SPEC CPU Suite Growth: An Historical Perspective

John L. Henning, Sun Microsystems
Contact: john dot henning at acm dot org

Prehistory

Since 1989, the SPEC CPU benchmarks have aspired to ambitious goals: fair, portable, comparable tests using the compute-intensive portion of real applications. It may be difficult today to remember just how much of a challenge these goals presented when SPEC was first founded, or how much of a break they were from previous industry practice.

A typical example, recently discovered while cleaning a basement, is a 1985 Performance Guide. Containing 190 pages, it is well-produced, with glossy covers, many graphs and tables, and a wide variety of tests. But for the most part, performance results are presented with only sparse descriptions of the conditions of observation (workload, versions of software, details of the hardware). The document – which obviously had much effort put into it and which would clearly be of great interest to customers – is officially marked "Internal Use Only", although one suspects that this marking may often have been ignored. Results are given for only one vendor's product line, with no competitive comparisons. If a third party wanted to generate their own competitive comparisons, they would be unable to do so, because the source code of the measured applications was not available.

At first glance, the exception appears to be four benchmarks with public source code: the 1985 Performance Guide does contain results for Dhrystone, LINPACK, Livermore Loops, and Whetstone. But the competitive comparisons are not given, and in some cases it is stated that the benchmark has been modified, without giving details of how or why. Finally, even if these four tests had been presented in a comparable form, these benchmarks are subject to the criticism that they are small, organized around short loop "kernels".

In short, although interesting performance work was done by vendors in the 1980s, it tended to be opaque, non-comparable, non-reproducible, and proprietary.

Comparable Results using Compute-Intensive, Portable Source from Real Applications

By contrast, SPEC values transparent benchmarks that can be compared across platforms and reproduced. For the first 15 years, SPEC's president, Kaivalya Dixit, evangelized the SPEC development culture with the principle that proprietary interests should be subordinated to the interest in technically credible, fair, vendor-neutral benchmarks.[1] An early, and often-repeated, SPEC motto is that "an ounce of honest data is worth a pound of marketing hype".

SPEC CPU benchmarks use portable source code with strict limits on modifications. In contrast to earlier portable benchmarks, SPEC CPU seeks real application code. Where Whetstone was 284 lines long (including comments), the first SPEC CPU suite included the Gnu C compiler as 001.gcc, with 138,901 lines of code, and 013.spice2g6 with 22,386.

Not all the benchmarks from that first set honored the goal of being real applications: for example, 020.nasa7 was a set of 7 kernels that were created as a benchmark, not as an application. As later versions of the suite were introduced, there have been others. But it is a substantial negative point during discussion of a benchmark candidate to say that it is "just a benchmark", and it is a substantial positive point to say "that's a real application with real users". Over the years, SPEC has included more real application code and proportionately less code that is "just" a benchmark.

SPEC trims I/O from CPU benchmarks, while preserving the intended areas of testing: CPU, memory, and compiler. What else should be trimmed? If a program has 10 major modules, and the supplied workload uses only 3 of them, should the other 7 be dropped? What about error pathways – should those be trimmed? Recent suites have tended to keep as much as practical, so as to remain closer to the original application.

The desire for real applications, and the hesitancy to trim them, are primary reasons why the SPEC CPU suites have grown, as summarized in Figure 1. On the following pages, Figures 2 and 3 detail the sizes of individual benchmarks, first in lines of code and then in terms of number of source files.

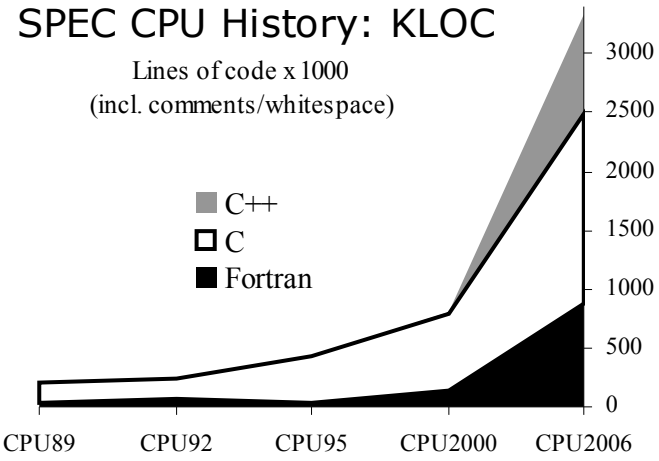

Figure 1: Suite growth. The term "CPU89" is an anachronism: it was then known as "SPECmark", and "CPU92" was then called "SPEC92". But since 1995, SPEC has often released more than one benchmark in a single year, as it has expanded to include web services, graphics, Java, etc. Therefore, for over 10 years, names have used the form SPEC <benchmark> <year>. (Note: there is no such thing as SPEC2006, nor did SPEC2000 ever exist.)

