IEEE TCCA Newsletter, June, 1996, pp. 1-8

A Brief Introduction To The SPEC CPU95 Benchmarks

Jeff Reilly
Chair, SPEC CPU Subcommittee
jwreilly@pro.specbench.org

Introduction and History

The Standard Performance Evaluation Corporation (SPEC) established itself in 1988 as a non-profit corporation devoted to "establishing, maintaining and endorsing a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers." [1] SPEC's first product was the SPEC Benchmark Release 1 suite, later identified as SPEC89, which provided a standardized comparative measure of compute-intensive microprocessor performance. This product replaced the vague and confusing MIPS and MFLOPS ratings then used in the computer industry [2]. SPEC had developed benchmarks from real applications and provided them as source code to allow compilation on various UNIX workstations. While this did include the memory subsystem and the compiler in the performance measurement, it did ensure that all platforms would perform precisely the same task, providing a comparative measure across different architectures.

But time progressed and also brought change. Technology drastically improved processor, system and compiler performance. And because of this, benchmarks were required to evolve and adapt to continue to be useful. SPEC obsoleted SPEC89 with the release of SPEC CPU92 in January 1992. In the summer of 1995, SPEC replaced SPEC CPU92 with an improved SPEC CPU95, with the subcomponents CINT95 (focusing on integer/non-floating point compute-intensive activity) and CFP95 (focusing on floating-point compute-intensive activity).

Goals Of The Benchmarks

Webster's II Dictionary defines a benchmark as: "A standard of measurement or evaluation". [3] In the computer industry, such standards are typically set up either to provide a means of predicting task performance on a single machine, to provide a diagnostic measure or to provide a common yardstick for comparing different machines. Being driven by market forces (to create a better comparative metric than "MIPS"), SPEC emphasizes the latter approach for the SPEC CPU benchmarks.

For the SPEC CPU95 benchmarks (as with the SPEC CPU benchmarks before it), the goals were to:

- Create a comparative compute intensive benchmark to replace "MIPS", "MFLOPS" and older SPEC CPU benchmarks.
- Be highly portable; remove hardware and OS dependencies.
- Create a benchmark that had the support of the industry.
- Keep the benchmark simple to run and report; improve the tools and availability of results.
- Use real applications that were "meaningful".
- Keep the benchmark different from previous version; to encourage general improvements as opposed to test specific optimizations.

Note that with these goals some compromises had to be made. As mentioned, aiming for portability meant that the benchmarks had to begin as source code and that the benchmarks would stress the (processor, memory, compiler) component of a computer system. Also, with such wide portability targets, a wide base of users would be covered, raising questions on how to select applications. In the end, this led to a lot of discussion and voting sessions during the SPEC selection process; availability of source code and portability issues also influenced the selection of code. SPEC then had to ensure that the reporting format included the details for each benchmark as well as a composite metric, to allow the user to use the numbers they felt were most appropriate for their interests (moving the determination of representitiveness to the end user).

Note that SPEC CPU95 is not designed to measure graphics, networking, I/O or operating system features. While it may be possible to intentionally configure a degenerate system such that those components affect SPEC CPU95 performance, this is neither the intent nor focus of SPEC CPU95. It is difficult to characterize system performance with a single benchmark and these components are covered by other SPEC and industry benchmarks. [4]

Choosing metrics

The choice of metrics also ran into a simplicity versus "meaningful" debate. While one overall number is simple, it hides too much information. On the other hand having multiple metrics and numbers can lead to confusion. In the end, SPEC choose to have multiple metrics and require all vendors publicizing results to make available upon request, the details behind the number. Note that his again requires the user of the data to understand what the benchmarks are and how they are measured and determine how they map to what their interests.

In terms of metrics, there are several distinct areas/categories for compute-intensive performance that were considered and chosen:

Integer versus floating point:

These are two distinct types of computation as reflected in different

software data types or functional units on a processor. SPEC recognized this by creating SPEC CINT95 and CFP95 as separate suites, each with its own metrics.

Conservative versus aggressive computation:

To achieve portability to a wide variety of platforms, SPEC provides the benchmarks as source code. But how should this source code be compiled? Some people test every possible compiler option to squeak the last bit of performance out of a compiler. Others simply use the standard options provided by the compiler vendor, and yet others may use a combination of both. While SPEC recognizes it cannot capture everyone's model of usage in a single metric, SPEC has defined a required baseline measure (with more conservative rules on how the applications are to be compiled) and an optional nonbaseline measure (with more aggressive rules governing compilation). This provides a required conservative point of reference plus an optional, more tuned reference point.

Speed versus throughput:

People, especially those with multi-threaded environments, are often not only concerned with how fast they can get a single task done but also the rate at which they can get multiple tasks done (throughput). With this in mind as well, the SPEC95 suites provide a comparative measure for speed of a single compute-intensive task and for a multiple process throughout metric (SPECrate).

