

Transaction Processing Workloads - A Comparison to the SPEC Benchmarks Using Memory Hierarchy Performance Studies

Gregory D. Thompson
Intel Corp.
Santa Clara, CA 95035

Brent E. Nelson
Brigham Young Univ.
EE Dept.
Provo, UT 84602

J. Kelly Flanagan
Brigham Young Univ.
CS Dept.
Provo, UT 84602

Abstract

This study analyzes the memory hierarchy performance of three SPEC benchmarks and two TPC benchmarks. It finds large differences between the benchmarks in instruction cache miss rates and smaller differences in data cache miss rates. It then breaks all of the miss rates down in their components: context switch misses, user misses, supervisor misses, and collision misses. It demonstrates that context switches contribute little to the miss rates as do collision misses. Finally, using temporal locality graphs, it shows that the inherent locality differences between the reference streams is the main cause of miss rate differences between the various benchmarks.

1 Introduction

1.1 Problem to Solve

In the computer industry, many benchmarks and measurement techniques are used to quantify systems performance with new benchmarks continually being produced. This study quantifies the differences in the memory hierarchy performance of computer systems executing five benchmarks: two from SPECint (li and GCC), one from SPECsdm (KENBUS), and two from TPC (TPC-B and TPC-C). It does this using trace-driven simulation with traces collected from a hardware monitor.

This is believed to be the first study that analyzes the memory hierarchy performance for transaction processing benchmarks in this manner. Gee et al [1] studied the single-user SPEC benchmarks. Flanagan [2] analyzed the cache performance of the SPEC multi-user benchmarks (KENBUS and SDET) was similar to this work, but without transaction processing workloads. McGrory et al [3] analyzed performance of a transaction processing benchmark and gave cache miss rates for only a single cache size and configuration. Cvetanovic and Bhandarkar [4] analyzed cache performance for SPEC single-user benchmarks and the TP1 transaction processing benchmark. They found that TP1 had more system references and higher miss rates than the single-user workloads but did not use the SPEC multi-user benchmarks. Maynard et al [5] studied the cache performance of SPEC multi-user benchmarks, two transaction processing benchmarks, and others but didn't quantify the context switch contribution to the miss rates nor give the user, supervi-

sor, and collision misses. Additionally, none of the above studies have quantified context switch contributions nor analyzed the benchmarks' locality.

1.2 Method and Outline of Paper

Throughout the rest of this paper, data and instruction reference streams are analyzed separately. This corresponds to current split cache architectures. We use caches that range in size from 4 KB to 256 KB with 1- and 2-way associativities and 16 byte lines. I/O is not cached. The caches use LRU replacement, allocate on write, and writeback. In this paper, miss rate is the number of cache misses divided by the number of memory references.

First, we describe the workloads used, the methods used to capture the address traces, and the resulting traces. We then analyze the instruction and data references separately, dividing the miss rates into the following categories: context switch misses, collision misses, supervisor misses, and user misses. Since dividing up the miss rates into these four components cannot fully explain the difference in miss rates, the temporal locality of the reference streams is then analyzed.

2 Workloads Used

The benchmarks used in this work include some from SPEC and TPC. All the workloads were run on an i486 machine at 20MHz with 16 MB of main memory running UNIX SysVR4.2. The TPC benchmarks were written in SQL embedded in C and run using the Informix On-line Database Engine 5.01.

LI and GCC from the SPECint92[6] were used for this study. The SPECsdm program KENBUS[7] was also used with a configuration that gave 120 scripts per hour (maximum throughput on the target i486).

The TPC programs TPC-B and TPC-C[8, 9] were also used. This TPC-B implementation models nine bank branches with three user-level (teller) processes and requires 32.6 MB of disk storage. This implementation of TPC-C models two distribution warehouses with four user-level processes. It does not model keyboard rate or terminal I/O. It takes 185 MB on disk.

As in [2], the address traces used in this study were collected using BACH (BYU Address Collection Hardware) [10, 11], developed at the Department of Electrical and Computer Engineering at Brigham Young University. BACH is a hardware monitor that collects

long, contiguous, and accurate address traces of every memory reference that the processor generates. Each record contains the physical address, data, and type of reference.

The trace characteristics are given in Table 1.

