Cache Performance for Selected SPEC CPU2000 Benchmarks

Version 1.0

July 2001

Jason F. Cantin
Department of Electrical and Computer Engineering
1415 Engineering Drive
University of Wisconsin-Madison
Madison, WI 53706-1691
jcantin@ece.wisc.edu
http://www.jfred.org

Mark D. Hill
Department of Computer Science
1210 West Dayton Street
University of Wisconsin-Madison
Madison, WI 53706-1685
markhill@cs.wisc.edu
http://www.cs.wisc.edu/~markhill

http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data

Abstract

The SPEC CPU2000 benchmark suite (http://www.spec.org/osg/cpu2000) is a collection of 26 compute-intensive, non-trivial programs used to evaluate the performance of a computer's CPU, memory system, and compilers. The benchmarks in this suite were chosen to represent real-world applications, and thus exhibit a wide range of runtime behaviors. On this webpage, we present functional cache miss ratios and related statistics for selected benchmarks in the SPEC CPU2000 suite. In particular, split L1 cache sizes ranging from 4KB to 1MB with 64B blocks and associativities of 1, 2, 4, 8 and full. Most of this data was collected at the University of Wisconsin-Madison with the aid of the Simplescalar toolset (http://www.simplescalar.org).

Methodology

All functional data was collected with simulators from the Alpha version of the Simplescalar toolset, version 3.0. These include sim-cache, sim-cheetah, and sim-outorder. Some of these simulators were modified for the task (for example, sim-cheetah was modified to handle programs longer than 2 billion instructions). For interval cache data, the simulators were modified to print stats every 100 million executed instructions. A combination of Perl and Tcsh scripts were used to launch, manage, and process the results of these simulations.

All benchmarks were compiled statically with heavy optimization for the Alpha AXP instruction set.

Optimizations were targeted at the Alpha 21264 processor implementations, and include prefetches, square-root instructions, byte/word memory operations, and no-ops for alignment. All benchmarks were run to completion with all reference inputs, with two exceptions. Two of the data sets for Perl (253.perlbmk) required new processes to be spawned, which is not supported by the Simplescalar tools at this time. The benchmarks simulated comprise nearly 2 trillion committed instructions for the reference input sets. Generating the functional L1 miss-ratio tables resulted in *100 trillion* simulated instructions. The total simulation load for all functional and timing-based simulations to be reported here totals *30 CPU-years.*

All simulations were carried out on a combination of x86 Linux machines and Alpha (Tru64 Unix) servers in the University of Wisconsin-Madison's Computer Science Department. The majority of this load was managed by Condor, which distributed these jobs to vacant machines throughout the building. The Alpha servers are not managed by Condor at this time, and those simulations were managed manually.

All cache configurations simulated had 64-byte blocks for both L1 and L2 caches. All data reported here is for the LRU replacement policy, though data for other replacement policies was collected and may be placed on this site soon. This data does not include operating system effects, and caches were not flushed periodically nor on system calls. Thus, actual miss-rates will be higher than those reported here.

Benchmarks Simulated

To date, we have collected data for 11 benchmarks with reference inputs: 6 integer, 5 floating-point.

Benchmark	Language	Type	Category	Simulated Inst's	Simulated Data Ref's	Ref's/Inst
164.gzip	C	Int	Compression	478,636,174,329	142,700,878,428	0.2981
176.gcc	C	Int	C Programming Language Compiler	116,093,336,744	243,597,914,726	0.4766
181.mcf	C	Int	Combinatorial Optimization	61,870,158,860	23,056,352,854	0.3727
253.perlbmk	C	Int	PERL Programming Language	143,122,956,639	61,829,661,201	0.4320
254.gap	C	Int	Group Theory, Interpreter	213,813,801,949	80,924,423,445	0.3785
300.twolf	C	Int	Place and Route Simulator (CAE)	346,489,363,383	111,857,479,345	0.3228
171.swim	Fortran 77	FP	Shallow Water Modeling	225,830,970,951	74,341,437,755	0.3292
173.applu	Fortran 77	FP	Parabolic/Elliptic Partial Diff. Eq'ns	223,883,653,813	85,459,068,028	0.3817
179.art	C	FP	Image Recognition / Neural Networks	86,834,976,688	30,279,186,530	0.3487
183.equake	C	FP	Seismic Wave Propagation Simulation	131,518,705,120	58,248,603,550	0.4429
189.lucas	Fortran 90	FP	Number Theory / Primality Testing	142,398,814,292	31,507,111,538	0.2213
Overall				2,170,492,912,770	943,802,117,400	0.4348

Miss Ratio Tables

- Get the complete archive at http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/miss-tables.tar.gz
- Overall First-level Cache Miss ratio tables:
  - Arithmetic mean of all selected benchmarks
  - Arithmetic mean of integer benchmarks (selected SPECint)
  - Arithmetic mean of floating-point benchmarks (selected SPECfp)
- First-level Cache Miss ratio tables for selected benchmarks:
  - 164.gzip
    - 164.gzip1.tab (Source code tar-file)
    - 164.gzip2.tab (Webserver log)
    - 164.gzip3.tab (Large TIFF image)
    - 164.gzip4.tab (Random data)
    - 164.gzip5.tab (Program binary)

