The Need for Measured Data in Computer System Performance Analysis or Garbage In, Garbage Out*

Alan Jay Smith
Computer Science Division
EECS Department
University of California
Berkeley, California 94720-1776, USA
smith@cs.berkeley.edu

Abstract

There are three general techniques for studying the design, operation and optimization of any type of system: implementation, modeling, and simulation. Implementation can be used when the time is available and the effort is justified. Mathematical modeling can be used if the system is simple enough or if gross approximations are acceptable. Simulation is needed for the remaining cases.

In all three cases, a crucial element is the need for accurate estimates of the "workload." For example, for a computer main or cache memory system, the workload may at one level of detail be the timing and sequence of memory addresses referenced; at another level of detail, it could be just the ratio of loads to stores. No matter how accurate the model of a system may be, any results from it are useless if the workload is not accurately represented.

In this short paper, we discuss various types of models and when and what type of accurate workload data is needed. We pay particular attention to performance analysis of memory hierarchies and the use of trace driven simulation.

*The author's research is supported in part by the National Science Foundation under grants MIP-9116578 and CCR-9117028, by NASA under Grant NCC 2-550, by the State of California under the MICRO program, and is or has been recently supported by Sun Microsystems, Apple Computer Corporation, Mitsubishi Electric Research Laboratories. Philips Laboratories, Digital Equipment Corporation, and the International Business Machines Corporation.
This paper appeared in the proceedings of the Compsac'94 Conference, Taipei, Taiwan, November, 1994.

Keywords: performance analysis, trace driven simulation, memory hierarchies, cache memory, main memory, disk caching, file migration, database locking, benchmarking.

1. Introduction

There are three techniques for studying the design, operation and optimization of any type of system [Ferr78]. Implementation is the construction of a version, or at least a prototype, of the target system. Implementation has the substantial virtue of being "real" (but see below), at the cost of substantial time and effort. In some of areas of computer science research, research work is often only taken seriously when proposals are actually implemented, whether or not implementation is the best or most valid approach.

At the other extreme is the mathematical modeling of a system. Such a model can be a mathematically complex queueing model, or may be a simple back-of-the-envelope calculation. Such models are often relatively easy to generate, but they can capture only a very limited degree of complexity, and are usually very limited in the range of assumptions they can incorporate - e.g. exponential distributions for parameters are frequently required, most all or events must be independent, simultaneous resource possession cannot be modeled, etc.

In cases when implementation is not justified or feasible, and when mathematical models are inadequate, simulation [MacD70] must be used. Simulations may be written at any desired level of detail, up to and including something that differs little from an implementation. Simulations may also be categorized as "trace driven" and random-number driven. *Trace driven simulations* [Sher72] are driven by a trace of some set of relevant events, measured from an environment believed to represent the one to be experienced by the target system. (Note that one must be careful that the trace is still valid when used to drive a system model significantly different than the system from which it was gathered.) In the absence of a trace, a random number generator is typically used to provide the relevant events. Such a random number generator may, however, fail to incorporate important aspects of the real workload; for example, a synthetic trace of memory addresses may fail to correctly represent locality of reference.

The weakest aspect of any of these three approaches to modeling is the dependence of the results on the "workload;" we use the term workload to include the actual (usually 'toy') workload on a prototype implementation, the traces used to drive a trace driven simulation, and the parameter values used for a mathematical model or a random number driven simulation. To the extent that the mathematical model or random number driven simulation fail to incorporate aspects of the behavior of the real workload, the inaccuracy of the result can be considered a workload problem.

It is important to emphasize the point that a prototype (or toy) implementation is often of far less value for some studies than a good trace driven simulation. There are several problems with a prototype implementation: First, it is usually not fully debugged, doesn't provide full functionality, and often doesn't offer features necessary for a commercial system such as reliability and recovery. Since the prototype isn't of commercial quality, it may exhibit significantly different behavior. Second, the workload is often grossly inadequate to test the research ideas. For example, a file system design exercised by routine graduate student workstation use is hardly representative of a commercial environment. Finally, the time, cost and effort for such an implementation is often disproportionate to the insight to be gained from it.

There are two other cases in which implementation can be the proper approach to research. First is the case where an existing system (e.g. Unix) can be modified or extended, with a relatively modest amount of work, to incorporate the new ideas, features, or measuring tools. The second case is that in which the research study can be "piggy-backed" on a full fledged commercial development effort; in that case, the system being developed is indeed real. In both of these cases, however, the results are still sensitive to the workload.