Name	Refs (*10 ⁶)	OS (%)	Rd (%)	Wr (%)	IF (%)	I/O (%)
LI	473	4	17	10	73	0.001
GCC	403	7	18	9	74	0.1
KENBUS	508	41	18	9	72	0.7
TPC-B	422	40	19	10	70	1.2
TPC-C	422	50	20	10	69	0.9

Table 1: Characteristics Of The Traces Used Including Number of References and Percentages of Operating System References (OS), Data Reads (Rd), Data Writes (Wr), Instruction Fetches (IF), and I/O

The traces are all of similar length, each accounting for between 66 and 88 seconds of real execution time. As expected, the single-user benchmarks have little operating system activity or I/O; the TPC benchmarks have the most I/O.

Since the data/instruction mix is similar across the benchmarks, treating data and instruction references separately in this work should not introduce anomalies due to widely varying numbers of references. However, the differences in the percentage of system references could lead to significant differences in miss rates; this will be explored in detail later in the paper.

3 First-Level Cache Miss Rates

The instruction cache (ICACHE) and data cache (DCACHE) miss rates are presented for direct-mapped caches in Tables 2 and 3 respectively. The ICACHE results show much higher miss rates for the transaction processing workloads compared to the SPEC workloads. In contrast, KENBUS has equal or higher DCACHE miss rates. This can be explained in that KENBUS consists of many small UNIX programs, each with private data. In contrast, the TPC benchmarks consist of a central database engine serving a few query processes. The majority of the data is shared via access through the engine and the number of active processes in the TPC benchmarks is much lower than in KENBUS.

Another way to view the data is to determine the cache size needed to achieve a given miss rate for each benchmark. Table 4 gives the direct-mapped cache size needed for a 3% miss rate. The major points of interest from this table are (i) the discrepancy for KENBUS between the required ICACHE and DCACHE sizes and (ii) the relatively modest DCACHE requirements for the TPC benchmarks.

4 Context Switch Behavior

This section examines the effect that context switches have on miss rates, demonstrating that for small caches, context switches have little impact on

Size	Workloads				
	LI	GCC	KEN	TPC-B	TPC-C
4	2.516%	6.23%	7.29%	11.18%	11.16%
8	0.585%	4.19%	5.80%	9.63%	9.68%
16	0.326%	2.70%	4.07%	7.86%	7.75%
32	0.153%	1.64%	2.85%	6.08%	6.07%
64	0.060%	1.07%	1.84%	4.42%	4.08%
128	0.010%	0.57%	1.14%	3.02%	2.60%
256	0.004%	0.25%	0.66%	1.80%	1.39%

Table 2: Direct-Mapped ICACHE Miss Rates

Size	Workloads				
	LI	GCC	KEN	TPC-B	TPC-C
4	5.41%	9.66%	13.17%	11.57%	13.06%
8	4.16%	7.32%	9.72%	8.90%	10.25%
16	2.75%	5.21%	7.36%	6.76%	7.79%
32	1.17%	3.17%	5.72%	4.84%	5.64%
64	0.72%	2.28%	4.50%	3.34%	3.87%
128	0.50%	1.38%	3.64%	2.31%	2.68%
256	0.28%	0.96%	3.03%	1.51%	1.86%

Table 3: Direct-Mapped DCACHE Miss Rates

the miss rate but for large caches, they constitute 20-30% of the misses. For this study, a *context switch* is defined as the switch from one user process to the supervisor and then to a different user process. *Process transitions* are switches from a user process to the supervisor and then back to the same user process. They result from interrupts, system calls, etc. The numbers of context switches and process transitions are given in Table 5.

At first glance, one might wonder why the single-user benchmarks have any context switches at all — during their execution *supervisor* processes execute in user space as part of normal daemon activity (accounting, however, for less than 1% of all references for the single-user benchmarks). Additionally, GCC consists of a few processes that run in sequence. Thus its trace

Workload	Instruction cache size needed for 3% miss rate	Data cache size needed for 3% miss rate
LI	4 KB	16 KB
GCC	16 KB	32 KB
KENBUS	32 KB	256 KB
TPC-B	128 KB	64 KB
TPC-C	128 KB	128 KB

Table 4: Cache Sizes Needed for a 3% Miss Rate (direct-mapped caches)

Workload	Context Switches	Process Trans
LI	13	11,119
GCC	28	10,935
KENBUS	2,949	25,070
TPC-B	7,399	20,260
TPC-C	12,090	25,676

Table 5: Context Switches and Process Transitions

contains context switches but they come between major program phases.

The large difference in context switches and process transitions between the workloads suggests that the context switching behavior could be a major contributor to the difference in the benchmark miss rates. In order to quantify their contribution to the miss rates, a batch trace was created for each benchmark by processing the original trace to give all of the references from one process first, followed by all of the references from the second process, and so on. The new trace contains only one context switch for each process (as it is started). The context switch contribution to the miss rate is then the batch miss rate subtracted from the overall miss rate.