SPEC has provided a metric for each permutation of the above choices leading to eight possible metrics in the SPEC95 suites:

	SPEED	**THROUGHPUT**
Aggressive Compilation	SPECint95 SPECfp95	SPECint_rate95 SPECfp_rate95
Conservative Compilation	SPECint_base95 SPECfp_base95	SPECint_rate_base95 SPECfp_rate_base95

These metrics provide the user with the ability to choose the criteria that best matches his or her need. (Note that this does require the users to be aware of the environment with which they are concerned.)

Details of the Benchmark Suite

CINT95 and CFP95 are based on compute-intensive applications provided as source code. After examining over 40 potential candidates (using criteria such as portability, verification of output and application area), SPEC selected 8 programs for SPEC CINT95 and 10 programs for SPEC CFP95. Included below are the SPEC provided descriptions of these benchmarks. [5]

CINT95 (all C programs)

099.go

SPEC classifies this program as an game-playing/artificial intelligence program involving lots of pattern matching and look ahead. It is based on an internationally ranked 'Go' playing-program called 'The Many Faces of Go' by David Fotland. The cache activity is small, but does require something larger than an 8KB cache to run well. Cross module optimization can allow inlining of low-level list manipulation routines (which otherwise can add up to about 1/3rd or the processing time).

124.m88ksim

124.m88ksim is a chip simulator for the Motorola 88100 microprocessor. It is a reasonably full implementation of a chip simulator, including cache latencies. The simulator is loaded with versions of Dhrystone and a memory tester. Due to costs of simulation, the choice of binary loaded into the simulator does not have much impact upon the profile of execution, either at the software or hardware level.

126.gcc

126.gcc is based on the GNU C compiler version 2.5.3 from the Free Software Foundation. The benchmark generates optimized code for a SPARC based system over about 50 input sources. The 50 input files means that this benchmark has the highest numbers of fork(2)/exec(2) during the benchmark run, and also the highest numbers of open(2) and other such system calls. 126.gcc, along with 147.vortex, is one of the largest tests in the CINT95 suite.

129.compress

129.compress is based on the common UNIX compression utility. This version has been modified to do its work in memory, rather than reading and writing files, to avoid disk I/O. It generates a large buffer-full of data, compresses that into another memory buffer, checks the size of the compressed data and then decompresses the buffer and verifies that the resulting data is the same size as the original. The buffer is then modified slightly and compressed again.

130.li

130.li is the same Lisp interpreter as found in the SPEC CPU92 benchmark suite, except that the workload has been changed to include a version of the Gabriel benchmarks from the "Performance Evaluation of Lisp Systems" by

Richard Gabriel. Cross module optimization (inlining) would be helpful in getting best results.

132.ijpeg

132.ijpeg performs image compression/decompression on in-memory images based on the JPEG facilities. Like 129.compress, the common utility has been adapted to run out of memory buffers rather than reading/writing files to avoid disk I/O. The benchmark application performs a series of compressions at differing quality levels over a variety of images. The workload is taken from the behavior of someone seeking the best tradeoff between space and time for a variety of images. This benchmark can provide good opportunities to show off superscaler integer capabilities.

134.perl

134.perl is an interpreter for the Perl language. This version of Perl has had much of the "UNIXisms" removed (such as /etc/passwd lookups, and setuid behavior) to simplify benchmarking under various operating systems. For benchmarking, Perl is fed scripts that perform some basic math calculations and word lookups in associative arrays. As much as 10% of the time can be spent in routines commonly found in libc.a: malloc, free, memcpy, etc.

147.vortex

The benchmark 147.vortex is taken from a full object oriented database program called VORTEx. (VORTEx stands for "Virtual Object Runtime EXpository.") The benchmark builds and manipulates three separate, but inter-related databases built from definitions in a text-file schema. The workload of VORTEx has been modeled after common object-oriented database benchmarks with enhancements to increase the variation in the mix of transactions. This, along with 126.gcc, is one of the largest benchmarks in the suite. 147.vortex has shown itself to be sensitive to how well the system's TLB handler works.

CFP95 (all FORTRAN programs):

101.tomcatv

101.tomcatv is a highly vectorizable program for the generation of two-dimensional boundary-fitted coordinate systems around general geometric domains such as airfoils, cars, etc. It is based on the method introduced in 1974 which uses two Poisson equations to produce a mesh which adapts to the physical region of interest. The transformed non-linear equations are replaced by a finite difference approximation, and the resulting system is solved using successive line overrelaxation. The original source was part of Prof. W. Gentzsch's benchmark suite. The input set has been changed and increased over the SPEC CPU92 version. This is one of the benchmarks most sensitive to the speed of the memory system.