In the remainder of this paper, we discuss various types of research studies with which the author is familiar, and consider the extent to which workload and measurement data are needed.

2. Cache Memory and Main Memory Studies

Cache and main memory studies [Smit82] can be classified into three categories: Gross sizing and capacity studies have to do with selecting the parameters for a real implementation. In this case, what is needed are basic parameter values: frequencies of reads and writes, miss ratios, acceptable delays for access, rates of reference, etc. These figures need to be obtained from traces, from measurements of similar systems, or from systems whose results can be extrapolated; a paper with this type of data is [Smit85b]. Another example is figure 33 in [Smit82], which presents hardware monitor measurements of miss ratios for various cache sizes and associativities. Once parameter values are known, the necessary 'modeling' can often be done on the back of an envelope because of the grossness of the approximations involved. The critical issue here is that the parameter values be reasonably accurate. Some parameters (e.g. the ratio of reads to writes) are available from traces; for others, (e.g. miss ratios as a function of cache size), accurate values can only be obtained by hardware measurement of real workloads in a production environment.

Algorithm and detailed design studies usually must be done with trace driven simulation, using real traces of representative workloads. Such traces are needed because synthetic reference patterns are unlikely to correctly reflect important aspects of reference behavior such as spatial and temporal locality. Examples of these studies include a study of the effect of line size on cache performance [Smit87], of associativity on the cache miss ratio [Hill89], of interleaving on memory access delays [Bask76], and of the comparative behavior of various consistency algorithms for multiprocessor systems [Gee93]. Note that when the necessary traces are not available, studies using a workload model can be useful [Arch86]; at the present time, a study that compared multiprocessor cache consistency algorithms without using traces would almost certainly be useless.

The third case is that of detailed implementation studies - e.g. the effect of split transactions on a memory bus. In this case, detailed simulations are usually needed, but in some cases, it may be sufficient to drive them with random number generated events rather than traces. In some such cases, timing information is the most critical need, rather than reference patterns. This third case may contain instances for which synthetic traces could be useful; in this case, it is the details of the implementation, not the nuances and subtleties of the workload that are crucial, and rough approximations to the workload (i.e. crash dummies, not people) may well be sufficient.

An important limitation in some types of trace driven simulations is that the number of events needed to achieve some necessary state (e.g. a full cache or a given number of page faults) may be prohibitive. In that case, filtered or reduced traces [Smit77] may be needed. Such filtered traces might, for example, include only references to not recently referenced blocks, or might include only page faults or TLB misses.

3. Disk and I/O Studies

There are several categories of disk and I/O system studies. We comment on each individually. Note that studies in these areas are similar in many ways to studies of main and cache memory, and the relationships described above between type of study and appropriate data and techniques generally apply here as well.

3.1. Disk Cache

Studies of algorithms for disk cache [Smit85a, Thom87] (e.g. replacement algorithm, effect of block size, location of the cache, effect of prefetch, etc.) generally require traces; there is no other way to determine the effect of varying the parameters of interest. Gross sizing, bandwidth and capacity studies are similar to the situation above for cache and main memory studies. Simple parameter values are needed and may be obtained from various types of measurements; the modeling for such studies is usually very simple.

3.2. Disk Optimization

Studies in this area include studies of disk arm scheduling, block placement, block size, use of overflow blocks, etc. [Smit81d]. Such studies generally require traces. An example of the problems that occur when traces are not used is evident from the large number of early studies of disk seek scheduling which assumed random requests to the disk surface [Smit81c]; in fact, the probability of rereferencing the same cylinder in most systems is quite high [Lync72].

3.3. I/O Configuration Management

I/O configuration management has to do with the configuration of I/O systems on mainframe computers; i.e. how many disks, controllers, channels, how should they be interconnected, how should the load among them be balanced, etc. [Smit79]. The necessary analysis can usually be done with queueing models, provided that parameter values are available. For example, the products of BGS Systems Inc. both collect the necessary data and do the necessary analysis; the data collection and reduction is by far the hardest part.

3.4. Database Locking

Database locking studies consider the effectiveness of various locking algorithms in terms of the overhead for their implementation, and the extent to which they impede execution by causing transactions to block. Because trace data is so hard to obtain, the large majority of studies in this area have been done using mathematical models. Traces are needed, however, to ensure the validity of the results. For example, [Sing94] shows that many of the assumptions frequently used in database locking studies (independence, time invariance, uniformity, etc.) are invalid, and that predictions from some of those studies may be in error.