For smaller caches, context misses account for a very small fraction of the misses (less than 10%). This is because a given process replaces many of its own cache lines before a context switch, and the number of lines replaced by other processes is small.

For larger caches, the context switches still make up less than 25% of the misses and, more importantly, are not responsible for the differences in miss rates between the benchmarks. This is in spite of the fact that there are large differences in the number of context switches between the benchmarks. From this we conclude that the majority of the miss rate differences must be due to other causes.

5 User/Supervisor Cache Collisions

This section analyzes the effects on the miss rates from the interaction of the user and supervisor. To do this, the batch misses from the previous section were divided into three categories: user misses, supervisor misses, and user/supervisor collision misses.

The user misses were found by using only the user references in a cache simulation; the supervisor misses by using only the supervisor references. Collision misses occur when a supervisor reference replaces a user line in the cache or vice versa. The number of collision misses is calculated by subtracting the sum of user and supervisor misses from the total batch misses.

5.1 ICACHE Miss Rate Components

The miss rates for direct-mapped and 2-way associative ICACHES are shown broken into these categories in Figures 1 and 2 for small and large ICACHES, respectively.

The instruction misses for GCC are mostly user misses, with a very small proportion of supervisor and

collision misses. This is simply because of the small amount of supervisor activity in GCC.

The figures show that for small caches, collisions and context switches have little impact on the miss rates. This is because the user replaces many of its own cache lines before the supervisor executes. Likewise for the supervisor code. As the cache size increases, the working set for a process fits into the cache better. However, the working set for the entire user workload plus the supervisor still does not fit well in the cache. For the largest caches and the multi-user benchmarks, the majority of misses are due to context switches and collisions. Thus, changing the user code to improve reference locality may have little effect on overall system performance.

5.2 DCACHE Miss Rate Components

KENBUS’s DCACHE performance is similar to TPC-B and TPC-C as shown in Figures 3 and 4. Again we see that most misses for the multi-user benchmarks are due to system activity and few due to user code. The TPC benchmarks have more collision and context switch misses, especially for large caches, than the others. In addition, for the largest caches, there is a bigger improvement in making a cache 2-way associative compared to doubling its size.

A final point worth noting from the figure is that the user miss rates of GCC are higher than for any of the others. Thus, the use of GCC as a serious benchmark is a good choice—its major shortcoming compared to the other workloads is its lack of supervisor activity and context switching.

6 Locality Analysis

In summary, although there are significant differences in the number of context switches in the benchmarks, the analysis above shows that this does not explain the differences in cache miss rates. The locality of the workloads is therefore analyzed in this section.

Locality is a notion of how addresses are reused. It is the idea that if an address is accessed once, it and addresses nearby will be accessed again soon. It is based on loops, repeated calls, array traversal, etc. It is locality that allows caches to perform so well.

A temporal locality graph is a representation of how many references must occur before the current cache line is accessed again. It is a two-dimensional plot with the horizontal axis being the number of references between two accesses. The vertical axis is the probability, calculated as a percentage, that the current line is reused within that distance. If the locality graphs for the benchmarks vary markedly, that would explain the differences in miss rates.

The temporal locality graphs of the complete instruction streams of the benchmarks are given in Figure 5. As an example of the significant differences illustrated, consider the probability for LI (76%) at a distance of 512 references. To achieve the same probability for the TPC benchmarks requires a distance 32 times as far or 16K references. Similar results were obtained for user-only, supervisor-only, and data-only streams. In fact, these temporal locality graphs were

good predictors of the user-only and supervisor-only miss rates.

In summary, the preceding sections have shown that context switches and system interaction contribute a small amount to the differences in miss rates among the multi-user benchmarks with their contribution being greater for larger caches. This section has shown through locality graphs that there are large differences in the temporal locality of the benchmarks' reference streams and thus that locality differences are the main cause of the differences in miss rates between KENBUS and the TPC benchmarks.

7 Conclusions and Summary

This study has investigated the memory hierarchy performance of five benchmarks:

- LI and GCC from SPECint,
- KENBUS from SPECsdm, and
- TPC-B and TPC-C from TPC.

This is believed to be the first study that analyzes the memory hierarchy performance of the TPC benchmarks in this manner.

