The Implications of Working Set Analysis on Supercomputing Memory Hierarchy Design

Richard Murphy⋆† Arun Rodrigues⋆† Peter Kogge⋆ Keith Underwood†

University of Notre Dame⋆
Computer Science and Engineering Department
384 Fitzpatrick Hall
Notre Dame, IN 46545

Sandia National Lab†*
PO Box 5800
MS-1110
Albuquerque, NM 87185-1110

{rcm,arodrig6,kogge}@cse.nd.edu, kdunder@sandia.gov

Abstract

Supercomputer architects strive to maximize the performance of scientific applications. Unfortunately, the large, unwieldy nature of most scientific applications has lead to the creation of artificial benchmarks, such as SPEC-FP, for architecture research. Given the impact that these benchmarks have on architecture research, this paper seeks an understanding of how they relate to real-world applications within the Department of Energy. Since the memory system has been found to be a particularly key issue for many applications, the focus of the paper is on the relationship between how the SPEC-FP benchmarks and DOE applications use the memory system. The results indicate that while the SPEC-FP suite is a well balanced suite, supercomputing applications typically demand more from the memory system and must perform more "other work" (in the form of integer computations) along with the floating point operations. The SPEC-FP suite generally demonstrates slightly more temporal locality leading to somewhat lower bandwidth demands. The most striking result is the cumulative difference between the benchmarks and the applications in terms of the requirements to sustain the floating-point operation rate: the DOE applications require significantly more data from main memory (not cache) per FLOP and dramatically more integer instructions per FLOP.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Performance Attributes

Keywords

Working Set, Supercomputing, Performance Modeling

*Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

ICS 2005 Cambridge, MA

1 Introduction and Motivation

There has been an explosion of interest in supercomputer architecture in the last few years in the wake of the Earth Simulator and the DARPA HPCS program. Most efforts are focusing on delivering sustained floating-point performance to scientific applications. Unfortunately, those scientific applications are known to be large, difficult to compile, difficult to run, and challenging to obtain representative datasets. Thus, all but the large industry players focus almost exclusively on benchmarks such as the SPEC Floating Point suite (SPEC-FP). This paper presents results from some real applications within the Department of Energy in use at Sandia National Labs. To our knowledge, this is the first study to discuss the architectural requirements of the given Sandia applications. We ask two simple questions: what are some of the fundamental requirements for making these applications run fast, and how do those relate to the SPEC-FP benchmark suite? The metrics for comparison used in this work are memory system demand and instruction mix.

It is well known that the memory wall presents significant challenges to computer architects in upcoming processor generations [19]. However, the problems associated with the von Neumann bottleneck are even more salient to scientific computing applications. While benchmarks, such as Stream Suite [18], or Giga-Updates Per Second (GUPS) measure memory performance directly, the memory requirements of data intensive and high performance scientific computing applications are less well understood. Furthermore, SPEC, the dominant benchmark for university and industry research, places a stronger emphasis on raw CPU performance than on the memory hierarchy[20]. When memory performance is studied, the focus tends to be on the performance of a given set of applications running on a particular memory hierarchy. Rather than performing yet another cache study, this work attempts to quantify the memory system requirements of an application by measuring the *temporal working set*, which is defined as the amount of data actively being used by the program during a particular phase of computation.

While the core work of both the SPEC-FP and Sandia suites is floating-point, they differ dramatically in their actual instruction mix. A common mistake in microprocessor research is to place a higher emphasis on providing peak floating-point capabilities than on balancing the microarchitecture to provide sustainable floating-point performance. One place that this becomes evident is in the relative number of "other" instructions to floating-point instructions, as has been highlighted in previous work[29] for multimedia bench-

marks. A coarse grained measure of integer versus floating-point operations shows that scientific applications perform significantly more integer operations per floating-point operation than SPEC-FP. Combining this with other measures, such as miss rate and data required from memory, yields insights into the significant differences in the number of bytes required from memory per FLOP that can be delivered. The result is a dramatically different overall balance point for scientific applications that needs to be the focus for supercomputer architects.

The remainder of this paper is organized as follows: Section 2 discusses the related work; Section 3 outlines the methodology for profiling the working set; Section 4 describes the benchmarks; Section 5 presents the results; and Sections 6 and 7: conclusions and future work.

2 Related Work

The focus of research into working set characterization mirrors changes in computer architecture research. Research on working sets in the 60s, 70s, and 80s, such as [8], were motivated by the need to develop efficient algorithms to swap virtual memory pages from main memory to a backing store. These works defined the "working set size" as the number of virtual memory pages touched over a given interval of instructions.

As efficient algorithms propagated and main memory got larger, the emphasis of later working set work changed, motivated now by the growing difference between main memory and processor performance (the von Neumann bottleneck). Later work, such as [10] or [7], used working set analysis to illuminate memory hierarchy design or compiler optimizations [11, 12]. As a result, the definition of working set changed. [26] showed that applications tend to have a hierarchy of working set sizes, and defined them as sizes of memory which could contain a significant portion of the programs' data. These working set sizes could be determined by the knees in graph of an applications cache miss rate against different caches sizes.

