

# MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research

AJ KleinOsowski  
ajko@ece.umn.edu

David J. Lilja  
lilja@ece.umn.edu

Department of Electrical and Computer Engineering  
Minnesota Supercomputing Institute  
University of Minnesota, Minneapolis, MN 55455

*Abstract*— Computer architects must determine how to most effectively use finite computational resources when running simulations to evaluate new architectural ideas. To facilitate efficient simulations with a range of benchmark programs, we have developed the MinneSPEC input set for the SPEC CPU 2000 benchmark suite. This new workload allows computer architects to obtain simulation results in a reasonable time using existing simulators. While the MinneSPEC workload is derived from the standard SPEC CPU 2000 workload, it is a valid benchmark suite in and of itself for simulation-based research. MinneSPEC also may be used to run large numbers of simulations to find “sweet spots” in the evaluation parameter space. This small number of promising design points subsequently may be investigated in more detail with the full SPEC reference workload. In the process of developing the MinneSPEC datasets, we quantify its differences in terms of function-level execution patterns, instruction mixes, and memory behaviors compared to the SPEC programs when executed with the reference inputs. We find that for some programs, the MinneSPEC profiles match the SPEC reference dataset program behavior very closely. For other programs, however, the MinneSPEC inputs produce significantly different program behavior. The MinneSPEC workload has been recognized by SPEC and is distributed with Version 1.2 and higher of the SPEC CPU 2000 benchmark suite.

## I. INTRODUCTION

Computer architects have long used benchmark programs representative of real programs to test their computer designs. The benchmark suite from the Standard Performance Evaluation Corporation, commonly known as SPEC [2], is one example of a collection of programs used by the research community to test and rate current and future computer architecture designs [7].

As with most benchmark programs, the SPEC 1995 benchmark suite was developed with the then current and next generation computer systems in mind. Now, seven years later, computer technology has advanced to the point where the 1995 benchmarks execute in such a short time that they are no longer useful for evaluating the performance of different systems. On current state-of-the-art computer systems, several of the SPEC 1995 benchmark programs execute in less than one minute, for instance [2]. In an effort to lead the rapid progress of computer systems, SPEC chose to dramatically increase

the runtimes of the new SPEC 2000 benchmark programs [7] as compared to the runtimes of the SPEC 1995 benchmark programs. In addition, several new benchmarks were introduced to the suite to further extend its applicability to contemporary performance testing.

These long runtimes are beneficial when testing performance on native hardware. When evaluating new computer architectures using detailed execution-driven simulators, however, computer architects must decide if they should run fewer, very long simulations, or if they should seek smaller workloads and run a larger number of simulations within the time available.

When running the SPEC benchmark programs with the MinneSPEC input sets developed in this project [9], the resulting execution times are much shorter than when using SPEC’s original reference inputs. This time savings allows the architect to simulate a larger number of configurations with more benchmark programs than could reasonably be simulated with the complete SPEC reference inputs. Alternatively, MinneSPEC can be used to find a small number of promising design points which subsequently can be investigated in more detail using the original SPEC reference inputs.

Our goal was to develop a new benchmark workload which adequately exercises new architectures being simulated. The computer architecture community generally accepts SPEC as a benchmark suite for evaluating many aspects of a computer system, so we document the MinneSPEC program behavior by comparing the MinneSPEC profiles to the original SPEC profiles. MinneSPEC development began with gathering function-level execution, instruction mix, and memory behavior profiles of the SPEC 2000 benchmarks when executed with the original reference workload. These profiles gave us a target for the profiles that would be produced when executing with the MinneSPEC inputs.

## II. BACKGROUND AND MOTIVATION

SimpleScalar [1] is one example of an execution-driven simulator [4], [8], [11] commonly used by the computer architecture research community. SimpleScalar includes several simulators, each of which provides a different level of detail and statistical information about the sim-

ulated hardware. For most intricate computer architecture studies, researchers use *sim-outorder*, the most detailed simulator in the SimpleScalar suite.

At each stage of the *sim-outorder* pipeline, several dozen statistics and performance counters are updated for each simulated instruction. This level of simulation detail requires over 3,000 host machine cycles to simulate each target machine cycle. On a 333 MHz Sun UltraSparc system (similar to the SPEC CPU 2000 reference system), the simulated machine runs at approximately 45 KIPS (kilo-instructions per second). At that speed, the *188.ammp* benchmark, when using the full reference dataset of 1.9 trillion instructions, would take over sixteen months to simulate! An uninterrupted multi-month simulation is unrealistic for most computer architecture research. Considering the combinatorial explosion that occurs when examining numerous benchmark programs with a variety of hardware configurations, the time required for each simulation can result in years of CPU time to gather enough information for one simple study. Clearly, we need another workload set for use in simulation-based studies.