3.5. File Migration

File migration research has to do with the selection of algorithms for the management of tertiary storage. This includes algorithms for moving (replacing) files to tertiary from secondary storage, fetching them from tertiary to secondary storage, moving files between nodes in a distributed system, and for organizing the placement of files on the very non-random-access tertiary storage media. Traces are necessary for most of these algorithmic studies [Smit81a,b, Kure88], although random number driven simulation or models can be used as a first approximation for some studies of media management.

4. CPU Performance Evaluation

CPU performance evaluation is the study of the design and implementation of the CPU; e.g. the effect of data path width, adder size, multiplier speed, register set size, BTB design [Lee84, MacD84, Peut77], etc. First order performance estimates for some items can be made with knowledge of basic parameters, such as the frequency of multiplies. More detailed studies, such as the effect of pipeline design or the need for bypasses around the ALU, should be conducted using traces. Short traces are sufficient for some studies, such as pipeline design, but very long traces may be needed in other cases (BTB design, register set size).

Note that for CPU performance analysis, an accurate model of the design under study is crucial. The personal experience of the author is that even the designers and implementers of a new CPU are unable to tell the performance analyst how to accurately model the CPU pipeline.

5. Benchmarking

Benchmarking is the process of measuring computer system performance by running real or synthetic programs. Traditional benchmarking has been limited to that effort alone. More recent work [Saav89,92a,b,c] involves the detailed measurement of the components of both the system and the workload; i.e. an analysis of CPU performance on individual workload operations, a measurement of the performance of the CPU memory system, an evaluation of compiler effectiveness, and measurements of the type of operations actually executed in the workload. The type of data gathered in the more recent studies is invaluable in the analysis of CPU and system performance.

Any benchmarking effort is highly sensitive to the workload. Different programs exercise different aspects of a machine (e.g. floating point, integer), and some programs may be especially suited or unsuitable for a given architecture. Measurements taken on a PC are unlikely to be useful for predicting the performance of a vectorized workload on a Cray, and software for the PC is unlikely to be suited to the Cray.

6. Measurements

The purpose of the above discussion has been to emphasize the need for and importance of measurements and workload data as a necessary component of any modeling or simulation study. Despite its evident importance, a glance at the literature will show a disappointingly low frequency of studies which make sufficient use of real measurements, although the situation is steadily improving. The reason for this is the extremely difficulty of obtaining appropriate data [Zhou85]. Getting I/O traces at the necessary level of detail usually takes a man(person)-year or more. Analysis of CPU and workload performance took years [Saav92c]. Collecting parallel system traces took a considerable amount of effort [Gee93]. As a general rule, collecting the necessary data and reducing it to usable form is as much or more work than the study for which it is destined. Note, however, that once collected, workload data or traces can be used in many studies and the collection effort can be amortized over many papers, many students and many products.

In the remainder of this section, we comment briefly on methods for trace generation.

6.1. Program Address and Instruction Traces

There are several techniques for generating traces for uniprocessor programs. (a) A hardware monitor can be used to pick addresses off of the address pins of the processor, provided that there is no on-chip cache, and that 'spurious' references (prefetches) are desired. (b) An interpreter for the machine architecture can be used to interpretively execute the target program. (c) A "trace trap" mode can be used to cause a trap after each instruction. (d) An "execute" instruction (available on the IBM 370 architecture) can be used to interpretively execute a program. (e) The object code can be instrumented to generate trace records for every load, store and instruction fetch (or branch); note that in this case, addresses must be 'fixed up' to compensate for the tracing code. (f) On a microcoded machine, the microcode can be modified.

Note that while all of the above techniques can also be used for parallel programs, steps must be taken to properly trace each thread, and maintain a consistent clock among the threads.

6.2. I/O and File Traces

Generating I/O and file traces is difficult. There are two general techniques: (a) Instrument the operating system to generate trace records at appropriate times. (b) Use the output of a standard trace or debugging package such as IBM's GTF.

The problems with the former are: (a) Instrumenting the OS requires considerable specialized knowledge and effort. (b) Collecting data requires access to and permission to modify the OS of a machine running a real, live workload. (c) The data desired is seldom available in one piece of OS code (e.g. file names and disk addresses are generally not available in the same pieces of code), so many different places in the OS must be modified, and then the data tied together. A major difficulty is that it may be very difficult to match up records with file names generated in one part of the OS code with corresponding records that contain physical disk addresses generated elsewhere.