Just as the focus of working set work changed to fit the focus of architecture research, the methodologies used to explore working sets changed. Earlier papers [28, 9, 8] developed analytical models estimate working set evolution. These models were used to predict page reuse and inform virtual memory paging systems. Measurements and validation were performed with performance counters or small traces [25].

Later papers relied more upon trace analysis [17], full system simulation, or hardware performance counters [14]. Analytical models were sometimes developed to avoid slower simulation [13]. Characterization was generally done in terms of cache performance, generally looking at cache sizes up to a megabyte[26].

The SPEC suites have seen extensive working set characterization. [10] found SPEC92 benchmarks to have less than 5% cache miss rate with less than 40KB data cache. [12] used hardware counters to compare SPEC CPU2000 performance between different compilers. SPEC has also been used as a baseline when comparing against other workloads. [15, 17] used SPEC to compare various commercial, scientific, and desktop applications.

Database and OLTP processing have also been the subject of many characterization studies [14, 16, 7, 5]. Several other benchmarks have also been the target characterization. [30] analyzed the SPLASH2 benchmark, finding that the first important working set is usually less than 16KB. [4] looked at the OMP2001 benchmarks under different scaling conditions.

3 Methodology

This work primarily seeks to understand the evolution of the given benchmark's *working set*. To do so, two measurements were chosen: First, the miss rate of the working set, modeled as a true least recently used (LRU) cache of varying sizes over a large stream of instructions (in this case, four billion). Each block in the working set is a single word. This measurement provides an understanding of both the temporal locality required by the instruction stream, and the compulsory miss rate for the sampled stream. And second, the number of bytes required of the memory system for each floating point operation performed by the program for a given temporal working set size in the miss rate experiment. In addition to the working set information, this work provides more typical profiling information in the form of an instruction mix.

3.1 Tracing

Traces were gathered using the Amber instruction trace generator for the PowerPC, which is part of the CHUD toolkit [3]. Amber executes the native PowerPC binary, forcing the processor to emit information about each instruction executed. Programs were sampled for 4 billion instructions from the application's core computation. In the case of Sandia applications, this required profiling and source code analysis. Extensive work exists on analyzing the performance of SPEC. General guidance for choosing the beginning of the SPEC instruction traces was gathered from the publicly available SimPoint [27] information for Alpha binaries, hand examination of those binaries and extrapolation to the PowerPC versions, and hardware performance counter profiling on the PowerPC. In each case, hardware performance counters were used to verify the native machine performance of the chosen instruction stream, and the relevance of the sampled trace. While 4 billion instructions clearly does not encompass the entire application, there are two critical benefits to normalizing the samples to a fixed instruction stream: first, program startup, I/O, and other peripheral operations are eliminated from consideration; and second, by forcing the working set to be measured from a "cold start" the compulsory miss rates for the stream can be measured and compared.

The traces provide the instruction word for instruction executed during the trace period, annotated with appropriate information. Branches are annotated with the program counter of the next instruction to be executed, and memory operations with the effective address of that operation. Each trace is stored statically for later analysis, ensuring that separate programs profiling the data are given the same instruction stream. (Dynamic tracing is also possible, however some benchmarks use random numbers to perform differing analysis on the same set of inputs.)

3.2 Modeling and Analysis

Conceptually, the working set miss rate data is generated by constructing a 128 MB, fully associative, true LRU cache with a block size of 4 bytes (the native word size on the 32-bit PowerPC traces being analyzed). The working set is modeled in software as a 32M entry doubly linked list. During each load or store operation, the list is searched for the requested word address. If a hit is found, the position in the list is used as the block number, and a hit counter for that block is incremented. (The head of the list is position 1, and the

tail is position 32M.) That entry in the list is then promoted to position one in the list, as it was the most recently used item. The hit counters for each block are used to provide miss rate information for working sets varying in size from 1 block to 32M blocks.

For performance purposes, the list is divided into multiple sections, and a balanced red-black tree is used to locate which section a given address is in quickly. That section is then searched linearly for the address so that the position within the list can be found. The list in this experiment consisted of 16K sections, each of 2K entries. Promotion to the head of the list can be accomplished in constant time, and searching in log time for the section number, and linear time within the section. Given the size of the list, this performance enhancement is required to allow for a reasonable run time.

The instruction frequency information is gathered by examining each instruction word in the stream and classifying it according to its encoding.

3.3 Miss Rate Interpretation

Because the temporal working set's miss rate does not directly translate into the intuitive understanding of a cache miss rate, it is critical to understand the meaning of various points on the curve. These points are analogous to the same points on a standard cache curve, except that the working set curve strictly represents temporal locality. As such, it is better thought of as the minimum bandwidth multiplier required by the program for a working set of a given size.

