SPEC CPU2006: An Overview and Round Table Discussion

Jeff Reilly
SPEC CPU Committee Chair
Purpose and agenda

Purpose: Provide an overview of SPEC CPU2006 and have a round table forum/question and answer session.

Agenda:
- Review: What are benchmarks and what makes a good benchmark?
- A brief overview of SPEC CPU2006
- Question and answers/discussion
Background: What are benchmarks?

A standard by which something can be measured or judged.

Specific to computers: A program (or suite of programs) used to assess the performance characteristics of one or more computer systems.

Examples:

• Fuel efficiency/kilometers per gallon
• Grades in school
• SPEC CPU2006

Benchmarks allow for comparisons between two or more items.
Background: Why use benchmarks?

Benchmarks provide information somewhere between “no information/guess” and “actual environment”.

In a perfect world, you would measure exactly want you want to evaluate but the following are issues...

- Time
- Money
- Available data
- Economy Of Scale

“Benchmarks provide successive approximations to reality”
This requires understanding both of the benchmark AND your needs!
A good benchmark is...

- Relevant
- Reproducible
- Fair
- Usable
- Well-defined/verifiable
- Recognized
- Simple
- Portable/Scalable

Not all benchmarks (including some popular ones) have all of these characteristics! Always considered this list...
What were SPEC’s goals with SPEC CPU 2006?

“SPEC designed this suite to provide a comparative measure of compute-intensive performance across the widest practical range of hardware using workloads developed from real user applications.”

Through the SPEC website, provide a resource to the performance community with reviewed results and other content about the benchmark.
What is SPEC CPU2006?

A benchmark suite, composed of real applications, for comparing the compute capabilities of a given system.

Emphasizes: Processor(s), memory, compiler

<table>
<thead>
<tr>
<th>8 Metrics</th>
<th>Integer/ Floating Point</th>
<th>Speed (For each benchmark, time to complete one invocation)</th>
<th>Rate (Throughput for running user selected number of concurrent copies)</th>
<th>Baseline (All programs of a given language compiled the same way; no FDO)</th>
<th>Peak (Each program may be compiled differently)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECint2006</td>
<td>Integer</td>
<td>Speed</td>
<td></td>
<td>Peak</td>
<td></td>
</tr>
<tr>
<td>SPECint_base2006</td>
<td>Integer</td>
<td>Speed</td>
<td></td>
<td>Baseline</td>
<td></td>
</tr>
<tr>
<td>SPECint_rate2006</td>
<td>Integer</td>
<td>Rate</td>
<td></td>
<td>Peak</td>
<td></td>
</tr>
<tr>
<td>SPECint_rate_base2006</td>
<td>Integer</td>
<td>Rate</td>
<td></td>
<td>Baseline</td>
<td></td>
</tr>
<tr>
<td>SPECfp2006</td>
<td>Floating point</td>
<td>Speed</td>
<td></td>
<td>Peak</td>
<td></td>
</tr>
<tr>
<td>SPECfp_base2006</td>
<td>Floating point</td>
<td>Speed</td>
<td></td>
<td>Baseline</td>
<td></td>
</tr>
<tr>
<td>SPECfp_rate2006</td>
<td>Floating point</td>
<td>Rate</td>
<td></td>
<td>Peak</td>
<td></td>
</tr>
<tr>
<td>SPECfp_rate_base2006</td>
<td>Floating point</td>
<td>Rate</td>
<td></td>
<td>Baseline</td>
<td></td>
</tr>
</tbody>
</table>
What is 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.  
Based on real applications. Full details can be seen at www.spec.org.
## Integer Workloads

