SPEC Developing New Component Benchmark Suites

By Kaivalya Dixit (Sun Microsystems, Inc.), Jeff Reilly (Intel)

Having established SPEC Release 1 in 1989, SPEC has continued to work on adding value to its CPU Benchmark suites. These efforts have culminated in new benchmark suites, the SPEC CPU integer suite (CINT2.0) and SPEC CPU floating-point suite (CFP2.0). These suites, created to replace SPEC Release 1, are expected to be formally announced in January 1992.

Why A New CPU Release?

SPEC Release 1 is two years old. New technologies (hot chips), architectures, smart preprocessors, and aggressive compiler optimizations have dramatically improved CPU performance. SPEC Release 1 is under attack from new technologies and shows signs of aging.

As SPEC has always pointed out to the public, one number is never representative of performance. The original SPEC suite contained integer and floating point benchmarks which represented vastly different application areas. There was also an imbalance in the number of benchmarks, four integer and six floating point. Combining these into the one metric, SPECmark, effectively hid useful information and emphasized floating point performance. This, along with market forces, necessitated the creation of SPECint and SPECfp. The creation of two separate suites, each with its own metric, SPECint92 and SPECfp92, clearly distinguishes two distinct areas where CPU performance can be compared.

Advances in technology have improved SPECmarks. With only ten benchmarks, even when using the geometric mean, one SPECratio could drastically alter the composite numbers. Thus effort was spent in ensuring that more classes of applications were represented. The developers of the new SPEC CPU suites have remedied many of the problems encountered in SPEC Release 1 by carefully analyzing and modifying the existing benchmarks.

Component Suites

The new integer suite (CINT2.0) contains six benchmarks and the floating point suite (CFP2.0) contains containing 14 benchmarks. The prefix "C" in CINT2.0 and CFP2.0 identifies these as component benchmark suites.

The computational characteristics of the suites are as follows:

- CINT2.0 - Six integer benchmarks
- CFP2.0 - Nine double precision floating point benchmarks and five single precision floating point benchmarks

Component Metrics

SPEC decided to introduce a new metric for each suite, SPECint92 and SPECfp92. Including the year of release within the metric name will prevent confusion with both past and future releases. The metrics from SPEC Release 1.2b will be now be referred to by SPEC as SPECint89 and SPECfp89.

The SPECratio for a benchmark is the quotient derived from dividing the SPEC Reference Time (elapsed time on VAX-780) by a particular machine's run time.

Table 1: Integer Suite (CINT2.0)

Benchmark	Application[1]	Source Size		Object Size	
		Lines	Words/Line	Text (KB)	Data+Bss (KB)
008.espresso	Circuit Theory	14838	3.6	208	64
022.li	Lisp Interpreter	7741	3.1	176	56
023.eqntott	Logic Design	3454	3.5	80	313
026.compress	Unix Utility	1503	4.2	56	450
072.sc	Spreadsheet	8485	3.7	208	131
085.gcc	GNU C compiler	87791	4.1	736	144

[1]Written in the C language

Table 2: CINT2.0: Dynamic Instruction Count Percentages[1]

Benchmark	LOAD+STORE %	NO-OPS %	Branches %	Others %
008.espresso	28.2	0.3	20.4	51.2
022.li	33.3	6.4	23.8	36.5
023.eqntott	16.5	0.1	26.5	56.9
026.compress	25.5	1.8	16.5	56.2
072.sc	23.5	4.3	21.6	50.3
085.gcc	26.6	1.0	20.2	52.1
TOTAL (GM)[2]	25.0	1.1	21.3	50.0

[1]Based on SPARC instruction set
[2]Geometric mean of instruction category

SPECint92 is the Geometric Mean (GM) of the six SPECratios of the CINT2.0 suite.

SPECint92 = $\sqrt[6]{\text{Product of 6 SPECratios}}$

SPECfp92 is the Geometric Mean (GM) of the 14 SPECratios of the CFP2.0 suite.

SPECfp92 = $\sqrt[14]{\text{Product of 14 SPECratios}}$

Integer Suite (CINT2.0)

The SPEC integer suite (CINT2.0) contains six benchmarks—008.espresso, 022.li, 023.eqntott, 026.compress, 072.sc and 085.gcc. All six benchmarks are written in C. Table 1 (page 14) describes the application area, the size of source code and the static size of SPARC executable of each benchmark. The size of the source code ranges from 1,503 to 87,791 lines of source code, and the static size of the executable ranges from 232 to 880 KB. The words/line provides a simple-minded code complexity measure.