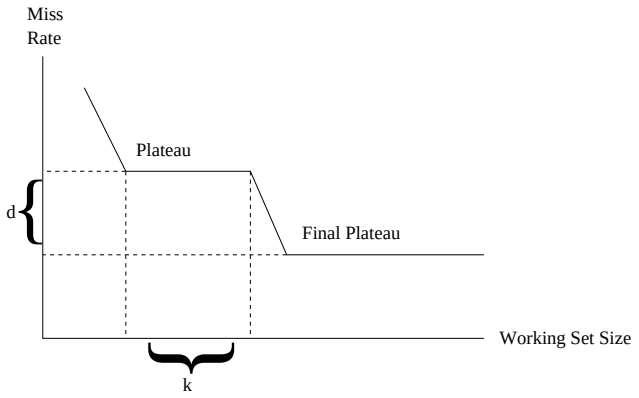


Figure 1. Miss Rate Interpretation

Figure 1 represents an annotated working set miss rate curve. The miss rate, as expected, is simply the percent of accesses that are not held in a working set of a given size. When the miss rate plateaus, additional growth in the size of the working set has no effect on the **number** of useful items captured. The final plateau represents the *compulsory miss rate*, or the probability that a given memory access over the interval will read truly new data. The difference between the miss rate at the final plateau and the miss rate at a given point (*d* in Figure 1) represents the probability that a particular access is examining "old" data not contained in the working set (e.g., that a smaller working set will have evicted due to size constraints). Similarly, *k* on the figure represents the increase in working set size required to overcome a plateau (that is essentially wasted until greater than *k* blocks are added).

3.4 Mean Performance Computation

In Section 5, each of the measures gathered experimentally include a "mean result" for the benchmark suite. In each case, the mean is computed by the same method. Since some programs in each suite contain multiple input sets, computing a straight mean for each run tends to overemphasize those benchmarks. Consequently, each mean is computed as the weighted sum *for each program* in the suite (rather than each run). In the case of SPEC-FP, there are 14 programs and 15 runs (ART has two input sets), consequently each ART input set only contributes $\frac{1}{28}$ to the mean. Similarly, the Sandia suite contains 3 programs and six runs, so each of 3 CTH runs contribute $\frac{1}{9}$th to the mean, and each of two LMP runs contributes $\frac{1}{6}$th to the mean.

4 Benchmarks

This study aims to compare scientific applications that typically consume supercomputer cycles to standard benchmarks used for microprocessor research. SPEC-FP is chosen to represent the most commonly studied computer architecture benchmark[6] of floating point performance. SPEC-FP is compared to a collection of important scientific applications at Sandia National Laboratories that are known to consume a significant portion of the compute cycles.

4.1 SPEC

The SPEC CPU2000 suite[2] is by far the most (currently) studied benchmark suite for processor performance[6]. This study focused on the SPEC-FP suite, as summarized in Table 1, because it is supposed to be most representative of scientific applications.

4.2 Scientific Applications

Scientific applications tend to be significantly different from common processor benchmarks. As such, the qualitative discrepancy between typical benchmarks and scientific applications has led to the specification of target applications as part of supercomputer procurements. Examples include the ASCI Purple Benchmark Suite[1] and the "7×" list for ASCI Red Storm[1]. This section briefly describes applications that were selected from common production applications at Sandia National Labs and we expect this set to grow as part of our ongoing efforts to understand real application behavior. Many of these applications have been previously studied for other properties[24].

4.2.1 LAMMPS

LAMMPS is a classical molecular dynamics (MD) code designed to simulate systems at the atomic or molecular level[22, 21, 23]. Typical applications include simulations of proteins in solution, liquid-crystals, polymers, zeolites, or simple Lenard-Jones systems. It runs on any parallel platform that supports the MPI message-passing library or on single-processor workstations. The original version was written in Fortran90 and consists of approximately 30,000 lines of code[2]. The version under examination in this work is an enhanced version of the original written primarily in C++. In a typical simulation, some combination of the features of LAMMPS

[1]This is a list of 10 applications that are expected to run 7 times faster on ASCI Red Storm than on ASCI Red.

[2]This text adapted with permission from http://www.cs.sandia.gov/ sjplimp/lammps.html.

Benchmark	Programming Language	Description
168.wupwise	Fortran 77	Physics - Quantum Chromodynamics
171.swim	Fortran 77	Shallow Water modeling
172.mgrid	Fortran 77	Multi-grid Solver
173.applu	Fortran 77	Parabolic PDEs
177.mesa	C	3d Graphics
178.galgel	Fortran 90	Computational Fluid Dynamics
179.art	C	Adaptive Resonance Theory Neural Net
183.equake	C	Seismic Wave Propagation
187.facerec	Fortran 90	Face Recognition
188.ammp	C	Computational Chemistry
189.lucas	Fortran 90	Primary Number Testing
191.fma3d	Fortran 90	Finite Element Crash Simulation
200.sixtrack	Fortran 77	High Energy Physics Accelerator Design
301.apsi	Fortran 77	Meteorology: Pollutant Distribution

Table 1. SPEC CPU2000 Floating Point Suite

is used. Thus, three different input problems that demonstrated different characteristics were chosen for analysis. These were:

1. *Lennard Jones Mixture*: This input simulated a 2048 atom system consisting of three different types
2. *Chain*: simulates 32000 atoms and 31680 bonds.

4.2.2 CTH

CTH is a multi-material, large deformation, strong shock wave, solid mechanics code developed at Sandia National Laboratories. CTH has models for multi-phase, elastic viscoplastic, porous and explosive materials. Three-dimensional rectangular meshes; two-dimensional rectangular, and cylindrical meshes; and one-dimensional rectilinear, cylindrical, and spherical meshes are available. It uses second-order accurate numerical methods to reduce dispersion and dissipation and produce accurate, efficient results. CTH is used extensively within the Department of Energy laboratory complexes for studying armor/anti-armor interactions, warhead design, high explosive initiation physics and weapons safety issues. It consists of approximately 500,000 lines of Fortran and C.