This study has demonstrated that the TPC benchmarks have significantly higher instruction cache miss rates than the SPEC benchmarks. In contrast, the data cache miss rates are higher for SPEC's KENBUS than for the TPC benchmarks. By categorizing the cache misses it was shown that although the TPC benchmarks have many more context switches than the other benchmarks, this is not a major contributor to the miss rate. Rather, the inherent locality characteristics of the various workloads was determined to be the main contributor to miss rate differences.

References

- [1] J. D. Gee, M. D. Hill, D. N. Pnevmatikatos, and A. J. Smith, "Cache performance of the SPEC92 benchmark suite," *IEEE Micro*, pp. 17 – 27, Aug. 1993.
- [2] J. K. Flanagan, *A New Methodology for Accurate Trace Collection and its Application to Memory Hierarchy Performance Modeling*. PhD thesis, Brigham Young University, 1993.
- [3] J. J. M. II, A. Carlton, and B. J. Askins, "Transaction processing performance on PA-RISC commercial unix systems," *IEEE Comcon*, pp. 199 – 206, Spring 1992.
- [4] Z. Cvetanovic and D. Bhandarkar, "Characterization of alpha AXP performance using TP and SPEC workloads," in *Proceedings of the 21st International Symposium on Computer Architecture*, pp. 60 – 70, Apr. 1994.
- [5] A. M. G. Maynard, C. M. Donnelly, and B. G. Olszewski, "Contrasting characteristics and cache performance of technical and multi-user commercial workloads," in *Proceedings of the International Conference on Architectural Support for*

Programming Languages and Operating Systems (ASPLOS), pp. 145 – 156, ACM (Association for Computing Machinery), Oct. 1994.

- [6] SPEC (Standard Performance Evaluation Corporation), *SPECint92 Benchmark*, Feb. 1992. official specifications.
- [7] SPEC (Standard Performance Evaluation Corporation), *SPEC SDM Release 1.1*, Mar. 1992. official specifications.
- [8] TPC (Transaction Processing Performance Council), *TPC Benchmark B Standard Specification Revision 1.1*, Mar. 1992.
- [9] TPC (Transaction Processing Performance Council), *TPC Benchmark C Standard Specification Revision 1.1*, June 1993.
- [10] K. Grimsrud, J. Archibald, M. Ripley, K. Flanagan, and B. Nelson, "BACH: A hardware monitor for tracing microprocessor-based systems," *Microprocessors and Microsystems*, vol. 17, pp. 443–459, Oct. 1993.
- [11] J. K. Flanagan, B. E. Nelson, J. K. Archibald, and K. Grimsrud, "BACH: BYU Address Collection Hardware, the collection of complete traces," in *Proc. of the 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pp. 128–137, Sept. 1992.