Figure 2: KLOC (Lines of Code x1000, incl. comments/whitespace)

Benchmark	.f	.c	.h
CPU89			
001.gcc1.35 (i)		116	23
008.espresso (i)		17	2
022.li (i)		19	.3
023.eqntott (i)		4	.1
013.spice2g6 (f)	21	2	
015.doduc (f)	5		
020.nasa7 (f)	2		
030.matrix300	.4		
042.fpppp (f)	3		
047.tomcatv (f)	.2		
CPU89 Totals	32	157	25

Benchmark	.f	.c	.h
CPU92 Integer			
008.espresso		14	1
022.li		7	.3
023.eqntott		3	.1
026.compress		2	
072.sc		8	.5
085.gcc		105	22
CPU92 FP			
013.spice2g6	37	1	
015.doduc	5		
034.mdljdp2	4		
039.wave5	8		
047.tomcatv	.4		
048.ora	.5		
052.alvinn		.3	
056.ear		5	.4
077.mdljsp2	4		
078.swm256	.5		
089.su2cor	3		
090.hydro2d	4		
093.nasa7	1	.05	
094.fpppp	3		
CPU92 Totals	70	145	25

Benchmark	.f	.c	.h
CPU95 Integer			
099.go		29	.7
124.m88ksim		18	2
126.gcc		194	16
129.compress		1	.5
130.li		7	.7
132.ijpeg		28	3
134.perl		24	3
147.vortex		53	15
CPU95 FP			
101.tomcatv	.2		
102.swim	.4		
103.su2cor	2		
104.hydro2d	4		
107.mgrid	.5		
110.applu	4		
125.turb3d	2		
141.apsi	7		
145.fpppp	3		
146.wave5	8		
CPU95 Totals	32	353	41

Benchmark	.f	.f90	.c	.h	.C
CPU2000 Integer					
164.gzip			8	1	
175.vpr			17	1	
176.gcc			210	18	
181.mcf			2	1	
186.crafty			19	2	
197.parser			11	.5	
252.eon				18	23
253.perlbmk			62	24	
254.gap			59	12	
255.vortex			53	15	
256.bzip2			5	.01	
300.twolf			20	1	
CPU2000 FP					
168.wupwise	2				
171.swim	.4				
172.mgrid	.5				
173.applu	4				
177.mesa			50	11	
178.galgel		15			
179.art			16		
183.equake			2		
187.facerec		2			
188.ammp			13	.2	
189.lucas		3			
191.fma3d		60			
200.sixtrack	48				
301.apsi	7				
CPU2000 Totals	63	81	542	102	23

Benchmark	.f	.f90	.c	.h	.C	.hh
CPU2006 Integer						
400.perlbench			124	46		
401.bzip2			7	1		
403.gcc			485	36		
429.mcf			2	1		
445.gobmk			190	7		
456.hmmer			33	3		
458.sjeng			13	1		
462.libquantum			3	1		
464.h264ref			46	5		
471.omnetpp				17	31	
473.astar				2	4	
483.xalancbmk			6	1	296	251
CPU2006 FP						
410.bwaves	1					
416.gamess	466					
433.milc			13	2		
434.zeusmp	37					
435.gromacs	23		72	13		
436.cactusADM	3		87	13		
437.leslie3d	4					
444.namd				3	2	
447.dealII				100	82	17
450.soplex				14	27	
453.povray				14	141	
454.calculix	44		97	26		
459.GemsFDTD		12				
465.tonto		165				
470.lbm			1	.3		
481.wrf		128	29	57		
482.sphinx3			18	7		
CPU2006 Totals	579	305	1228	370	252	267

SPEC CPU89 – then called "SPECmark" – did not have a distiction between SPECfp and SPECint.