### CINT2006 (Integer Component of SPEC CPU2006):

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

**CFP2006 (Floating Point Component of SPEC CPU2006):**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Language</th>
<th>Application Area</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>410.bwaves</td>
<td>Fortran</td>
<td>Fluid Dynamics</td>
<td>Computes 3D transonic transient laminar viscous flow.</td>
</tr>
<tr>
<td>416.gamess</td>
<td>Fortran</td>
<td>Quantum Chemistry</td>
<td>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.</td>
</tr>
<tr>
<td>433.milc</td>
<td>C</td>
<td>Physics / Quantum Chromodynamics</td>
<td>A gauge field generating program for lattice gauge theory programs with dynamical quarks.</td>
</tr>
<tr>
<td>434.zeusmp</td>
<td>Fortran</td>
<td>Physics / CFD</td>
<td>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.</td>
</tr>
<tr>
<td>435.gromacs</td>
<td>C, Fortran</td>
<td>Biochemistry / Molecular Dynamics</td>
<td>Molecular dynamics, i.e. simulate Newtonian equations of motion for hundreds to millions of particles. The test case simulates protein Lysozyme in a solution.</td>
</tr>
<tr>
<td>437.leslie3d</td>
<td>Fortran</td>
<td>Fluid Dynamics</td>
<td>Computational Fluid Dynamics (CFD) using Large-Eddy Simulations with Linear-Eddy Model in 3D. Uses the MacCormack Predictor-Corrector time integration scheme.</td>
</tr>
<tr>
<td>444.namd</td>
<td>C++</td>
<td>Biology / Molecular Dynamics</td>
<td>Simulates large biomolecular systems. The test case has 92,224 atoms of apolipoprotein A-I.</td>
</tr>
<tr>
<td>447.dealII</td>
<td>C++</td>
<td>Finite Element Analysis</td>
<td>deal.II is a C++ program library targeted at adaptive finite elements and error estimation. The test case solves a Helmholtz-type equation with non-constant coefficients.</td>
</tr>
<tr>
<td>450.soplex</td>
<td>C++</td>
<td>Linear Programming, Optimization</td>
<td>Solves a linear program using a simplex algorithm and sparse linear algebra. Test cases include railroad planning and military airlift models.</td>
</tr>
<tr>
<td>453.povray</td>
<td>C++</td>
<td>Image Ray-tracing</td>
<td>Image rendering. The test case is a 1280x1024 anti-aliased image of a landscape with some abstract objects with textures using a Perlin noise function.</td>
</tr>
<tr>
<td>454.calculix</td>
<td>C, Fortran</td>
<td>Structural Mechanics</td>
<td>Finite element code for linear and nonlinear 3D structural applications. Uses the SPOOLES solver library.</td>
</tr>
<tr>
<td>459.GemsFDTD</td>
<td>Fortran</td>
<td>Computational Electromagnetics</td>
<td>Solves the Maxwell equations in 3D using the finite-difference time-domain (FDTD) method.</td>
</tr>
<tr>
<td>465.tonto</td>
<td>Fortran</td>
<td>Quantum Chemistry</td>
<td>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.</td>
</tr>
<tr>
<td>470.lbm</td>
<td>C</td>
<td>Fluid Dynamics</td>
<td>Implements the &quot;Lattice-Boltzmann Method&quot; to simulate incompressible fluids in 3D.</td>
</tr>
<tr>
<td>481.wrf</td>
<td>C, Fortran</td>
<td>Weather</td>
<td>Weather modeling from scales of meters to thousands of kilometers. The test case is from a 30km area over 2 days.</td>
</tr>
<tr>
<td>482.sphinx3</td>
<td>C</td>
<td>Speech recognition</td>
<td>A widely-known speech recognition system from Carnegie Mellon University.</td>
</tr>
</tbody>
</table>
How can you use SPEC CPU2006?

Run it on your own machine.

Order a license from SPEC.

Compile and run on your own equipment.

If you wish, submit to SPEC for review and publication.
(https://www.spec.org/osg/submitting_results.html)

Ask your platform/compiler supplier to provide results.

Use/compare the results on the SPEC website.

The website has a database of results (over 30,000)
How to search for CPU2006 results

Go to www.spec.org and in the top menu, click on “results” and select CPU2006

From there you can look at results by time or click on the link for “CPU2006 Search Form”

CPU2006 Results -- Query

This configuration offers access to summary information across all CPU2006 results.

Simple Request

Fetch just the summary information for all results.

• Optional: Return only those results where
  - Hardware Vendor Matches

Execute Simple Fetch
### Example of CPU2006 result

**Operating System Notes**

- `ulimit = 131072` (shells); increases stack
- `/etc/system parameters`
  - `timeout = false`:
    - Controls how many seconds elapse between runs of the page fault daemon. If timeout is `false`, the contents of pages older than the listed number of seconds to be written by faultless.

**Platform Notes**

- Default RJCC settings were used.

**Base Compiler Invocation**

```bash
CC = -x86 -mcpu=athlon -march=athlon64
```

**Base Optimization Flags**

```bash
-C = -O3 -fomit-frame-pointer
```

**Base Portability Flags**

```bash
-C = -D SPEC_CPU_1044 -D SPEC_CPU_386_444
```

**Base Other Flags**

- `-D SPEC_CPU_386_444`
- `-D SPEC_CPU_95 Barton`
- `-D SPEC_CPU_1044`
- `-D SPEC_CPU_386_444`
- `-D SPEC_CPU_95 Barton`
- `-D SPEC_CPU_1044`
- `-D SPEC_CPU_386_444`
- `-D SPEC_CPU_95 Barton`
- `-D SPEC_CPU_1044`
- `-D SPEC_CPU_386_444`
- `-D SPEC_CPU_95 Barton`
- `-D SPEC_CPU_1044`

**Peak Compiler Invocation**

Same as Base Compiler Invocation
Thank you!

info@spec.org
www.spec.org