Detailed computer architecture research typically requires all of the statistics generated by a simulator such as *sim-outorder*. With this need in mind, using a less detailed simulator often is not an option. Various techniques have been proposed which either take representative samples of the execution [6], or which combine statistical and functional simulation [3], [12], [13]. In our case, we want full-detail execution from start to finish of the benchmark program. That means our best option is to find a quantitatively defensible way to reduce the input datasets, and, consequently, the runtimes, of the SPEC 2000 benchmarks. Our goal is not to exactly match the profiles obtained with the reference inputs. Instead, we attempt to provide another workload suitable for producing reportable results for simulation-based studies. In the process of creating this new workload, we analyze the differences in appropriate execution metrics produced by this new workload compared to the SPEC reference workload.

### III. METHODOLOGY

SPEC provides three standard datasets with their benchmark programs. The *test* dataset gives a quick test of the benchmark on the desired architecture, while the *train* dataset gives an intermediate length run. The *reference* dataset is intended to give a complete evaluation of the host computer system’s performance. These datasets are similar to our desired new workload except on a much larger time scale.

We began our analysis by compiling the benchmarks with the SimpleScalar version of gcc (version 2.6.3) on a Sun UltraSparc running Solaris. This modified version of gcc builds binaries for the simulator’s PISA architecture. We compiled at four different optimization levels,

O0 through O3. We then ran each SPEC dataset (*test*, *train*, and *ref*) with *sim-fast*, a simple simulator for determining instruction counts. The output from *sim-fast* gave us an indication of the size (in terms of instruction count) of each benchmark.

The *sim-profile* simulator, run with the SimpleScalar architecture binaries, gave us an instruction mix profile for the *reference* dataset. The *iclass* flag gave us a breakdown of instruction totals by class (i.e. loads, stores, unconditional branches, conditional branches, integer computation, floating-point computation, and traps) and the *iprof* flag gave us a count of each individual instruction executed.

In an effort to characterize the memory behavior of the SPEC 2000 benchmarks, we used the *sim-cache* simulator to obtain the miss rates of the level-1 data cache. We ran simulations with six different level-1 data cache sizes, 16k, 32k, 64k, 128k, 256k, and 512kbytes. In all of these simulations, we used a 32-byte block size and 4-way set associativity with an LRU replacement policy.

We proceeded to gather a function-level execution profile of the reference dataset. We recompiled the benchmarks with the Sun UltraSparc version of gcc using the *gprof* [5] compiler flags to insert profile counting routines. Once again, we compiled with the four optimization levels, O0 through O3. We ran the benchmark on native hardware to obtain an execution profile showing the fraction of total execution time spent in each function.

Now that we had base profiles to document the program behavior of the SPEC reference dataset, we began reducing the datasets for each benchmark. Our method of reducing the dataset varied widely from benchmark to benchmark. Our first choice was to vary the command line parameters to shorten execution time. If the benchmark did not have command line parameters, or if modifying the command line parameters did not shorten execution time, we proceeded to truncate or take representative samples from the input files. In a few cases, we created entirely new input files. The specific reduction technique used for each benchmark is shown in Table I.

The reference run of several SPEC CPU 2000 benchmarks use multiple runs of the same executable with different command line arguments or different input files. For these programs, we treat each of these different command lines as separate sub-benchmarks.

Our goal is to have three sets of inputs to provide simulation times of a few minutes, a few hours, and no more than a few days when executed on a detailed simulator, such as *sim-outorder*. After each reduction attempt, we used *sim-fast* to determine if the resulting runtime, measured in dynamic instructions, was in one of the three target ranges. Our targets were 100 million dynamic instructions for the short simulation, 500 million dynamic instructions for the medium length simulation, and 1 billion dynamic instructions for the full length, reportable, simulation.

TABLE I

BENCHMARKS INCLUDED IN THE CURRENT MINNESPEC INPUT SET DISTRIBUTION. THE *Input Reduction* COLUMN DOCUMENTS WHAT METHODS WERE USED TO DEVELOP THE MINNESPEC WORKLOADS. THE *FuncProf* AND *InstMix* COLUMNS DOCUMENT WHERE THE MINNESPEC FUNCTION-LEVEL EXECUTION PROFILE AND THE INSTRUCTION MIX ARE NEARLY IDENTICAL TO THE SPEC REFERENCE WORKLOAD, AS DETERMINED BY THE CHI-SQUARED TEST (SEE TEXT FOR DETAILS). FOR PROGRAMS WITH <NUM>[YN], THE MINNESPEC PROFILE IS COMPARED TO THE MULTIPLE SPEC REFERENCE WORKLOADS FOR THAT PROGRAM.