102.swim

102.swim solves the system of shallow water equations using finite difference approximations. For SPEC CPU95, the matrix has been greatly expanded to 513x513 and several numerical instabilities have been settled. This benchmark is also sensitive to the speed of the memory system. It can be parallelized well.

103.su2cor

103.su2cor is a Monte-Carlo method applied to the computation of masses of elementary particles in the framework of the Quark-Gluon theory. It can be parallelized reasonably well.

104.hydro2d

104.hydro2d solves hydrodynamical Navier Stokes equations to compute galactical jets. It can be parallelized reasonably well.

107.mgrid

107.mgrid is a Multi-grid solver in a 3D potential field. This should be one of the easiest to parallelize.

110.applu

110.applu solves parabolic/elliptic partial differential equations. In particular it uses a diagonalized approximate factorization method for inverting the Jacobian matrix.

125.turb3d

125.turb3d simulates isotropic, homogeneous turbulence in a cube. It includes a large 1D FFT.

141.apsi

141.apsi was derived from one of the Perfect Club benchmarks. It solves problems regarding temperature, wind, velocity, and distribution of pollutants. It is difficult to parallelize.

145.fpppp

145.fpppp was derived from the Gaussian series of quantum chemistry benchmarks (two electron integral derivative). The atoms in 145.fpppp are placed in a relatively compact region of space positioned in a graphite-like lattice. To accomplish this, code has been added that computes an atom's coordinates on an orthorhombic lattice with a Gaussian deviation. It would be very difficult to parallelize (the behavior of each atom is affected by the behavior of each one of the other atoms). It has a minimal cache footprint and is mostly a test of register allocation/performance. It may allow good exhibition of superscaler performance features.

146.wave5

146.wave5 solves Maxwell's equations and particle equations of motion on a Cartesian mesh with a variety of field and particle boundary conditions.

Some comparative characterization data for these benchmarks can be found at:

http://www.specbench.org/osg/cpu95/

For measuring results: each benchmark is compiled and run according to the rules for the desired metric (see Choosing Metrics). The run time in seconds in then entered into an appropriate formula to obtain either a SPECratio (for the speed measurements) or a SPECrate (for the rate measurements) for the given benchmark. All of the SPECrate/ratios are then combined via the geometric mean to obtain the overall metric. Full details are provided in the documentation that accompanies the benchmarks.

Issues

The following should be considered when using SPEC CPU95 benchmarks when comparing systems:

The benchmarks are performing a single task. The benchmarks do not stress the context switching ability of the machine or mimic lots of users on a system.

The benchmark workloads are fixed in size and do not scale with the power of the machine. This means that it may be theoretically possible to build a system big enough (i.e. - big enough first level cache) to bound the SPEC problem set.

All of the benchmarks within a given suite are written in the same language. On one hand this makes benchmarking easier (only compiler needs to be purchased), but on the other hand some areas may not be covered (floating point C programs for instance). SPEC is continually trying to find a balance between these factors and simplicity in benchmarking.

Futures

Creating benchmarks is harder than one might think. Just as technology continues to move forward, the methods for evaluating and comparing computer systems can not stay static. New uses for computer are developed and paradigms shift and benchmarks must be developed to reflect this.

SPEC is now beginning to address the possibilities of a SPEC CPU98 suite. Several questions need to be addressed as this proceeds:

- Can the same format as SPEC CPU95 be used for SPEC CPU98?

- Is it still useful to compare (processor, memory, compiler) components of a system with single compute intensive bounds?
- What application areas are developing that will be of interest in the 1998 time frame? Is it possible to obtain source code for these applications now?

SPEC will be addressing these questions and more over the next several months. And SPEC is willing to work with people in developing new benchmark suites or programs. Please feel free to contact SPEC through its Web page or by sending mail to: jwreilly@pro.specbench.org. Contacts with graduate students or research programs would be most helpful.

Conclusion

SPEC CPU95 is a step forward for comparitive compute-intensive benchmarking. SPEC has made efforts to improve the quality of the applications used, to improve the portability of the applications and to improve the ease of use and the portability of the tools. The challenge for SPEC now is education. Benchmarks provide a useful approximation to reality and SPEC needs to improve the guidance with these tools to help the users understand how and when to use the results.

More Information

Additional information about SPEC and its benchmark products can be found on SPEC's World Wide Web Site at http://www.specbench.org.

References:

Note: Much of this material has been published previously in the SPEC Newsletter or on the SPEC WWW Site: http://www.specbench.org.

[1] SPEC Newsletter, Sep-95, "About SPEC..."

[2] SPEC Newsletter, Fall-89, "SPEC Benchmark Suite: Designed For Today's Advanced Systems"

[3] Websters II Dictionary

[4] See Usenet comp.benchmarks Frequently Asked Questions list at: http://sacam.oren.ortn.edu/~dave/benchmark-faq.html

[5] http://www.specbench.org/osg/cpu95