The problem with the use of a standard trace package is that it seldom generates what is needed, and often generates much that isn't needed, placing a very heavy overhead load on the system.

For file migration purposes, it is often sufficient to know only if a file has been used on a given day. In that case, gathering data can be very easy-just read the file system directory every night and save information on which files have (or have not) been used in the preceding 24 hours.

6.3. Trace Validity and Stability Over Time

There are several issues about the validity of traces. (a) Has gathering the trace perturbed the system? For example, the overhead of a tracer may upset the time sequence of events. (b) Is the workload traced representative of the target workload? (c) Does the workload change rapidly over time, so that old trace data is obsolete? (d) Is the system being simulated different enough from the system traced that the nature of the trace (e.g. the sequence and timing of events) becomes invalid?

All of the issues mentioned immediately above are valid ones and must be considered by the researcher or analyst. Note that in almost all cases, however, some data is better than none, and a larger sample better than a small one.

7. Conclusions

The factor that most determines whether a research study of a computer or system design is useful is the accuracy of the workload information used to drive, or incorporated in, the system model. Highly sophisticated models incorporating incorrect assumptions or wrong parameter values are of little use.

In this paper, we have discussed various types of system optimization studies, concentrating on topics on which the author and his students have worked. In each case, we have discussed the type of data or workload information needed for the study results to be useful. For many of the types of studies described, trace driven simulation is the preferred technique. We briefly considered how traces can be obtained and what some of the limits on their utility are.

8. Bibliography

[Arch86] Archibald, James and Jean-Loup Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model", ACM TOCS, 4, 4, November, 1986, pp. 273-298.

[Bask76] Forest Baskett and Alan Jay Smith, "Interference in Multiprocessor Computer Systems with Interleaved Memory", Communications of the ACM, 19, 6, June, 1976, pp. 327-334.

[Ferr78] Domenico Ferrari, "Computer Systems Performance Evaluation", Prentice Hall, Englewood Cliffs, NJ., 1978.

[Gee93] Jeffrey Gee, "Analysis of Caches in Vector Processors and Multiprocessors", Ph.D. Dissertation, Computer Science Division, UC Berkeley, Berkeley, CA., April, 1993.

[Hill89] Mark Hill and Alan Jay Smith, "Evaluating Associativity in CPU Caches", IEEETC, December, 1989, 38, 12, pp. 1612-1630.

[Kure88] Oivind Kure, "Optimization of File Migration in Distributed Systems", UC Berkeley Computer Science Division Technical Report UCB/CSD 88/413, April, 1988.

[Lee84] John K-F Lee and Alan Jay Smith, "Analysis of Branch Prediction Strategies and Branch Target Buffer Design", IEEE Computer, 17, 1, January, 1984, pp. 6-22.

[Lync72] W.C. Lynch, "Do Disk Arms Move", Performance Evaluation Review, 1, 4, December, 1972, pp. 3-16.

[MacD70] M. H. MacDougal, "Computer System Simulation: An Introduction", Computing Surveys, 2, 3, September, 1970, pp. 191-209.

[MacD84] Myron H. MacDougall, "Instruction Level Program and Processor Modeling", IEEE Computer, July, 1984, pp. 14-24.

[Peut77] Bernard L. Peuto and Leonard J. Shustek, "An Instruction Timing Model of CPU Performance", Proc. 4'th Ann. Symp. on Computer Arch., March, 1977, pp. 165-178.

[Saav89] Rafael Saavedra-Barrera, Alan Jay Smith and Eugene Miya, "Machine Characterization Based on an Abstract High Level Language Machine", IEEE Transactions on Computers, special issue on Performance Evaluation, December, 1989, 38, 12, pp. 1659-1679.

[Saav92a] Rafael Saavedra-Barrera and Alan Jay Smith, "Performance Prediction by Benchmark and Machine Analysis", Computer Science Technical Report UCB/CSD 90/607, December, 1990.

[Saav92b] Rafael Saavedra-Barrera and Alan Jay Smith, "Performance Characterization of Optimizing Compilers", UC Berkeley Computer Science Technical Report UCB/CSD 92-699, August, 1992.

[Saav92c] Rafael Saavedra-Barrera, Ph.D. Dissertation, February, 1992, "CPU Performance Evaluation and Execution Time Prediction Using Narrow Spectrum Benchmarking", UC Berkeley Computer Science Report UCB/CSD 92/684.