Benchmark	Type	Description	Input Reduction Method	FuncProf	InstMix
164.gzip,graphic	Int	Compression	modified command line, new input file	Y	N
164.gzip,log	Int	Compression	modified command line, truncated ref file	Y	N
164.gzip,program	Int	Compression	modified command line, new input file	Y	N
164.gzip,random	Int	Compression	modified command line, modified ref file	Y	N
164.gzip,source	Int	Compression	modified command line, truncated ref file	Y	N
175.vpr,place	Int	FPGA Place Route	same as test	Y	N
175.vpr,route	Int	FPGA Place Route	same as test	Y	N
176.gcc	Int	C Compiler	same as train	5Y1N	2Y4N
177.mesa	Float	3-D Graphics	modified command line, modified test file	N	N
179.art	Float	Image Rec	modified command line	2N	2N
181.mcf	Float	Combinatorial Opt	modified ref file	N	N
183.quake	Float	Seismic Wave	modified train file	N	N
188.amp	Float	Chemistry	truncated ref file	N	N
197.parser	Int	Word Processing	sampled ref file	N	N
253.perlbnk	Int	PERL Language	same as ref.makerand	2Y4N	1Y5N
255.vortex	Int	Database	modified train file	3Y	1Y2N
256.bzip2,graphic	Int	Compression	modified command line, modified ref file	Y	N
256.bzip2,program	Int	Compression	modified command line, new input file	Y	N
256.bzip2,source	Int	Compression	modified command line, sampled ref file	Y	N
300.twolf	Int	Place and Route	sampled train file	Y	N

#### IV. RESULTS

Our results to date document the program behavior of all SPEC CPU 2000 programs which are portable to SimpleScalar’s PISA instruction set, as shown in Table I. We are currently proceeding to reduce the input sets for the remainder of the SPEC CPU 2000 programs using the Compaq Alpha OSF UNIX platform.

We compared the MinneSPEC function-level execution profiles and instruction mix profiles to the profiles produced with the SPEC reference inputs using the chi-squared goodness-of-fit test [10]. The column labeled *FuncProf* in Table I shows the programs for which the function-level execution profiles matched when the programs were executed with both the MinneSPEC inputs and the reference inputs. In particular, a *Y* in this column indicates that the differences in the execution profiles produced with the two input sets was no larger than what would be expected due to random fluctuations, as measured with a chi-squared test at the 90 percent confidence level. An *N* in this column, however, indicates that the chi-squared test showed a statistically significant difference.

These comparisons show that the MinneSPEC function-level profiles are statistically close to the reference inputs for 21 out of the 33 benchmark configu-

rations evaluated. The *InstMix* column, however, shows that the mix of dynamic instructions produced with the MinneSPEC inputs is significantly different than that produced with the reference inputs in almost all cases.

Statistically comparing the memory behavior of the two workloads is problematic since we typically are interested in the variations of some parameter, such as the miss ratio, as another parameter, such as the cache size, is varied. Consequently, to compare the memory behavior produced by the two workloads, we plotted the miss rates as a function of cache size and workload, rather than comparing some sort of memory distribution parameter. Not surprisingly, these comparisons show that the MinneSPEC workload produces a substantially smaller memory footprint than the reference workload. These differences in memory behavior will cause differences in cache miss ratios and other important memory behaviors.

The actual histograms comparing the function-level profiles, the dynamic instruction execution mixes, and plots of the cache behavior for each benchmark when executed with the MinneSPEC inputs and the SPEC reference inputs are available at <http://www.arctic.umn.edu/~lilja/minnespec/>. Additionally, the full profiles and input sets of the bench-

marks we have completed to date are available to all SPEC licensees, either by ftp or on the CD distribution of the SPEC CPU 2000 suite, version 1.2.

The differences between MinneSPEC profiles and SPEC reference profiles mean that simulations which are sensitive to memory behavior, instruction mix, and so forth could produce substantially different results with MinneSPEC than with the reference inputs. These differences suggest that care must be taken when using both MinneSPEC and the SPEC reference workloads in the same study, as one would do when using MediaBench, TPC, or other workloads in combination with the SPEC reference workloads. That is, MinneSPEC should be treated as a separate workload.