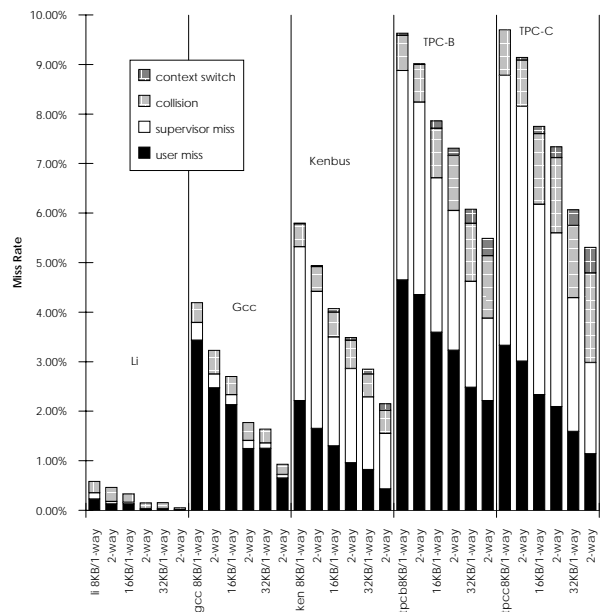


Figure 1: ICACHE Miss Rates For Small Caches

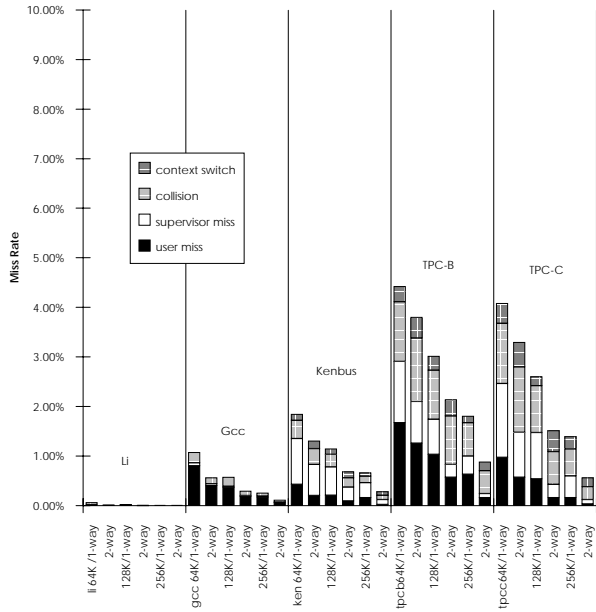


Figure 2: ICACHE Miss Rates For Large Caches

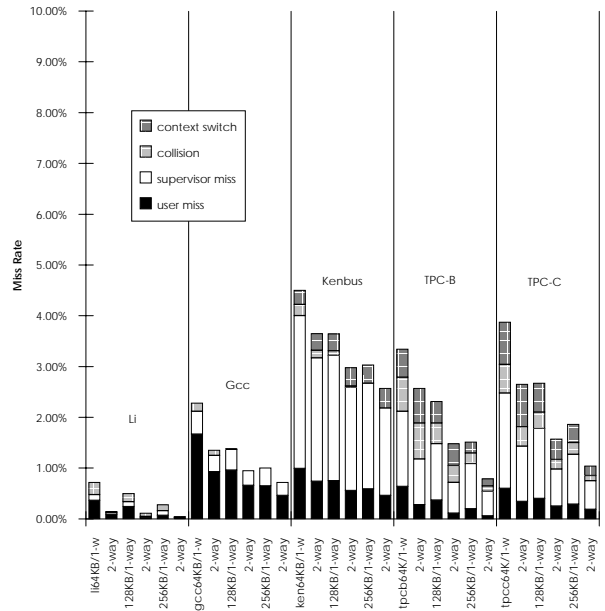


Figure 4: DCACHE Miss Rates For Large Caches

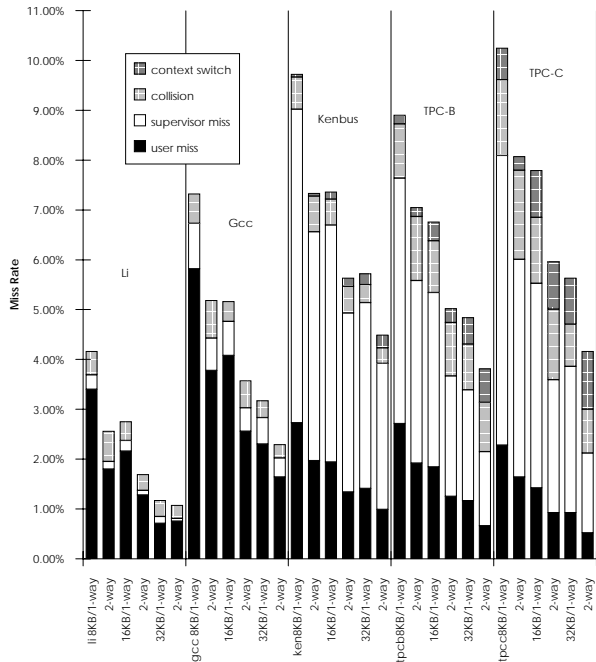


Figure 3: DCACHE Miss Rates For Small Caches

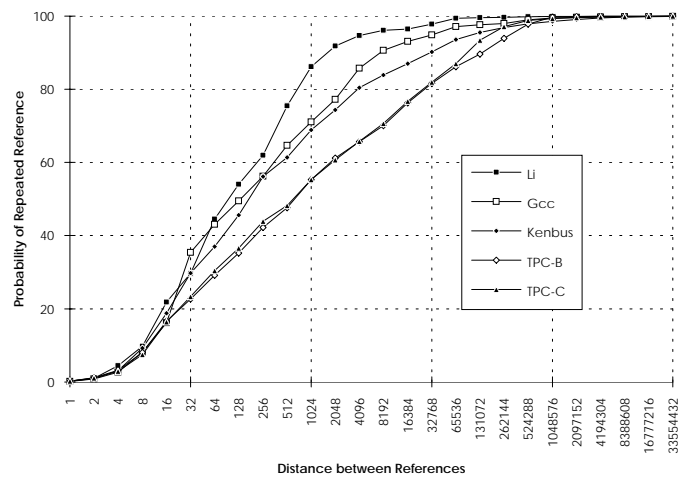


Figure 5: Temporal Locality Of The Instruction Streams