[Sher72] Stephen Sherman, Forest Baskett III and J. C. Browne, "Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System", CACM, December, 1972, 15, 12, pp. 1063-1069.

[Sing94] Vigyan Singhal and Alan Jay Smith, "Characterization of Contention in Real Relational Databases" Technical Report UCB/CSD-94-801, Computer Science Division, UC Berkeley, March, 1994. Submitted for publication.

[Smit77] Alan Jay Smith, "Two Methods for the Efficient Analysis of Memory Address Trace Data", IEEE Transactions on Software Engineering, SE-3, 1, January, 1977, pp. 94-101.

[Smit79] Alan Jay Smith, "An Analytic and Experimental Study of Multiple Channel Controllers", IEEE Transactions on Computers, C-28, 1, January, 1979, pp. 38-49.

[Smit81a] Alan Jay Smith, "Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms", IEEE Transactions on Software Engineering, SE-7, 4, July, 1981, pp. 403-417.

[Smit81b] Alan Jay Smith, "Long Term File Migration: Development and Evaluation of Algorithms", Communications of the ACM, 24, 8, August, 1981, pp. 521-532.

[Smit81c] Alan Jay Smith, "Bibliography on File System and Input/Output Optimization and Related Topics", Operating Systems Review, 15, 4, October, 1981, pp. 39-54.

[Smit81d] Alan Jay Smith, "Input/Output Optimization and Disk Architecture: A Survey", Performance Evaluation, 1, 2, 1981, pp. 104-117.

[Smit82] Alan Jay Smith, "Cache Memories", Computing Surveys, 14, 3, September, 1982, pp. 473-530.

[Smit85a] Alan Jay Smith, "Disk Cache - Miss Ratio Analysis and Design Considerations", ACM Transactions on Computer Systems, 3, 3, August, 1985, pp. 161-203.

[Smit85b] Alan Jay Smith "Cache Evaluation and the Impact of Workload Choice", March, 1985, Proc. 12'th International Symposium on Computer Architecture, June 17-19, 1985, Boston, Mass, pp. 64-75.

[Smit87] Alan Jay Smith, "Line (Block) Size Selection in CPU Cache Memories", IEEE Transactions on Computers, C-36, 9, September, 1987, pp. 1063-1075.

[Thom87] James Thompson, "Efficient Analysis of Caching Systems", UC Berkeley Computer Science Division Technical Report 87/374, October, 1987.

[Zhou85] Songnian Zhou, Herve DaCosta, and Alan Jay Smith, "A File System Tracing Package for Berkeley Unix", Proc. 1985 USENIX Summer Conference, Portland, Oregon, June 12-14, 1985, pp. 407-419.

9. Biography

Alan Jay Smith was raised in New Rochelle, New York, USA. He received the B.S. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, Massachusetts, and the M.S. and Ph.D. degrees in computer science from Stanford University, Stanford, California. He was an NSF Graduate Fellow.

He is currently a Professor in the Computer Science Division of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California, USA; he was vice chairman of the EECS department from July, 1982 to June, 1984. His research interests include the analysis and modeling of computer systems and devices, computer architecture, and operating systems. He has published a large number of research papers, including one which won the IEEE Best Paper Award for the best paper in the IEEETC in 1979. He also consults widely with computer and electronics companies.

Dr. Smith is a Fellow of the Institute of Electrical and Electronic Engineers and a Fellow of the Association for Computing Machinery, a member of IFIP Working Group 7.3, the Computer Measurement Group, Eta Kappa Nu, Tau Beta Pi and Sigma Xi. He is on the Board of Directors (1993-2001), and was Chairman (1991-93) of the ACM Special Interest Group on Computer Architecture (SIGARCH), was Chairman (1983-87) of the ACM Special Interest Group on Operating Systems (SIGOPS), was on the Board of Directors (1985-89) of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS), was an ACM National Lecturer (1985-6) and an IEEE Distinguished Visitor (1986-7), was an Associate Editor of the ACM Transactions on Computer Systems (TOCS) (1982-93), is a subject area editor of the Journal of Parallel and Distributed Computing and is on the editorial board of the Journal of Microprocessors and Microsystems. He was program chairman for the Sigmetrics '89 / Performance '89 Conference, program co-chair for the Second (1990) Sixth (1994) and Ninth (1997) Hot Chips Conferences, and has served on numerous program committees.