## V. CONCLUSION

In this work, we develop new benchmark workloads for use in simulation-based studies. We compare the behavior of the SPEC CPU 2000 benchmark programs when executed with these new workloads to the program behavior with the reference workloads by examining: 1) the fraction of total execution time spent in functions when executed on a real system, as measured by the *gprof* profiling tool; 2) the dynamic mix of instructions, as measured with the *sim-profile* simulator from the SimpleScalar suite of simulators; and 3) the level-1 cache miss rates, as measured with the *sim-cache* simulator from SimpleScalar. While these metrics are far from complete, they do provide us with some appropriate points with which to compare the architectural-level behavior of simulations executed with the MinneSPEC inputs compared to those executed with the reference inputs.

A second contribution of this work is that our execution profile and cache miss results show that it is possible to obtain small datasets that reasonably mimic the behavior of the full reference datasets of the SPEC CPU 2000 benchmarks. We discovered, though, that developing input sets with exactly proportional profiles to the full reference profiles was very difficult for most programs, and outright impossible for others.

Nevertheless, the MinneSPEC inputs are suitable for conducting architectural studies using detailed execution-driven simulators. Although the MinneSPEC workloads are derived from the SPEC workloads, they are a unique entity. They may be used by themselves for reporting results, or they may be used to narrow the simulation parameter space before running a small number of lengthy simulations with the SPEC reference workload. Simulation study results generated with the MinneSPEC workload, however, should not be directly compared to the results generated with the SPEC reference workload. While these workloads are similar, they definitely are not the same.

## VI. ACKNOWLEDGEMENTS

This work was supported in part by National Science Foundation grants EIA-9971666 and CCR-9900605, NSF Research Experiences for Undergraduates grants EEC-9619750 and CCR-9610379, the Minnesota Supercomputing Institute, and Compaq's Alpha Development Group. A dedicated group of undergraduate research assistants made valuable contributions toward this project. To them, we extend our thanks: Nancy Meares, John Flynn, Khai Le, Matt Holmes, Alberto Baez, Thong Nguyen, Amy Josephson, Abdul Bahar, Syreeta Knight, Mark Nguyen, Mike Morse, Jeremy Kiser, Ilya Khazon, Kettly Joseph, Peter Holm, and Jed Deitrick.

## REFERENCES

- [1] Todd Austin, Eric Larson, and Dan Ernst. SimpleScalar: An infrastructure for computer system modeling. *IEEE Computer*, pages 59–67, February 2002.
- [2] Standard Performance Evaluation Corporation. SPEC benchmark suites. Various benchmark suites available. Details available at <http://www.spec.org>.
- [3] L. Eeckhout and K. DeBosschere. Hybrid analytical-statistical modeling for efficiently exploring architecture and workload design spaces. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 25–34, 2001.
- [4] Joel Emer, Pritpal Ahuja, Eric Borch, Artur Klauser, Chi-Keung Luk, Srilatha Manne, Shubhendu S. Mukherjee, Harish Patil, Steven Wallace, Nathan Binkert, Roger Espasa, and Toni Juan. Asim: A performance model framework. *IEEE Computer*, pages 68–76, February 2002.
- [5] S.L. Graham, P.B. Kessler, and M.K. McKusick. *gprof*: A call graph execution profiler. In *SIGPLAN Symposium on Compiler Construction*, pages 120–126, June 1982.
- [6] John Haskins and Kevin Skadron. Minimal subset evaluation: Rapid warm-up for simulated hardware state. In *International Conference on Computer Design (ICCD)*, September 2001.
- [7] John L. Henning. SPEC CPU 2000: Measuring CPU performance in the new millennium. *IEEE Computer*, 33(7):28–35, July 2000.
- [8] Christopher J. Hughes, Vijay S. Pai, Parthasarathy Ranganathan, and Sarita V. Adve. Rsim: Simulating shared-memory multiprocessors with ILP processors. *IEEE Computer*, pages 40–49, February 2002.
- [9] AJ KleinOowski, John Flynn, Nancy Meares, and David J. Lilja. Adapting the SPEC 2000 benchmark suite for simulation-based computer architecture research. *Workload Characterization of Emerging Computer Applications*, pages 83–100, 2001. Lizy Kurian John and Ann Marie Grizzaffi Maynard, editors. Paper originally published at the Workshop on Workload Characterization, International Conference on Computer Design (ICCD), September 2000.
- [10] David J. Lilja. *Measuring Computer Performance: A Practitioners Guide*. Cambridge University Press, New York, NY, 2000.
- [11] Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Halberg, Johan Hogberg, Fredrik Larsson, Adreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *IEEE Computer*, pages 50–58, February 2002.
- [12] S. Nussbaum and J.E. Smith. Modeling superscalar processors via statistical simulation. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 15–24, 2001.
- [13] Mark Oskin, Fred Chong, and M. Farrens. HLS: Combining statistical and symbolic simulation to guide microprocessor designs. In *International Symposium on Computer Architecture (ISCA)*, pages 71–82, 2000.