CTH has two fundamental modes of operation: with or without adaptive mesh refinement (AMR). Adaptive mesh refinement changes the application properties significantly and is useful for only certain types of input problems. Therefore, we have chosen one AMR problem and two non-AMR problem for analysis.

Three input sets were examined:

1. **2-Gas:** The input set uses an $80 \times 80 \times 80$ mesh to simulate two gases intersecting on a 45 degree plane.
2. **Explosively Formed Projectile (EFP):** The simulation represents a simple Explosively Formed Projectile (EFP) that was designed by SNL staff. The original design was a combined experimental and modeling activity where design changes were evaluated computationally before hardware was fabricated for testing. The design features a concave copper liner that is formed into an effective fragment by the focusing of shock waves from the detonation of the high explosive. The measured fragment size, shape, and velocity is accurately (within 5%) modeled by CTH.
3. **CuSt AMR (AMR):** This input problem simulates a 4.52 km/s impact of a 4 mm copper ball on a steel plate at a 90 degree angle. Adaptive mesh refinement is used in this problem.

4.2.3 sPPM

The sPPM benchmark is part of the ASCI Purple benchmark suite[1] as well as the $7\times$ application list for ASCI Red Storm. It solves a 3D gas dynamics problem on a uniform Cartesian mesh using a simplified version of the PPM (Piecewise Parabolic Method) code. The hydrodynamics algorithm requires three separate sweeps through the mesh per timestep. Each sweep requires approximately 680 FLOPs to update the state variables for each cell. The sPPM code contains over 4000 lines of mixed Fortran 77 and C routines. The problem solved by the sPPM involves a simple, but strong (about Mach 5) shock propagating through a gas with a density discontinuity. sPPM is the only Department of Energy application under study used as part of a defined benchmark suite.

5 Results

This section describes the results of four key measures of application performance: the instruction mix, the temporal working set's miss rate, the bandwidth required from the memory system for that miss rate, and the number of bytes/flop demanded by each application. Together, these properties define a significant portion of the balance that an architecture requires to deliver sustained floating-point performance.

5.1 Instruction Mix

Figures 2(a) and (b) show the instruction mixes for the benchmarks and applications studied, with the means shown in Figures 2(c). These results clearly indicate that the SPEC-FP applications perform significantly more floating point operations than the Sandia applications; however, the overall ratio between computation and memory operations is approximately the same. Given that, at their core, both suites are performing floating point work, it is clear that the Sandia codes are more complex. They require more integer work (e.g., address calculations, logical operations, etc.) to perform the core floating point operation. The only Sandia application that performs large amounts of floating point work (typical of the SPEC-FP suite) is the most synthetic (sPPM) in that it comes from the ASCI Purple Suite and is packaged as a benchmark rather than a full application. The three CTH runs and the two LAMMPS runs perform approximately 10% floating point. The SPEC-FP suite shows both a significantly higher ratio of floating point operations on av-

