

Using Set Sampling in Level Three Cache Studies

A Thesis Proposal
Presented to the
Department of Computer Science
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Niki C. Thornock
August 1999

1 Introduction

Processor speeds are increasing at a much greater rate than memory speeds. The resulting difference causes a bottleneck in the system and decreases performance. With the increasing popularity of multiprocessor computers, the bottleneck is becoming even worse. Researchers are searching for ways to reduce this problem. Since it is now common to have two caches (level 1 and level 2) integrated onto each processor chip, adding a third, very large, off-chip cache (level 3 or L3) seems a likely candidate for reducing the bottleneck.

Simulation, especially trace-driven simulation, is a frequently used method of testing new cache configurations. A cache configuration generally consists of cache size, line size (the amount of data stored for each address requested) and associativity (the number of lines in one row of the cache). Creating a simulator is a fairly straightforward, albeit very time consuming, task. The difficulty lies in obtaining the long, accurate traces necessary for simulating extremely large L3 cache systems used in current and future multiprocessor systems.

1.1 Background

For our purposes, a trace is a stream of successive address requests containing references for memory reads, writes, and instruction fetches. Here are five different trace collection methods; instruction modification [1, 2, 3], microcode modification[4], single stepping [4], processor simulation [5], and hardware monitors [6, 7, 8]. These methods introduce four types of errors into collected trace data, (1) missing operating system references, (2) absent multitasking behavior, (3) time dilation, (4) short traces [9]. These introduced errors mean that simulation runs are not completely accurate. A hardware monitor which overcomes many of these problems is described in [10, 11], but this technique requires frequently halting the system under test (SUT). Thus, although this type of hardware monitor eliminates the four problems mentioned above, it can introduce new errors due to halting the SUT.

One major disadvantage associated with trace-driven simulation is the storage requirements of long traces. Another disadvantage is the long run times of simulations. It takes several days to simulate a long trace and several gigabytes

to store it. Sampling is a technique that partially overcomes these disadvantages. This is a statistical method where a selected fraction of a population is used to represent the whole population. This method has been demonstrated to work effectively with first- and second-level caches in single processor systems [12, 13]. Sampling has three potential advantages. It reduces disk space needed to store traces, enables simulations to run faster, and effectively enlarges trace buffers of hardware monitors by storing only a fraction of the data. We will investigate whether the sampling techniques described in [12] (time sampling and two types of set sampling) perform acceptably when simulating L3 caches in multiprocessor systems.

2 Thesis Statement

Sampling is a technique which gives results similar to that of a complete trace but reduces the disk space needed to store traces, enables simulations to run faster, and effectively enlarges the trace buffers of hardware monitors.

3 Methods

1. Develop a tracing mechanism for a symmetric multiprocessor (SMP).
2. Collect long traces of different workloads from an SMP system.
3. Run the traces through a cache simulator to find the real miss rate.
4. Run the traces through a cache simulator after using the different sampling techniques and find the miss rate.
5. Compare the results of the real and sample miss rates and see which sampling technique is the most accurate using the smallest fraction of a trace. As a baseline, we determined that the sample miss rate should be within 10% of the real miss rate using at most 10% of the trace.
6. Give confidence intervals for different sampling techniques.

4 Contribution to Computer Science

This thesis will investigate a sampling technique which can be integrated with a tracing mechanism in order to achieve more effective collection of address requests for use with cache simulations. It also provides a tracing mechanism for SMP systems.

5 Delimitations of the Thesis

We will not investigate sample sizes other than 6%, 10%, and 25%. We will not investigate cache levels other than level three. We will not investigate the effectiveness of sampling techniques for methods other than cache simulation. We will not simulate cache sizes greater than 32 megabytes or less than 8 megabytes or line sizes greater than 64 bytes or less than 32 bytes. We will not simulate associativities greater than 8. We will not consider non-cacheable references.

6 Thesis Outline

1. Introduction (2 - 3 pages)
 - (a) Background
 - (b) Describe the rest of the paper
 - (c) Summary of contributions
2. Cache Memory and Trace-Driven Simulation (4 - 6 pages)
3. Trace Collection Techniques (3 - 5 pages)
4. Multiprocessor Tracing (5 - 10 pages)
 - (a) Trace Collection Technique
 - (b) Verification and Evaluation
5. Sampling (8 - 10 pages)
 - (a) Sampling Techniques
 - (b) Workloads

(c) Results

6. Conclusions (2 - 4 pages)

7 Thesis Schedule

I will attempt to follow this schedule while working on my thesis.

