Evolve or Die: Making SPEC's CPU Suite Relevant Today and Tomorrow

Jeff Reilly, Chair: SPEC CPU Subcommittee

IISWC-2006

October 27, 2006

With thanks to Alex Carlton, John Henning, Anirudha Rahatekar, Cloyce Spradling, the SPEC CPU Subcommittee and all those who contributed to SPEC's efforts

Agenda

- Benchmark Context
- We benchmarked… CPU2000
- We are benchmarking… CPU2006
- We will benchmark… ???

Please speak up if you have any questions or comments!

What Are Benchmarks?

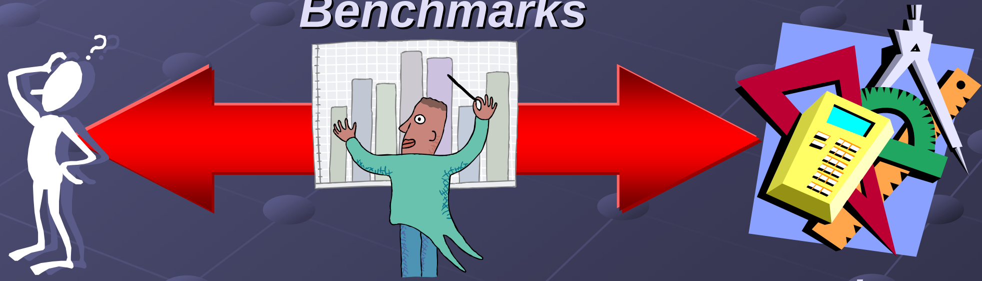
- A benchmark is a standard by which something can be measured or judged

- Examples Of Benchmarks:
  - Car Efficiency: miles per gallon
  - Sports Statistics: batting average
  - School: Grade point average

Benchmarks allow for evaluations or comparisons between two or more items

Why Use Benchmarks?

- Benchmarks lie between the extremes of “wild guess” and “actual environment”
- Benchmarks ideally, measure exactly want you want to evaluate but the following are issues...
  - Time
  - Money
  - Available data
  - Economy Of Scale



Benchmarks

Guess?

Actual Environment

“Benchmarks provide successive approximations to reality” – J. Mashey
This requires understanding both of the benchmark AND your needs!

A GOOD Benchmark Is...

- Relevant
- Recognized
- Simple
- Portable
- Scaleable

Not all benchmarks (including some popular ones) have all of these characteristics! Always assess this...

Source: Jim Gray

What Makes A Good Benchmark Go Bad?

- Technology improvements
  - Hardware tends to evolve faster than software; scalability issues
- Introduction of unanticipated technology
  - Rule issues; test may no longer be meaningful.
- Misuse
  - To many numbers, need for education
- Evolution of environment and usage models
  - Capture what is important to users today