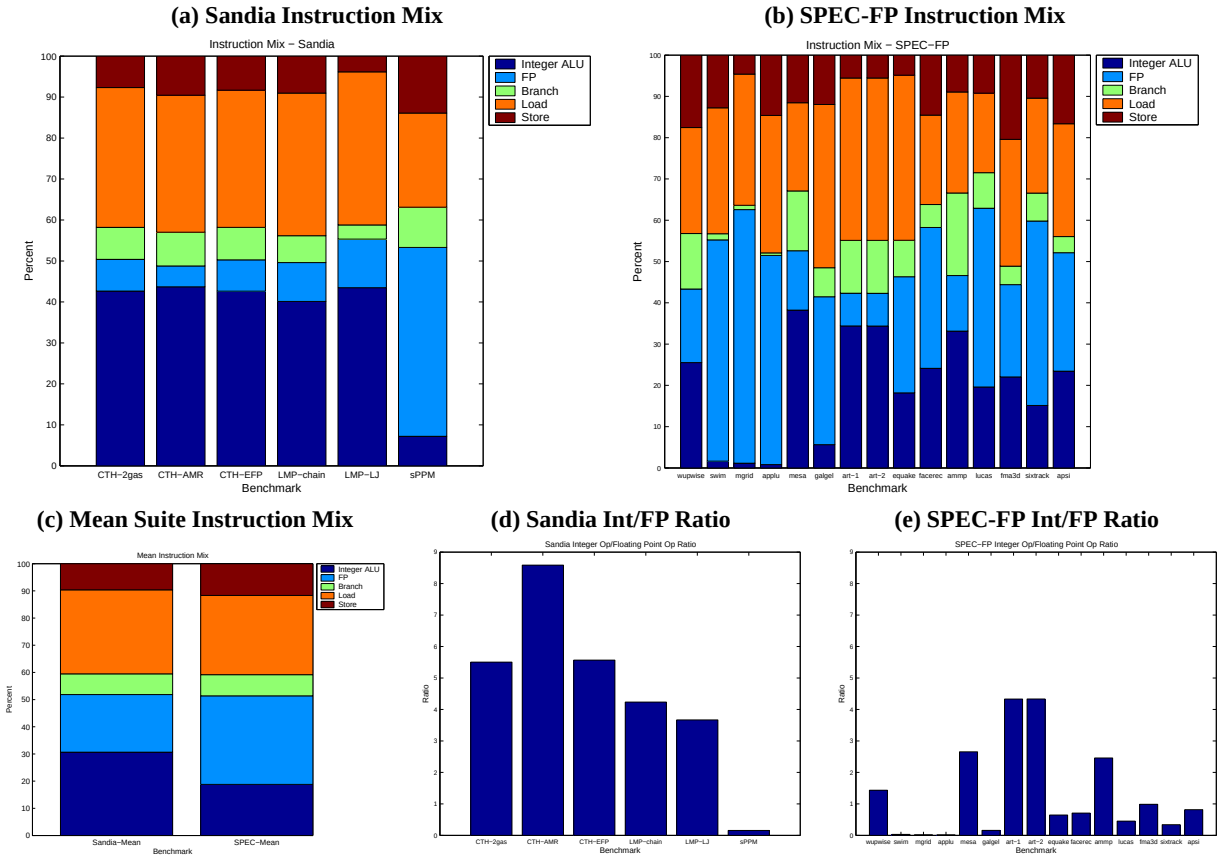


Figure 2. Instruction Mix Information (a-c) and Integer/Flop Ratios (d-e)

erage, and in the individual benchmarks (only the two ART runs perform less than 10% floating point operations). In fact, Figures 2(d) and (e) show that the Sandia applications perform, on average, more than 3.5 times the number of integer operations per flop that SPEC-FP does.

Two other interesting observations can be made from Figure 2. While the benchmark suites perform a similar number of memory operations, the SPEC suite, on average, performs more stores. As a first order approximation, this demonstrates that the Sandia codes consume more memory for each result generated. Also, the variance among the SPEC-FP codes is significantly greater. Partially this is a reflection of the quality of the SPEC suite in choosing a broad range of applications. However, it also shows that radically different supercomputing applications share remarkably similar properties.

5.2 Miss Rate

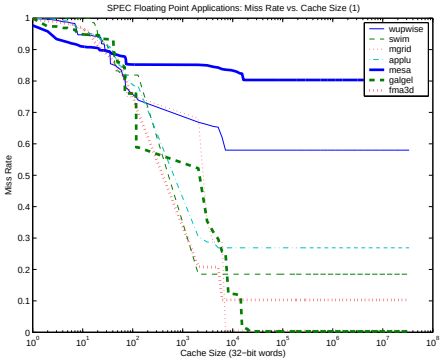
Figure 3 shows the individual and summary miss rates for the temporal working set when modeled as described in Section 3. The results demonstrate that while a small working set is slightly more beneficial for the Sandia applications, as the data is fully captured the Sandia applications benefit significantly less. In general, this shows that the Sandia applications operate on a significantly larger working set (as more "new data" is required over the given instruction interval). The Sandia codes have both a larger intrinsic working set (e.g., they operate on more data), and a larger stride (e.g., they tend to use data from farther back in time).

The *miss rate* in these results should be looked at more as a bandwidth multiplier than as a "cache miss rate". The working set, as modeled, ignores all spatial locality effects. Consequently, it truly represents the minimum number of words required by the program (for a given size). The overarching question is: given a working set size, how many bytes of old data are required, and how many bytes of unseen data are required? The miss rate represents both of these ratios.

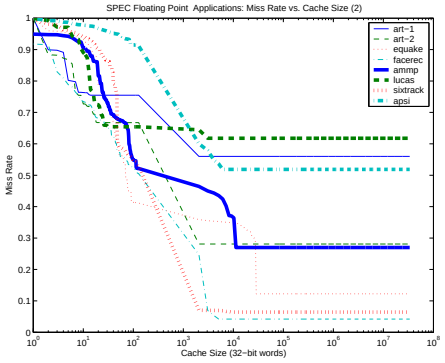
5.3 Bandwidth

Figure 4 shows the raw bandwidth requirements for both suites (note that the y-axis is in gigabytes). While the SPEC-FP applications show, on average, a greater absolute bandwidth requirement from the memory system in the region significantly smaller than the L1 cache on most processors, the Sandia applications require, on average, a greater bandwidth utilization in the larger working set region. Furthermore, the Sandia curves show significantly less improvement as the available working size increases. The individual Sandia curves are relatively flat, while virtually all of the SPEC-FP curves show large drops (typically before reaching an L1 cache size). In fact, nearly a third of the SPEC-FP suite requires more bandwidth than the Sandia applications in the very-small working set region; however, once the temporal locality is captured, the Sandia applications require more bandwidth from the memory system.