- Develop an SMP tracing mechanism (6 weeks)
- Collect multiprocessor traces. (6 weeks)
- Run original simulations (4 weeks)
- Run sampled simulations (4 weeks)
- Evaluate the sampling techniques (2 weeks)
- Write thesis. This will happen concurrently with the rest of the project.

8 Bibliography

References

- [1] A. Borg, R. E. Kessler, and D. W. Wall, Generation and analysis of very long address traces, In *Proc. of 17th Int. Symp. on Computer Architecture*, pages 270–279. ACM, 1990. The authors introduce a technique which collects very long address traces. They then analyze the performance of first- and second-level caches.
- [2] Sun Microsystems, Introduction to Shade / Shade User’s Manual, Technical report, Sun Microsystems Laboratories, Inc., April 1992. Shade is a tool that uses inlining on the executable. The introduction explains what Shade is and how it works. The User’s Manual is a hard copy of the man pages.
- [3] J. R. Larus and T. Ball, Rewriting executable files to measure program behavior, Technical Report 1083, Dept. of Computer Science, University of Wisconsin, March 1992. The paper analyzes the qp and qpt profiling tools, and discuss how the tools could be modified to improve performance.

- [4] A. Agarwal, R. L. Sites, and M. Horowitz, ATUM: A new technique for capturing address traces using microcode, In *Proc. of 13th Int. Symp. on Computer Architecture*, pages 119–127. IEEE, 1986. The authors describe a new technique for collecting address traces by instrumenting the microcode to save addresses in a reserved section of memory.
- [5] Gurindar S. Sohi and Manoj Franklin, High-bandwidth data memory systems for superscalar processors, In *Proc. of 4th Int. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, pages 53–62. ACM, 1991. The authors discuss a cache that is capable of supporting the bandwidth of a superscalar processor capable of issuing 10 instructions per cycle.
- [6] Douglas W. Clark, Cache performance in the VAX-11/780, *ACM Transactions on Computer Systems*, **1**(1):24–37, February 1983. Clark uses a hardware monitor rather than trace-driven simulation to measure cache performance.
- [7] D. Nagle, R. Uhlig, T. Stanley, S. Sechrest, T. Mudge, and R. Brown, Design tradeoffs for software-managed TLBs, In *Proc. of 20th Int. Symp. on Computer Architecture*, pages 27–38. ACM, 1993. The authors found that newer operating systems are changing the types of TLB misses. The TLB miss rate varied widely for the same application under different operating systems.
- [8] Josep Torrellas, Anoop Gupta, and John Hennessy, Characterizing the caching and synchronization performance of a multiprocessor operating system, In *Proc. of 5th Int. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, pages 162–174. ACM, 1992. The authors analyze the causes and effects of operating system misses on CPU performance. They found that instruction fetches, process migration, and data accesses were the major causes of operating system cache misses.
- [9] D. T. Harper III and Z. Zhang, Cache miss ratio analysis using sampled traces, Revision in progress. Submitted to ACM TOMACS., 1993. The authors investigate whether a sampled trace accurately represents the execution. They conclude that it does.

- [10] J. Kelly Flanagan, *A New Methodology for Accurate Trace Collection and its Application to Memory Hierarchy Performance Modeling*, PhD thesis, Brigham Young University, December 1993. Flanagan developed a tool and methodology for collecting accurate and long trace data (BACH). He collected data from various machines and operating systems and used it in a memory hierarchy study.
- [11] Niki Crockett and J. Kelly Flanagan, i486 address trace collection using a TLA510, Technical report, Department of Computer Science, Brigham Young University, Provo, UT 84602, 1994. This document describes a trace collection setup using the Tektronix TLA510 logic analyzer. The setup collects very long traces.
- [12] R. E. Kessler, Mark D. Hill, and David A. Wood, A comparison of trace-sampling techniques for multi-megabyte caches, *IEEE Transactions on Computers*, pages 664–675, June 1994. The authors investigate time and set sampling methods for large caches in a single processor machine. They conclude that set sampling is accurate, but time sampling is not.
- [13] Margaret Martonosi, Anoop Gupta, and Thomas E. Anderson, Tuning memory performance of sequential and parallel programs, *Computer*, pages 32–40, April 1995. The authors discuss MemSpy, a performance monitoring tool which uses cache simulations to gather detailed memory statistics. They use time sampling to optimize their work.

9 Artifacts

This thesis will result in the following artifacts:

- A multiprocessor trace collection mechanism
- An accurate sampling method
- Long traces for various workloads
- Cache statistics for various workloads

This thesis proposal by Niki C. Thornock is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the thesis proposal requirement for the degree of Master of Science.

J. Kelly Flanagan, Committee Chair

Michael A. Goodrich, Committee Member

Douglas M. Campbell, Committee Member

Scott N. Woodfield, Graduate Coordinator