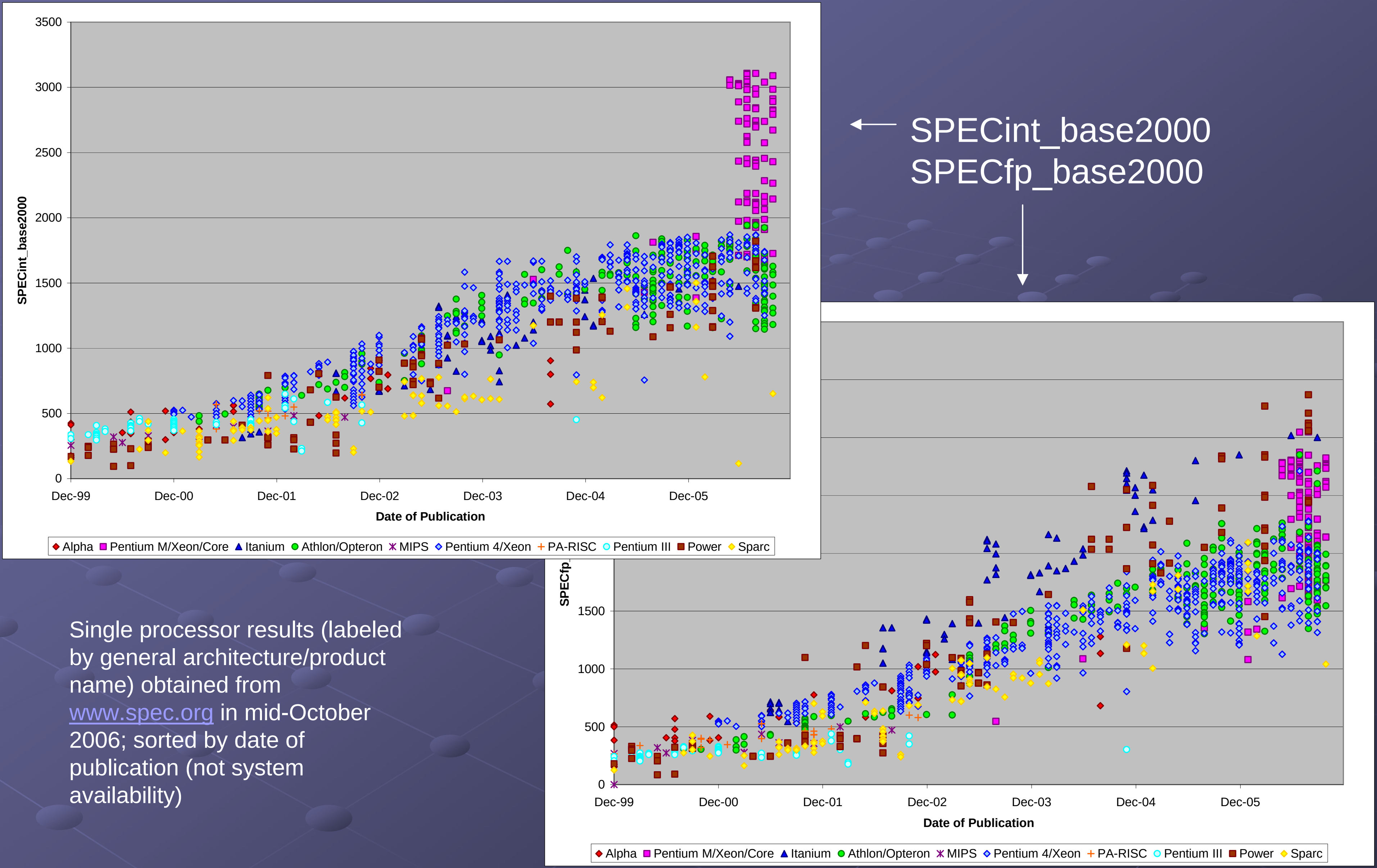
An often overlooked fact is that benchmarks need to evolve with TIME… Or Yesterday's 'good' benchmark may NOT be today's 'good' benchmark.

Where SPEC Was: CPU2000

- Introduced in December 1999
  - Expected to be retired in February 2007
- Continued using two suites of benchmarks:
  - CINT2000: 12 benchmarks; 11 in C, 1 in C++
  - CFP2000: 14 benchmarks; 4 in C, 10 in FORTRAN
- 6525 results published at www.spec.org as of October 27, 2006:
  - 1225: CINT2000
  - 1249: CFP2000
  - 2072: CINT2000 Rate
  - 1979: CFP2000 Rate

Meets the "good benchmark" definition of recognized and supported
Full details on http://www.spec.org

The CPU2000 Performance Landscape (speed metric)



← SPECint_base2000

SPECfp_base2000 ↓

Single processor results (labeled by general architecture/product name) obtained from www.spec.org in mid-October 2006; sorted by date of publication (not system availability)

SPECint_base2000: High: 3108; Low: 93.7; Delta: ~33x (~13-14 hrs to ~32+ minutes)