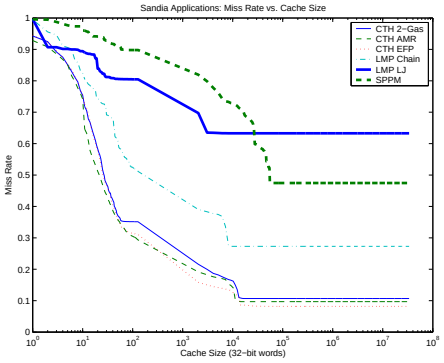
(a) SPEC-FP Miss Rate (1)



(b) SPEC-FP Miss Rate (2)



(c) Sandia Miss Rate



(d) Mean Miss Rate

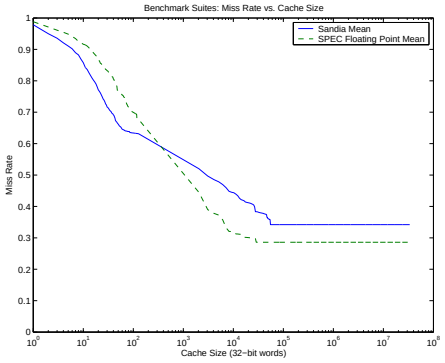
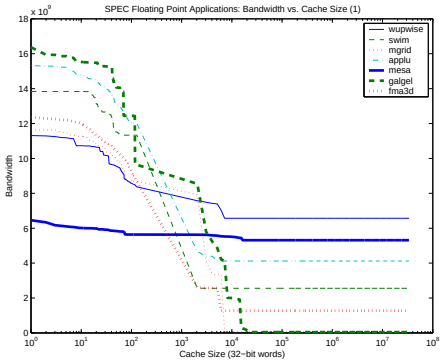
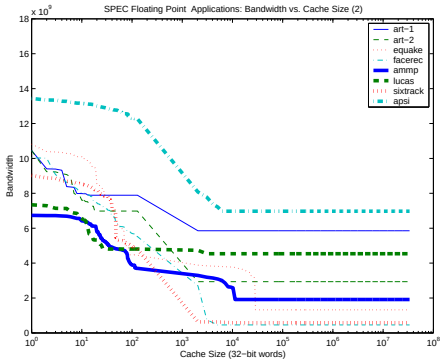


Figure 3. Miss Rates

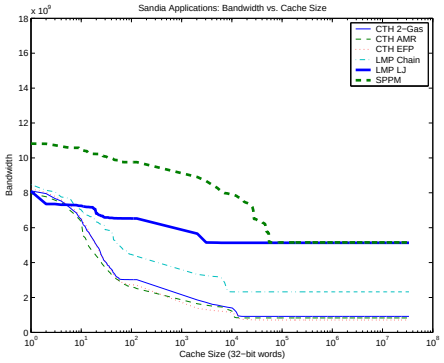
(a) SPEC-FP Bandwidth (1)



(b) SPEC-FP Bandwidth (2)