Dynamic Instruction Profile for CINT2.0

The benchmarks were analyzed with SpixTools, a set of SUN internal performance measurement tools. The data presented below were extracted from the output of spixstats. The salient run time behavior (e.g. memory reference, branch, no-ops, and instruction count (percentages) are detailed in Table 2 (above). Table 2 shows the range of instruction mix for the CINT2.0 suite:

- Load/Store operations - 17% to 33%
- Branch operations - 17% to 27%
- Other (ALU and miscellaneous instructions) - 37% to 57%

The CINT2.0 suite executed a total of about 18 billion instructions.

Floating Point Suite (CFP2.0)

The SPEC floating point suite (CFP2.0) contains 14 benchmarks — 013.spice2g6, 015.doduc, 034.mdljdp2, 039.wave5, 047.tomcatv, 048.ora, 052.alvinn, 056.ear, 077.mdljsp2, 078.swm256, 089.su2cor, 090.hydro2d, 093.nasa7 and 094.fpppp. Twelve of the CFP2.0 benchmarks are written in FORTRAN and two benchmarks are written in C. Nine benchmarks perform double precision arithmetic and five benchmarks perform single precision arithmetic. Table 3 (page 16) describes the application area, the size of source code and the static size of SPARC executable of each benchmark. The size of the source code ranges from 184 to 18,912 lines of source code, and the static size of the executable ranges from 198 to 14,622 KB.

Dynamic Instruction Profile for CFP2.0

The benchmarks were analyzed with SpixTools. The data presented below were extracted from the output of spixstats. The salient run time behavior (e.g. memory reference, branch, no-ops, floating point operations, and instruction count (percentages) are detailed in Table 4 (page 17).

Table 4 shows the range of instruction mix for the CFP2.0 suite:

- Load/Store operations - 13% to 48%
- Floating point operations (FPOPs) - 4% to 56%
- Branch operations - 2% to 14%
- Other (ALU and miscellaneous instructions) - 15% to 57%

The CFP2.0 suite executed a total of about 93.7 billion instructions. Obviously, the benchmarks containing a high percentage of floating point, load/store and branch operations are more likely to cause pipeline stalls, and cache misses. This type of information along with cache miss data is useful in identifying performance bottlenecks.

The instruction mix information is highly dependent on the architecture, instruction set, compiler efficiency and libraries. The benchmarks were compiled for the Sun SPARC instruction set with varying optimization levels (-02 to -04). The instruction mix will be different on other platforms (e.g. HP, IBM, Intel, Motorola).

Summary

The SPEC Release 1 will be de-emphasized starting January 1992. The CINT2.0 and CFP2.0 suites are better suites than the Release 1 because:

- Integer and floating point benchmarks are in separate suites.
- Most benchmarks read data from an input file to prevent unrealistic optimizations from constant propagation.
- 001.gcc35 is replaced by a longer running 085.gcc and removes the need for alternate results file.
- CINT2.0 includes new application areas like compression algorithms, and spreadsheets.
- 020.nasa7 is replaced by 093.nasa7 which provides performance data on individual kernels and reads parameters from an input file to prevent unrealistic optimizations.
- CFP2.0 includes single precision benchmarks and two of them are in C.
- CFP2.0 includes new application areas like quantum chemistry, robotics, shallow water models, quantum physics and astrophysics.

CINT2.0 and CFP2.0 still suffer from following problems:

- Most benchmarks do not stress caches. HP has performed analysis indicating that all of the benchmarks fit comfortably in just a 32 KB instruction cache and all have a miss rate of less than 1% for a 128 KB instruction cache.

SPEC is actively soliciting portable applications from critics who complain about kernel and synthetic benchmarks.