SPECfp_base2000: High: 3369; Low: 84.4; Delta: ~39x (~29-30 hrs to ~48-49 minutes)

What Drives SPEC’s Evolution

- Run-time
  - Want meaningful workloads; want meaningful measurement interval; possible conflict with cost of benchmarking
    - Run times are dropping below 30 seconds on the fastest machines
- Application type
  - Want workloads that are meaningful in a performance context
    - Code has been updated in the last ~7 years; new areas of interest exist.
- Application size
  - Want workloads that are taxing for today’s systems; enable remonstrate what is capable with coming systems
    - For example, system cache and memory sizes are increasing.
- Moving target
  - SPEC CPU is provided as source code; addresses compiler evolution.
    - 7 years is a long time for an “open book” test of compilers.

CPU2000 is starting to get “old”.

Some background logistics on the development process

- SPEC is an industry consortium (H/W, S/W, education, end-users) cooperating to develop benchmarks
  - CPU benchmarks are developed by the CPU Sub-committee of the Open Systems Group (OSG)
- Current members (as of mid-October 2006) of the CPU Subcommittee:
  - AMD, Apple, Dell, Fujitsu, Fujitsu-Siemens, HP, IBM, Intel, PGI, Qlogic, Sun
- Basic philosophy
  - To develop CPU benchmarks that provide a comparative measure of CPU, compiler and memory performance with relevant, real-world applications across the widest range of platforms
- Decision making is meant to be by consensus; voting sets directions and establishes final release.

SPEC development is a team effort.

What was sought for CPU2006?

- Same general paradigm as CPU2000 (speed and rate metrics, measure CPU(s)/memory/compiler.
- For a program to be included in a SPEC CPU benchmark:
  - Source code needs to be available to SPEC to use and distribute (e.g. – a search program has been used to work with authors) The author needs to provide 3 workloads – test, train and ref
  - The program needs to be portable across all OS/hardware combinations represented within SPEC (and attempts are made to cover others)
  - The program should spend 95% or more of its execution time in its source code
  - The program, once ported, must do the same amount of work on all systems
  - The program is expected not to significantly violate language standards
  - The program should have a reasonably flat profile and not be susceptible to huge improvements in performance due to compiler optimizations
  - The program must run without paging in a fixed amount of RAM on a 32-bit OS (256MB for CPU2000 and 1GB for CPU2006)
  - The program must run correctly in both 32-bit and 64-bit environments
  - The program should have a meaningful workload that takes a “noticable” amount of time on today’s fastest machines.
- CPU benchmarks have integer and floating point suites
  - a program with < 1% fp instructions is an integer program
  - a program with > 10% fp instructions is a floating point program
  - Items in between will be handled on a case by case basis

To some this is a surprising number of requirements to consider and work toward.

How did SPEC decide?

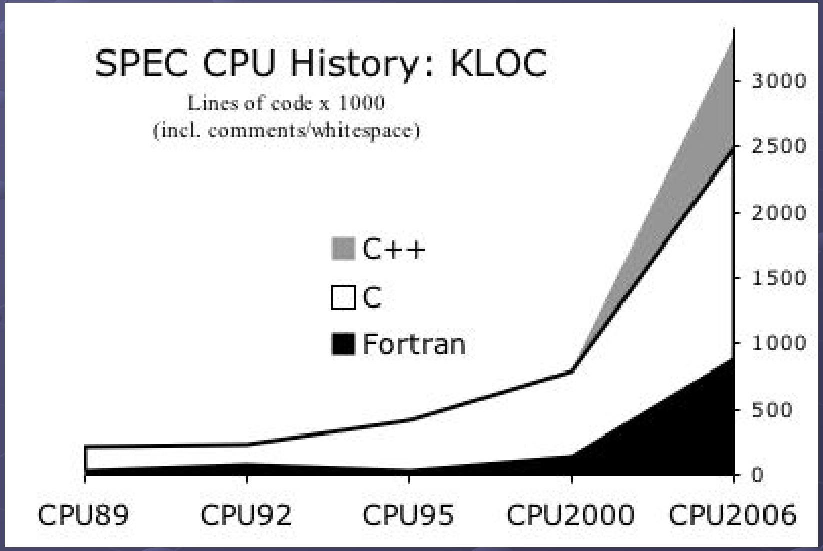
- Programs were brought to SPEC by authors/submitters and sought out by members of the SPEC CPU Subcommittee.
- Candidate programs are assigned a project lead within the SPEC CPU Subcommittee.
- Candidates are evaluated on the criteria listed on the previous page.
- Cost of benchmarks was and is a concern. How many is enough?
  - Things considered included:
    - Coverage from an application area view
    - Coverage from an architectural view
      - Clustering analysis employed (University of Texas (SPEC Associate))
    - Significant number of candidates to ensure one benchmark does not dominate.
    - Total run time