(c) Sandia Bandwidth



(d) Mean Bandwidth

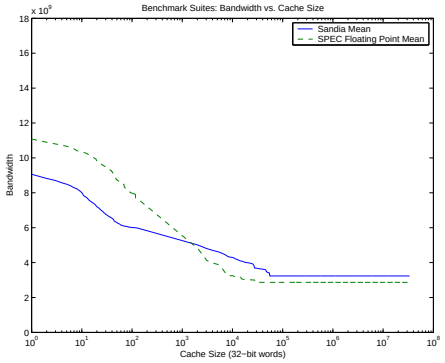


Figure 4. Raw Bandwidth

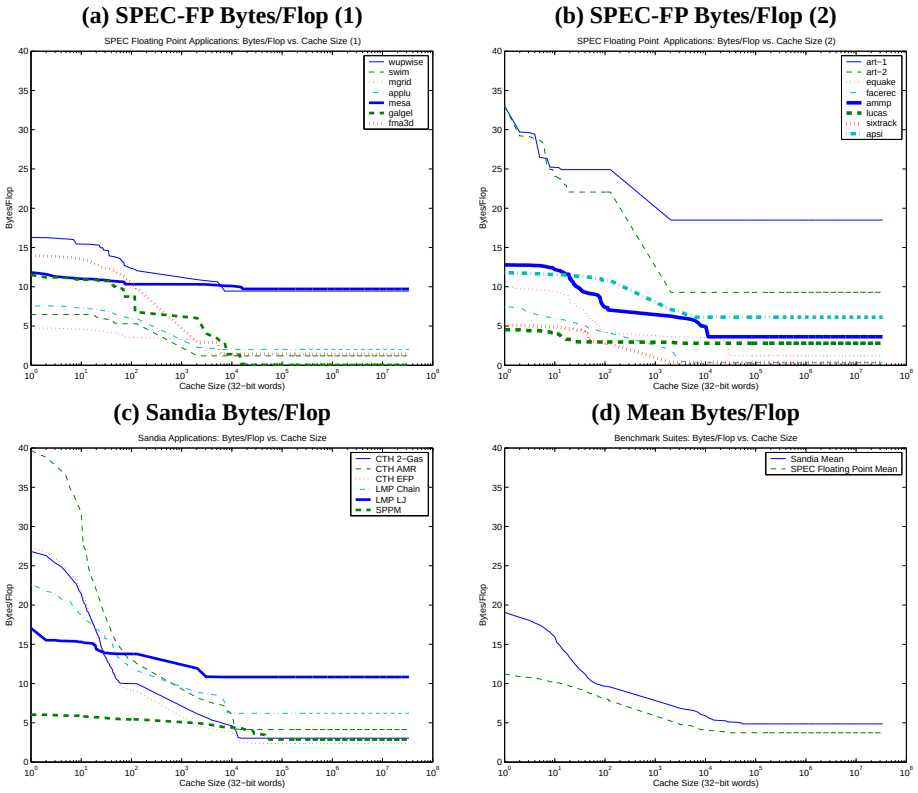


Figure 5. Bytes Per Flop

5.4 The Bytes per FLOP Balance

Since sustained FLOPs is the ultimate goal of scientific computing, the final metric considers the number of bytes needed from memory for each floating-point operation performed. The bytes per FLOP balance factor combines the effects from both the raw bandwidth (Figure 4) and the instruction mix data shown in Figure 2. The bytes per FLOP summary metric is shown in Figure 5. It shows that the Sandia applications demand significantly more from the memory system on a per flop basis (between 30% and 74% more, in fact). The SPEC-FP suite shows a much more smooth curve in the region representing a typical L1 cache size (32 KB or less). In terms of the individual results, the ART runs from the SPEC-FP suite have the greatest bytes per flop requirement of any application, but represent outliers in the SPEC-FP suite. The ART runs begin with over twice the bandwidth requirement of any other SPEC application, and show data capture in the region of a standard L1 cache. (This is to be expected as the SPEC benchmarks are generally designed for cache capture.) sPPM gives the minimum byte per flop ratio of any Sandia application for very large working set sizes (with a minimum of 2.87 bytes/flop). Ten of the SPEC applications (two thirds of the suite) bottom out below the lowest Sandia application. On the high end, only the two ART input sets in the SPEC-FP suite demands more than 20 bytes/flop of data for a cache size of one word, whereas only the LMP-LJ and sPPM input sets in the Sandia suite demand less. The curves for the averages have a striking difference from both the miss rate and bandwidth curves: the Sandia average is consistenly higher than the SPEC-FP average over the full range of cache sizes.

The bytes/flop ratio represents the critical figure in understanding the relationship between floating point performance and memory performance. Given the importance of the von Neumann bottleneck, a balanced floating point and memory configuration are significantly more important than raw floating point performance in evaluating potential supercomputing architectures. Furthermore, SPEC-FP does not provide an optimal measure of the demands placed on the system by supercomputing applications. While it clearly achieves its goal of measuring raw floating point performance, it is off balance in measuring effective supercomputing application performance. While the raw bandwidth numbers look somewhat similar, it is the balance that is key in evaluation.

6 Conclusions

In conclusion, this work has compared a set of real-world supercomputing applications to the SPEC-FP suite in terms of their intrinsic memory requirements (the temporal working set) and their floating point performance. From the results, it is clear that the Sandia suite of applications behaves very differently from the SPEC-FP benchmarks, which are designed to account for only raw floating point performance. Given that, on average, a Sandia program will perform 3.5 times the number of integer operations as a SPEC-FP program, it is clear that measures of raw floating point performance are not sufficient to characterize the architectures designed for those applications.

Minimum memory bandwidth requirements are examined in terms of the application's *temporal working set*, which removes spatial locality effects from classical cache models and focuses on the amount of **reuse** of recently touched data, and on the amount of truly new data required. The results demonstrate that the Sandia codes hit a higher miss rate plateau later, which indicates that they

have both larger working sets and less reuse per data item. This provides insight to the intrinsic properties of Sandia applications for supercomputer memory hierarchy designers.

Finally, when the bandwidth is considered in the context of the instruction mix, it becomes clear that the Sandia codes require, on average, between 30% and 74% more bandwidth per flop than their SPEC-FP counterparts. As intrinsic program characteristics rather than performance observations on a given architecture, these measures should impact both supercomputer architects and supercomputer benchmark suite designers. Furthremore, they provide a calibration point between reported SPEC-FP results and the requirements of the supercomputing community.

7 Future Work

The continuation of this work will naturally examine the orthogonal intrinsic measure of memory performance, specifically the effectiveness of spatial locality in supercomputing and SPEC-FP codes. Additionally, the bandwidth requirements of the code segment, and differences between supercomputing and traditional benchmarks have yet to be analyzed, though initial analysis has shown that supercomputing applications tend to have significantly longer basic blocks [24].

Acknowledgments

This work would not have been possible without the support and help of a number of individuals. Steve Plimpton (Sandia National Labs) provided LAMMPS, the appropriate input sets for it, and valuable assistance in describing those inputs. Sue Goudy (Sandia National Labs) provided valuable help in CTH and its input sets. Thanks also go to Jeffrey Vetter and John Engle for their help with sPPM.

The application analysis and data gathering discussed here was funded in part by Sandia National Laboratories, using tools funded in part by DARPA through Cray, Inc. under the HPCS Phase 1 program.

8 References

[1] ASC Purple benchmark codes, July 2004. http://www.llnl.gov/asci/purple/benchmarks/limited/code_list.html.

[2] SPEC website, July 2004. http://www.spec.org.

[3] Apple Architecture Performance Groups. *Computer Hardware Understanding Development Tools 2.0 Reference Guide for MacOS X*. Apple Computer Inc, July 2002.