Credits and Caveats

We would like to thank Sanjay Jain, Prasad Wagle, Walter Bays, Rajiv Khemani, Nhan Chu, and Raj Dixit of Sun Microsystems for making this readable. We are indebted to

Table 3: Floating-point Suite (CFP2.0)

Benchmark	Application	Source Size		Object Size	
		Lines	Words/Line	Text (KB)	Data+Bss (KB)
013.spice2g6	Circuit Design (FDP)[1]	18912	2.6	480	7939
015.doduc	Monte-Carlo Simulation (FDP)	5334	1.7	272	134
034.mdljdp2	Quantum Chemistry (FDP)	4456	3.3	224	230
039.wave5	Maxwell's Equation (FSP)[2]	15062	2.4	344	14288
047.tomcatv	Coordinate Translation (FDP)	184	3.6	160	3636
048.ora	Optics- Ray Tracing (FDP)	533	3.9	168	30
052.alvinn	Robotics (CSP)[3]	272	2.9	88	490
056.ear	Model human ear (CSP)	5237	3.3	144	65
077.mdljsp2	Quantum Chemistry (FSP)	3883	3.2	224	191
078.swm256	Shallow Water Model (FSP)	487	3.8	184	3638
089.su2cor	Quantum Physics (FDP)	2514	2.2	248	4117
090.hydro2d	Astrophysics (FDP)	4448	3.7	224	327
093.nasa7	NASA Kernels (FDP)	1177	4.0	216	2842
094.fpppp	Quantum Chemistry (FDP)	2718	2.2	232	330

[1](FDP) - FORTRAN Double Precision
[2](FSP) - FORTRAN Single Precision
[3](CSP) - C Single Precision

Table 4: CFP2.0 - Dynamic Instruction Count Percentages[1]

Benchmark*	LOAD+STORE %	FPOPS %	NO-OPS %	Branches %	Others %
013.spice2g6	24.98	4.23	0.46	13.56	56.78
015.doduc	29.42	25.94	2.61	8.44	33.58
034.mdljdp2	26.67	47.67	4.38	1.99	19.28
039.wave5	33.69	21.91	1.56	7.38	35.45
047.tomcatv	45.03	32.84	0.03	1.78	20.33
048.ora	13.20	56.33	2.20	7.39	20.87
052.alvinn	48.34	27.63	0.29	5.09	18.64
056.ear	35.32	26.37	1.64	6.18	30.50
077.mdljsp2	21.38	52.77	5.07	2.37	18.42
078.swm256	39.16	34.17	0.01	1.65	25.01
089.su2cor	33.33	31.78	0.56	4.32	30.00
090.hydro2d	28.48	38.00	2.89	3.94	26.69
093.nasa7	38.89	28.00	0.29	4.98	27.84
094.fpppp	47.21	34.71	0.44	2.96	14.68
TOTAL (GM)[2]	31.57	29.15	0.65	4.28	25.43

[1]Based on SPARC instruction set
[2]GM: Geometric mean of the instruction category

many SPEC members who worked on these suites at benchathons. We are grateful to Reinhold P. Weicker (Siemens) for support and helpful hints. We want to thank Robert Cmelik (Sun) for developing spixstats and creating analysis data for 085.gcc. Finally, we want to thank Bud Funk (Unisys) and Subra Balan (IBM) for editing.

Kaivalya Dixit is the SPEC president and an engineering program manager for Sun Microsystems, Inc. of Mountain View, California. Jeff Reilly is the Release Manager for the SPEC CINT2.0 and CFP2.0 suites and works for Intel in Santa Clara, California.

SDM *(continued from page 8)*

UNIX commands usage. 057.sdet is formulated to use system resources in a proportion identical to a commercial software development environment. These are valid underpinnings for the workload to be considered representative of user software development environments.

SDM benchmarks are applicable directly when the systems being compared are to be used for software development. A user may wish to see the actual mix of commands executed by 057.sdet and 061.kenbus1 before determining whether one or both are relevant to their environment. The analysis and interpretation sections of this article covered in detail the aspects of comparing computer system performance based on these workloads.

SPEC SDM 1 and SPEC Release 1 are useful tools for predicting application performance in many of today's UNIX Workstation/Server user environments. SPEC is currently working on benchmark suites to address additional applications and environments.

Krishna Dronamraju works at AT&T Bell Laboratories in Naperville, Illinois. Subra Balan works at the IBM Performance Evaluation Center in Roanoke, Texas. Tom Morgan works at Data General in Westborough, Massachusetts. All are computer system performance specialists who regularly represent their companies at SPEC meetings, and contribute to the organization's technical work.