Ultimately, the SPEC CPU Subcommittee votes.

SPEC®, SPECint®, SPECfp®, SPECrate® are registered trademarks of the Standard Performance Evaluation Corporation (SPEC). Other brands may be claimed as the property of others.

What did SPEC end up with?

- SPEC CPU2006:
  - CINT2006: 12 benchmarks; 9 in C and 3 in C++.
  - CFP2006: 17 benchmarks; 3 in C, 4 in C++, 6 in FORTRAN and 4 in a mix of C and FORTRAN.
- Covers new application areas
- Provides much more code to consider
- 106 results published to date



SPECint_base2000 takes 5-6 hours on the fastest machine reported as of 10/27/2007. SPECfp_base2000 takes 9-10 hours on the fastest machine reported as of 10/27/2007. Reference machine takes ~12 days for both.

What did SPEC end up with?

CINT2006 (Integer Component of SPEC CPU2006):

Benchmark	Language	Application Area	Brief Description
400.perlbench	C	Programming Language	Derived from Perl V5.8.7. The workload includes SpamAssassin, MHonArc (an email indexer), and specdiff (SPEC's tool that checks benchmark outputs).
401.bzip2	C	Compression	Julian Seward's bzip2 version 1.0.3, modified to do most work in memory, rather than doing I/O.
403.gcc	C	C Compiler	Based on gcc Version 3.2, generates code for Opteron.
429.mcf	C	Combinatorial Optimization	Vehicle scheduling. Uses a network simplex algorithm (which is also used in commercial products) to schedule public transport.
445.gobmk	C	Artificial Intelligence: Go	Plays the game of Go, a simply described but deeply complex game.
456.hmmer	C	Search Gene Sequence	Protein sequence analysis using profile hidden Markov models (profile HMMs)
458.sjeng	C	Artificial Intelligence: chess	A highly-ranked chess program that also plays several chess variants.
462.libquantum	C	Physics / Quantum Computing	Simulates a quantum computer, running Shor's polynomial-time factorization algorithm.
464.h264ref	C	Video Compression	A reference implementation of H.264/AVC, encodes a videostream using 2 parameter sets. The H.264/AVC standard is expected to replace MPEG2
471.omnetpp	C++	Discrete Event Simulation	Uses the OMNet++ discrete event simulator to model a large Ethernet campus network.
473.astar	C++	Path-finding Algorithms	Pathfinding library for 2D maps, including the well known A* algorithm.
483.xalancbmk	C++	XML Processing	A modified version of Xalan-C++, which transforms XML documents to other document types.

More information available at http://www.spec.org

What did SPEC end up with?