[4] Vishal Aslot and Rudolf Eigenmann. Performance characteristics of the spec omp2001 benchmarks. *SIGARCH Comput. Archit. News*, 29(5):31–40, 2001.

[5] Luiz André Barroso, Kourosh Gharachorloo, and Edouard Bugnion. Memory system characterization of commercial workloads. In *Proceedings of the 25th annual international symposium on Computer architecture*, pages 3–14. IEEE Computer Society, 1998.

[6] Daniel Citron, John Hennessy, David Patterson, and Gurindar Sohi. The use and abuse of SPEC: An ISCA panel. *IEEE Mico*, 23(4):73–77, July-August 2003.

[7] Z. Cvetanovic and D. Bhandarkar. Characterization of alpha AXP performance using TP and SPEC workloads. In *Proceedings of the 21ST annual international symposium on Computer architecture*, pages 60–70. IEEE Computer Society Press, 1994.

[8] Peter J. Denning. The working set model for program behavior. In *Proceedings of the first ACM symposium on Operating System Principles*, pages 15.1–15.12. ACM Press, 1967.

[9] Domenico Ferrari. A generative model of working set dynamics. In *Proceedings of the 1981 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 52–57. ACM Press, 1981.

[10] Jeffrey D. Gee, Mark D. Hill, Dionisios N. Pnevmatikatos, and Alan Jay Smith. Cache performance of the SPEC benchmark suite. Technical Report CS-TR-1991-1049, 1991.

[11] Somnath Ghosh, Margaret Martonosi, and Sharad Malik. Cache miss equations: An analytical representation of cache misses. In *International Conference on Supercomputing*, pages 317–324, 1997.

[12] Swathi Tanjore Gurumani and Aleksandar Milenkovic. Execution characteristics of spec cpu2000 benchmarks: Intel c++ vs. microsoft vc++. In *Proceedings of the 42nd annual Southeast regional conference*, pages 261–266. ACM Press, 2004.

[13] Magnus Karlsson and Per Stenström. An analytical model of the working-set sizes in decision-support systems. In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 275–285. ACM Press, 2000.

[14] Kimberly Keeton, David A. Patterson, Yong Qiang He, Roger C. Raphael, and Walter E. Baker. Performance characterization of a quad pentium pro SMP using OLTP workloads. In *International Sympoisum on Computer Architecture*, pages 15–26, 1998.

[15] Dennis C. Lee, Patrick Crowley, Jean-Loup Baer, Thomas E. Anderson, and Brian N. Bershad. Execution characteristics of desktop applications on windows NT. In *International Symposium on Computer Architecture*, pages 27–38, 1998.

[16] Jack L. Lo, Luiz Andre Barroso, Susan J. Eggers, Kourosh Gharachorloo, Henry M. Levy, and Sujay S. Parekh. An analysis of database workload performance on simultaneous multithreaded processors. In *International Symposium on Computer Architecture*, pages 39–50, 1998.

[17] Ann Marie Grizzaffi Maynard, Colette M. Donnelly, and Bret R. Olszewski. Contrasting characteristics and cache performance of technical and multi-user commercial workloads. In *Proceedings of the sixth international conference on Architectural support for programming languages and operating systems*, pages 145–156. ACM Press, 1994.

[18] McCalpin, John D. *Stream: Sustainable memory bandwidth in high performance computers*, 1997.

[19] Sally A. McKee. Reflections on the Memory Wall. In *Computing Frontiers (CF04), Ischita, IT*, April 2004.

[20] SPEC Open Systems Steering Committee. Spec cpu 2000 run and reporting rules (revised). March 15, 2001.

[21] Steven J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal Computation Physics*, 117:1–19, 1995.

[22] Steven J. Plimpton. Lammps web page, July 2004.

http://www.cs.sandia.gov/ sjplimp/lammps.html.

[23] Steven J. Plimpton, R. Pollock, and M. Stevens. Particle-mesh ewald and rRESPA for parallel molecular dynamics. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, Minneapolis, MN, March 1997.

[24] Arun Rodrigues, Richard Murphy, Peter Kogge, and Keith Underwood. Characterizing a new class of threads in scientific applications for high end supercomputers. In *Proceedings of the 2004 International Conference on Supercomputing (ICS2004)*, St. Malo, France, June 2004.

[25] Juan Rodriguez-Rosell. Empirical working set behavior. *Commun. ACM*, 16(9):556–560, 1973.

[26] Edward Rothberg, Jaswinder Pal Singh, and Anoop Gupta. Working sets, cache sizes, and node granularity issues for large-scale multiprocessors. In *Proceedings of the 20th annual international symposium on Computer architecture*, pages 14–26. ACM Press, 1993.

[27] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. Automatically characterizing large scale program behavior. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2002), San Jose, CA*, October 2002.

[28] D. R. Slutz and I. L. Traiger. A note on the calculation of average working set size. *Commun. ACM*, 17(10):563–565, 1974.

[29] D. Talla, L.K. John, and D.C. Burger. Bottlenecks in multimedia processing with simd-style extensions and architectural enhancements. *IEEE Transactions on Computers*, 52(8), August, 2003.

[30] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the 22th International Symposium on Computer Architecture*, pages 24–36, Santa Margherita Ligure, Italy, 1995.