Benchmarks marked "(i)" in Figure 2 and Figure 3 were later classifed as integer; those marked "(f)" were later classified as floating point

Figure 3: SPEC CPU Modules (source code files)

Benchmark	.f	.c	.h
CPU89			
001.gcc1.35 (i)		79	94
008.espresso (i)		50	9
022.li (i)		44	1
023.eqntott (i)		26	2
013.spice2g6 (f)	13	3	
015.doduc (f)	41		
020.nasa7 (f)	2		
030.matrix300	1		
042.fpppp (f)	41		
047.tomcatv (f)	1		
CPU89 Totals	99	202	106

Benchmark	.f	.c	.h
CPU92 Integer			
008.espresso		44	7
022.li		22	1
023.eqntott		22	2
026.compress		1	
072.sc		11	5
085.gcc		75	98
CPU92 FP			
013.spice2g6	24	2	
015.doduc	41		
034.mdljdp2	2		
039.wave5	2		
047.tomcatv	2		
048.ora	2		
052.alvinn		1	
056.ear		16	6
077.mdljsp2	1		
078.swm256	1		
089.su2cor	2		
090.hydro2d	2		
093.nasa7	21	1	
094.fpppp	28		
CPU92 Totals	138	195	119

Benchmark	.f	.c	.h
CPU95 Integer			
099.go		17	4
124.m88ksim		98	24
126.gcc		67	66
129.compress		2	1
130.li		22	3
132.ijpeg		70	23
134.perl		20	18
147.vortex		66	57
CPU95 FP			
101.tomcatv	1		
102.swim	1		
103.su2cor	2		
104.hydro2d	2		
107.mgrid	1		
110.applu	1		
125.turb3d	1		
141.apsi	1		
145.fpppp	38		
146.wave5	2		
CPU95 Totals	50	362	196

Benchmark	.f	.f90	.c	.h	.C
CPU2000 Integer					
164.gzip			14	6	
175.vpr			20	21	
176.gcc			66	61	
181.mcf			11	14	
186.crafty			39	4	
197.parser			17	1	
252.eon				172	151
253.perlbmk			38	58	
254.gap			32	31	
255.vortex			66	57	
256.bzip2			2	1	
300.twolf			75	10	
CPU2000 FP					
168.wupwise	22				
171.swim	1				
172.mgrid	1				
173.applu	1				
177.mesa			58	65	
178.galgel		38			
179.art			9		
183.equake			1		
187.facerec		11			
188.ammp			28	3	
189.lucas		1			
191.fma3d		101			
200.sixtrack	124				
301.apsi	1				
CPU2000 Totals	150	151	476	504	151

Benchmark	.f	.f90	.c	.h	.C	.hh
CPU2006 Integer						
400.perlbench			59	67		
401.bzip2			9	3		
403.gcc			155	123		
429.mcf			11	14		
445.gobmk			67	29		
456.hmmer			57	15		
458.sjeng			19	4		
462.libquantum			16	15		
464.h264ref			42	39		
471.omnetpp				69	85	
473.astar				8	11	
483.xalancbmk			19	2	738	1012
CPU2006 FP						
410.bwaves	5					
416.gamess	151					
433.milc			68	21		
434.zeusmp	56					
435.gromacs	7		124	131		
436.cactusADM	6		265	180		
437.leslie3d	1					
444.namd				21	11	
447.dealII				141	116	194
450.soplex				60	63	
453.povray				109	100	
454.calculix	192		205	93		
459.GemsFDTD		18				
465.tonto		245				
470.lbm			2	4		
481.wrf		138	29	78		
482.sphinx3			44	50		
CPU2006 Totals	418	401	1194	1278	1124	1206

In both Figure 2 and Figure 3, the column labelled:

.f also includes **.F**

.f90 also includes: **.F90**, **.int**, **.use**

.h also includes **.def**, **.inc**

.C also includes **.cpp** **.cc**

.hh .also includes **.hpp**, **.icc**

The CPU2006 benchmarks mentioned in Figures 2 and 3 are described at [2], and earlier benchmarks at [3]. The grand totals for each suite are:

	KLOC	Modules
CPU89	214	407
CPU92	240	452
CPU95	425	608
CPU2000	811	1,432
CPU2006	3,334	5,621

Other factors affecting suite growth

In addition to the desire for real applications, and a hesitancy to trim them, other factors have also contributed to the growth of the SPEC CPU suites in recent years.

Implementation languages have changed. The earlier suites used Fortran-77 and C. More recently, Fortran 90 and C++ have come into use. C++ is, of course, an Object-Oriented Programming language; and the largest Fortran 90 program, 465.tonto, is written in an OOP style. While one would not wish to cast aspersions on OOP, it is probably fair to say that encapsulation, message passing, and abstraction are somewhat unlikely to be associated with brevity.