CFP2006 (Floating Point Component of SPEC CPU2006):

Benchmark	Language	Application Area	Brief Description
410.bwaves	Fortran	Fluid Dynamics	Computes 3D transonic transient laminar viscous flow.
416.gamess	Fortran	Quantum Chemistry.	Gamess implements a wide range of quantum chemical computations. For the SPEC workload, self-consistent field calculations are performed using the Restricted Hartree Fock method, Restricted open-shell Hartree-Fock, and Multi-Configuration Self-Consistent Field
433.milc	C	Physics / Quantum Chromodynamics	A gauge field generating program for lattice gauge theory programs with dynamical quarks.
434.zeusmp	Fortran	Physics / CFD	ZEUS-MP is a computational fluid dynamics code developed at the Laboratory for Computational Astrophysics (NCSA, University of Illinois at Urbana-Champaign) for the simulation of astrophysical phenomena.
435.gromacs	C, Fortran	Biochemistry / Molecular Dynamics	Molecular dynamics, i.e. simulate Newtonian equations of motion for hundreds to millions of particles. The test case simulates protein Lysozyme in a solution.
436.cactusADM	C, Fortran	Physics / General Relativity	Solves the Einstein evolution equations using a staggered-leapfrog numerical method
437.leslie3d	Fortran	Fluid Dynamics	Computational Fluid Dynamics (CFD) using Large-Eddy Simulations with Linear-Eddy Model in 3D. Uses the MacCormack Predictor-Corrector time integration scheme.
444.namd	C++	Biology / Molecular Dynamics	Simulates large biomolecular systems. The test case has 92,224 atoms of apolipoprotein A-I.
447.dealII	C++	Finite Element Analysis	deal.II is a C++ program library targeted at adaptive finite elements and error estimation. The testcase solves a Helmholtz-type equation with non-constant coefficients.
450.soplex	C++	Linear Programming, Optimization	Solves a linear program using a simplex algorithm and sparse linear algebra. Test cases include railroad planning and military airlift models.
453.povray	C++	Image Ray-tracing	Image rendering. The testcase is a 1280x1024 anti-aliased image of a landscape with some abstract objects with textures using a Perlin noise function.
454.calculix	C, Fortran	Structural Mechanics	Finite element code for linear and nonlinear 3D structural applications. Uses the SPOOLES solver library.
459.GemsFDTD	Fortran	Computational Electromagnetics	Solves the Maxwell equations in 3D using the finite-difference time-domain (FDTD) method.
465.tonto	Fortran	Quantum Chemistry	An open source quantum chemistry package, using an object-oriented design in Fortran 95. The test case places a constraint on a molecular Hartree-Fock wavefunction calculation to better match experimental X-ray diffraction data.
470.lbm	C	Fluid Dynamics	Implements the "Lattice-Boltzmann Method" to simulate incompressible fluids in 3D
481.wrf	C, Fortran	Weather	Weather modeling from scales of meters to thousands of kilometers. The test case is from a 30km area over 2 days.
482.sphinx3	C	Speech recognition	A widely-known speech recognition system from Carnegie Mellon University

More information available at http://www.spec.org

What did SPEC Change?

- No feedback-directed optimization in baseline
- No limit on number of switches at baseline
- CPU2006 updated expectations for language standards
- CPU2006 included mixed-language benchmarks (C and Fortran) which will be considered as Fortran benchmarks by the tools
- Clearer/more rigorous rules.
  - CPU2006 will impose a requirement for all benchmarks to validate the test, train and reference workloads during an official run whereas CPU2000 only requires validation of ref.
  - Baseline allows switches for alignment on natural boundaries
- The reporting format was updated.
  - Among other things, information about compiler flags is more easily accessed.

Expectation: SPEC CPU will continue to be of interest for compute intensive performance comparisons.

What does SPEC do next?

?

Thoughts?
Now is the time to start kick starting the evolution.

What does SPEC do next?

- Address any issues that arise with CPU2006; provide new revisions as expected.
- Brainstorming has started
  - A need is still seen for providing a means for comparing compute intensive performance
  - Likely consideration for the multi-core evolution
  - Expect finding code to become more difficult.
- SPEC to start planning in earnest in January 2007.

SPEC welcomes input!

Thank you!

Email: info@spec.org; jwreilly@mipos2.intel.com