple HTTP, MPEG player, UDP latency (2), bandwidth (3), msg latency, synthetic (2)
Objects	1	OO7
Parallel programming	3	FFT, LU, SOR (3), SPLASH-2 Raytrace, SPLASH-2 Water (3), TSP (2)
Performance studies	2	Powerpoint, Word, microbenchmarks, traces of (SPEC92 ear, SPEC92 tomcatv, SQL server, compiler)
Scheduling	5	configure, dhrystone (2), gcc, gzip, media player (2), microbenchmarks, synthetic, Tiger, type-in emulator
OS structure	5	AIM, simple HTTP, SPLASH2 raytrace, Sybase simple, TCP microbenchmarks, hbench:OS, kernel compile, lmbench, microbenchmarks, pmake, unix apps, verilog
Synchronization	1	microbenchmarks
Tools	3	TPC-D, AltaVista, gcc, gzip, McCalpin, MPEG, sort, SPEC92, SPEC95 (2), X11perf

Table 3-1: Statistics from operating systems papers

Topic area	# of papers	Benchmarks used
Branch prediction	1	chess, groff, gs, pgp, plot, python, SPEC95, ss, tex
CPU design	3	khoros, perfect, SPEC95 (3)
ILP	2	Nasa7, rawbench, SPEC92, SPEC95
I/O	1	TCP bandwidth, UDP bandwidth
Memory hierarchies	11	Olden pointer benchmark, SPLASH2 (2), TPC-C, awk, deltablue, espresso, groff, loop nests, merge, numeric kernels, particles, photo, sed, SPEC89, SPEC92, SPEC95 (6), tex, typecheck, wave5, wc, yacc
Multiprocessors	2	TPC-D, file copy, flashlite, ocean, pmake, vcs
Optimization	2	chameleon, espresso, gs, protein string matching, sis, stencil, vis, yacr
Simulation	1	SPEC95
Translators	1	perl, SPEC95

Table 3-2: Statistics from computer architecture papers

Acknowledgments

I would like to thank the HotOS referees for their comments, and also Margo Seltzer, for taking not quite the same point of view as I did.

References

[1] M. Baker, J. Hartman, M. Kupfer, K. Shirriff and J. Ousterhout. Measurements of a Distributed File System. In *Proc. 13th Symp. on Operating System Principles*, pp. 198-212. Pacific Grove, CA, Oct., 1991. Traces at http://now.cs.berkeley.edu/Xfs/SpriteTraces/sprite_traces.html.

[2] G. Banga and P. Druschel. Measuring the Capacity of a Web Server. In *Proc. USENIX Symp. on Internet Technologies and Systems*, pp. 61-71. Monterey, CA, Dec., 1997.

[3] G. Banga and J. Mogul. Scalable kernel performance for Internet servers under realistic loads. In *Proc. 1998 USENIX Annual Technical Conf.*, pp. 1-12. USENIX, New Orleans, LA, June, 1998.

[4] B. Bershad, R. Draves, and A. Forin. Using Microbenchmarks to Evaluate System Performance. In *Proc. Third Workshop on Workstation Operating Systems*, pp. 148-153. IEEE Technical Committee on Operating Systems, Key Biscayne, FL, Apr., 1992.

[5] A. Brown and M. Seltzer. Operating System Benchmarking in the Wake of Lmbench: A Case Study of the Performance of NetBSD on the Intel x86 Architecture. In *Proc. 1997 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pp. 214-224. Seattle, WA, June, 1997.

[6] P. Cao. Characterization of Web Proxy Traffic and Wisconsin Proxy Benchmark 2.0. In *Web Characterization Workshop*. World Wide Web Consortium, Cambridge, MA, Nov., 1998. http://www.cs.wisc.edu/~cao/w3c-webchar-position/.

[7] P. Chen and D. Patterson. Storage Performance--Metrics and Benchmarks. *Proceedings of the IEEE* 81(8):1151-1165, Aug., 1993.

[8] P. Chen and D. Patterson. A New Approach to I/O Performance Evaluation -- Self-Scaling I/O Benchmarks, Predicted I/O Performance. *TOCS* 12(4):308-339, 1994.

[9] Y. Endo, Z. Wang, J. B. Chen, and M. Seltzer. Using Latency to Evaluate Interactive System Performance. In *Proc. 2nd Symp. on Operating Systems Design and Implementation*, pp. 185-199. USENIX, Seattle, WA, Oct., 1996.

[10] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, San Mateo, CA, 1990.

[11] C. Holt, M. Heinrich, J. P. Singh, E. Rothberg, and J. Hennessy. *The Effects of Latency, Occupancy, and Bandwidth in Distributed Shared Memory Multiprocessors*. Tech. Report CSL-TR-95-660, Computer Science Laboratory, Stanford University, Jan., 1995.

[12] Internet Traffic Archive. http://ita.ee.lbl.gov/index.html.

[13] A. Iyengar and J. Challenger. Improving Web Server Performance by Caching Dynamic Data. In *Proc. USENIX Symp. on Internet Technologies and Systems*, pp. 49-60. Monterey, CA, Dec., 1997.

[14] R. Jones. The Public Netperf Homepage. http://www.netperf.org/.

[15] L. McVoy and C. Staelin. lmbench: Portable Tools for Perfor mance Analysis. In *Proc. 1996 USENIX Technical Conf.*, pp. 279-295. San Diego, CA, Jan., 1996.

[16] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz. Trace-Based Mobile Network Emulation. In *Proc. ACM SIGCOMM '97*, pp. 51-61. Cannes, France, Sep., 1997.

[17] J. Ousterhout. Why Aren't Operating Systems Getting Faster As Fast as Hardware? In *Proc. Summer 1990 USENIX Conference*, pp. 247-256. Anaheim, CA, June, 1990.

[18] Y. Saito. A Usenet Performance Study. http://www.research.digital.com/wrl/projects/newsbench/. 1998

[19] M. Seltzer, D. Krinsky, K. Smith, and X. Zhang. The Case for Application-Specific Benchmarking. In *Proc. HotOS-VII*. Rio Rico, AZ, Mar., 1999.

[20] C. Small, N. Ghosh, H. Saleed, M. Seltzer, and K. Smith. *Does Systems Research Measure Up?*. Tech. Rep. TR-16-97, Harvard University Computer Science, Nov., 1997. ftp://ftp.das.harvard.edu/techreports/tr-16-97.ps.gz.

[21] K. Smith and M. Seltzer. File System Aging -- Increasing the Relevance of File System Benchmarks. In *Proc. 1997 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pp. 203-213. Seattle, WA, June, 1997.