        - 164.gzip-ave.tab (Average of above)
    - 176.gcc
        - 176.gcc1.tab (Preprocessed source from a SPECint2000 candidate)
        - 176.gcc2.tab (Preprocessed source from SPECfp2000 200.sixtrack)
        - 176.gcc3.tab (Preprocessed expr.i from gcc source)
        - 176.gcc4.tab (Preprocessed integrate.i from gcc source)
        - 176.gcc5.tab (Preprocessed version of Scilab program)
        - 176.gcc-ave.tab (Average of above)
    - 181.mcf
        - 181.mcf1.tab (Single-depot vehicle scheduling in public mass transportation)
    - 253.perlbmk
        - 253.perlbmk1.tab (Specdiff applied to email)
        - 253.perlbmk2.tab (Finding perfect numbers --not reported, crashes Simplescalar)
        - 253.perlbmk3.tab (Testing pseudo random numbers --not reported, crashes Simplescalar)
        - 253.perlbmk4.tab (Converting Email to HTML)
        - 253.perlbmk5.tab (Converting Email to HTML)
        - 253.perlbmk6.tab (Converting Email to HTML)
        - 253.perlbmk7.tab (Converting Email to HTML)
        - 253.perlbmk-ave.tab (Average of above)
    - 254.gap
        - 254.gap1.tab (Comb. functions, big numbers, finite fields, lattice computations, normalizers, ag-groups)
    - 300.twolf
        - 300.twolf1.tab (structured circuit from MCNC benchmark suite)
    - 171.swim
        - 171.swim1.tab (Large 1335x1335 array over 512 timesteps)
    - 173.applu
        - 173.applu1.tab (Large mesh over many timesteps)
    - 179.art
        - 179.art1.tab (Finding helicopter and airplane in a thermal image)
        - 179.art1.tab (Finding helicopter and airplane in a thermal image)
        - 179.art-ave.tab (Average of above)
    - 183.equake
        - 183.equake1.tab (1994 Northridge Earthquake aftershock in California)
    - 189.lucas
        - 189.lucas1.tab (Lucas-Lehmer test for primality of Mersenne numbers 2^p-1)

- Overall Second-level Cache Miss ratio tables:

- Miss ratios for intervals of 100 million instructions:
    - Tabular data for split 64K L1 caches (http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/splitl1cache64K_tables.tar.gz)
    - Tabular data for a unified 1MB L2 cache (http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/ul2cache1MB_tables.tar.gz)
    - Graphs for 64K L1 data cache (http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/dcache64K_graphs.tar.gz)
    - Graphs for 64K L1 instruction cache

(http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/icache64K_graphs.tar.gz)

- Graphs for 64K L2 unified cache
  (http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/ul2cache1MB_graphs.tar.gz)

Table Format

All miss-ratio tables are in ASCII text format, generated with Perl scripts. They include the name of the file, the name of the benchmark, the command line for the benchmark, the number of instructions, the number of data references, miss-ratios (misses/reference) for a set of cache sizes and associativities, and compulsory miss rates. For each benchmark and data set, miss ratios are rounded to 6 decimal places. The computed arithmetic means for each benchmark are rounded to 5 significant digits, and the overall means are rounded to 4 significant digits. Miss ratios are reported for sizes of 4KB - 1MB, with associativities of 1-way, 2-way, 4-way, 8-way, and full. In all cases the block size was 64 Bytes and the replacement policy was LRU. Compulsory miss-rates were measured as the miss-rate of a fully-associative 256MB cache with no flushing. Note that there is sufficient data to calculate the 3C's for the various configurations. See the example below (overall arithmetic mean for selected benchmarks)

Block size: 64 bytes, Repl: LRU

Arithmetic Mean for Data References

Size	Associativity				
	1	2	4	8	full
4K	0.0599--	0.0511--	0.0491--	0.0488--	0.0487--
8K	0.0514--	0.0455--	0.0439--	0.0434--	0.0429--
16K	0.0455--	0.0419--	0.0410--	0.0408--	0.0403--
32K	0.0412--	0.0387--	0.0383--	0.0380--	0.0378--
64K	0.0377--	0.0359--	0.0356--	0.0355--	0.0353--
128K	0.0344--	0.0335--	0.0333--	0.0333--	0.0333--
256K	0.0326--	0.0322--	0.0323--	0.0323--	0.0323--
512K	0.0268--	0.0290--	0.0309--	0.0312--	0.0311--
1024K	0.0164--	0.0169--	0.0208--	0.0225--	0.0229--

Compulsory: 2.357738e-05

For example, for a 4KB direct-mapped L1 data cache with 64B blocks, 5.99 out of every 100 data references miss.

Future Work

- The rest of the SPEC CPU2000 Suite
- Data for TLBs
- Impact of L2 cache sizes on IPC

Related Work

- SPEC CPU2000: Measuring CPU Performance in the New Millennium, John L. Henning. http://www.spec.org/osg/cpu2000/papers/COMPUTER_200007-abstract.JLH.html
- Simulating SPEC CPU2000: Reduced input sets for SPEC CPU2000, research by the University of Minnesota. http://www.spec.org/osg/cpu2000/research/umn/
- Prefetching and memory system behavior of the SPEC95 benchmark suite, M.J. Charney and T.R. Puzak. http://www.research.ibm.com/journal/rd/413/charney.html
- Cache Performance of the SPEC92 Benchmark Suite, Jeffrey D. Gee, Mark D. Hill, Dionisios N. Pnevmatikatos, and Alan Jay Smith. http://www.cs.wisc.edu/~markhill/spec92miss.html

---

Acknowledgements

- Cache numbers were generated with computing resources provided by Wisconsin Condor, Midship (NSF 144-GB67), and Multifacet (NSF EIA-9971256) projects.
- Special thanks to David A. Patterson for advice in generating this data.
- Jason F. Cantin is supported by a Peter Schneider Wisconsin Distinguished Graduate Fellowship

---

Publications

- A subset of this data will appear in the third edition of John Hennessy and David Patterson's "Computer Architecture, A Quantitative Approach".

Disclaimer

Data in this directory is correct to the best of our knowledge. However, we provide it, *AS IS* without an expressed or implied warranty, and we accept no responsibility for the consequences of the use or misuse of this data.

Last updated July 2001 by Jason Cantin.