Search programs have brought new codes. Both CPU2000 and CPU2006 solicited contributions. The "Benchmark Search Programs" were advertised via the SPEC web site, relevant Usenet groups, and *Computer Architecture News*.[4] Contributors provided source code and workloads, assisted with porting, helped solve validation issues, and were compensated up to $5000. A list of successful entries to the CPU2006 benchmark search program may be found at [5].

"More is better." The CPU subcommittee has had several discussions of whether having more benchmarks makes a better suite. Several concerns have been raised to argue against large suites. (1) SPEC CPU2006 has a grand total of 3.3 million lines of code. From a maintainability perspective, having this much code is a risk. It is a certainty that any program set this large has bugs. SPEC hopes that the worst of them have been shaken out during testing, and that the remainder will not affect the usefulness of performance results. But clearly, a larger suite carries more opportunities for bugs. (2) As the number of benchmarks grows, it becomes increasingly likely that hardware designers will not bother to, or will not be able to, simulate the entire suite. (3) As the number of benchmarks grows, some researchers may even complain of "duplication" in the suite. [6]

Despite these concerns, the majority of the subcommittee tended to agree that "more is better", for several reasons. (1) Having more benchmarks allows more application areas to be represented. (2) Having more benchmarks allows more programming styles to be represented. Consider, for example, 416.gamess vs. 465.tonto: both are chemistry codes. Both are written in Fortran. ONE OF THEM IS WRITTEN IN UNABASHED OLD-STYLE FORTRAN, AND HAS BEEN AROUND SINCE BEFORE THE INVENTION OF LOWER CASE LETTERS. The other is far more recent, and makes unabashed use of the features of Fortran-95 (with grudging support for Fortran-90). After discussion, the subcommittee decided that both programming styles deserve representation. (3) Having more benchmarks may tend to encourage compiler developers to implement optimizations that help a wide variety of programs, while decreasing the payoff from more narrowly targeted optimizations. (4) Although two programs may appear to "duplicate" each other on today's hardware, there is always the possibility that one of them – and only one of them – may turn out to be sensitive to some difference in tomorrow's system implementations.

Larger codes are useful for compiler QA. The SPEC CPU suites are performance packages, not quality assurance packages. Nevertheless, it is clear that SPEC CPU development efforts expose bugs in both candidate benchmarks and in compilers. For example, as of February 28, 2005, SPEC had testing reports for 52 candidate benchmarks on 33 platforms, with various versions of compilers, operating systems, and hardware. At that moment, there were 135 unsuccessful tests which were tentatively believed to be the fault of various platforms under test, rather than the fault of candidate benchmarks.

In particular, the SPEC CPU suites contribute to optimizer quality. End users and ISVs may feel relatively little motivation to report optimizer bugs to compiler vendors, because it is easy to just turn down the dial on the optimizer. ("Oh, it fails with `-O5`? Well, does it succeed at `-O3`?"). The ISV may not want to spend the effort to determine the root cause of the problem – e.g. program bug, standards violation, or actual compiler problem. But for SPEC CPU, compiler developers are highly motivated to use the optimizer, and highly motivated to find root causes.

Of course, compiler QA test suites include thousands of tests, only a small portion of which are derived from SPEC. This article does not mean to exaggerate SPEC's QA role. But the SPEC tests do provide a contribution to quality. Within the SPEC CPU subcommittee, if a benchmark candidate is said to be "a compiler challenge", that is usually considered to be a compliment; and larger codes, such as 481.wrf and 483.xalancbmk, have posed challenges.

Large applications are available as open source software. Finally, this article would be remiss if it did not emphasize the fact that much of the growth of the SPEC CPU suites has relied upon the growth of the open source movement. The credits for SPEC CPU2006 [5] list various benchmarks that SPEC adapted in whole or in part from open source software, and SPEC is grateful to the open source community for the availability of these programs.

References

[1] See www.spec.org/spec/kaivalya

[2] John L. Henning (ed.), "SPEC CPU2006 Benchmark Descriptions", Computer Architecture News, Volume 34, No. 4, September 2006.

[3] www.spec.org/<suite name>, e.g. www.spec.org/cpu95

[4] The CPU2006 search program page is archived at www.spec.org/cpu2005/search

[5] www.spec.org/cpu2006/docs/credits.html

[6] See for example A.Phansalkar, A. Joshi and L. John, "Subsetting the SPEC CPU2006 Benchmark Suite" *Computer Architecture News,* Volume 35, No. 1, March 2007