White Paper
FUJITSU x86 Servers & Workstations
Benchmark overview SPECcpu2006

This document presents the benchmark SPECcpu2006 which was developed and published by the Standard Performance Evaluation Corporation (SPEC).

This benchmark, in which processor and memory performance are measured, is a benchmark that is generally accepted by the industry. As a result of the benchmark concept realized in SPECcpu2006 it is possible to make across-the-board manufacturer comparisons.

Version
1.3
2015-09-16

Contents

Document history

Version 1.0

Initial version

Version 1.1

Layout changed

Version 1.2

Layout changed

Version 1.3

Paragraph inserted about the rounding of measurement results to three significant digits

SPECcpu2006 – An overview

The “Standard Performance Evaluation Corporation“ or briefly SPEC is an organization specialized in the development and issue of standardized benchmarks to evaluate the performance of computer systems. Members of the SPEC consortium are the leading companies of the computer industry such as Fujitsu. One of the benchmarks issued by SPEC and the successor to CPU2000 is CPU2006. The purpose of this document is to provide an overview of the importance of this benchmark and the meaningfulness of its results.

Objective of the benchmark

When comparing systems, in principle the individual customer application would be the measure of all things. However, due to shortage of time or for complexity or other reasons you often do not have the opportunity to test a variety of possible systems with a customer application in order to find out which is the most suitable one. Benchmarks have been developed to nevertheless provide customers with objective criteria when choosing their system. These are standardized tools, measuring the speed of individual system components or the speed of the entire system.

CPU2006 is a benchmark developed to allow a performance comparison between systems in case of compute-bound activities. Strictly speaking, CPU2006 is not a single benchmark but consists of two benchmark groups:

- CINT2006 for measurement of the system performance in case of integer operations
- CFP2006 for measurement of the system performance in case of floating-point operations

The “C“ in the name of the benchmark groups stands for “component“. Therefore, this is a component benchmark in contrast to a system benchmark. As a compute-bound benchmark, it concentrates besides on the processor also on the memory architecture and the compiler. Important: The last-named components should not be ignored. Performance is not only a matter of the processor’s clock pulse frequency. Also important: Other system components, i.e. I/O, graphics or network, are not subject to a load test. If the performance bottleneck of an application is due to one of these components, the CPU2006 benchmark does not allow to recognize this. Therefore, it depends on the load profile of an application whether or not CPU2006 is the right benchmark in the individual case.

Components of the benchmark

The benchmark group CINT2006 consists of 12 applications written in C and C++.

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

The benchmark group CFP2006 consists of 17 applications written in C, C++ and FORTRAN:

Benchmark	Application Area
410.bwaves	Fluid Dynamics Computes 3D transonic transient laminar viscous flow.
416.gamess	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	Physics / Quantum Chromo dynamics A gauge field generating program for lattice gauge theory programs with dynamical quarks.
434.zeusmp	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	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	Physics / General Relativity Solves the Einstein evolution equations using a staggered-leapfrog numerical method.
437.leslie3d	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	Biology / Molecular Dynamics Simulates large biomolecular systems. The test case has 92,224 atoms of apolipoprotein A-I.
447.dealII	Finite Element Analysis 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.
450.soplex	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	Image Ray-tracing Image rendering. The testcase is a 400x400 anti-aliased image of a landscape with some abstract objects with textures using a Perlin noise function.
454.calculix	Structural Mechanics Finite element code for linear and nonlinear 3D structural applications. Uses the SPOOLES solver library.
459.GemsFDTD	Computational Electromagnetics Solves the Maxwell equations in 3D using the finite-difference time-domain (FDTD) method.
465.tonto	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 wave function calculation to better match experimental X-ray diffraction data.
470.lbm	Fluid Dynamics Implements the "Lattice-Boltzmann Method" to simulate incompressible fluids in 3D.
481.wrf	Weather Weather modeling from scales of meters to thousands of kilometers. The test case is from a 30km area over 2 days.
482.sphinx3	Speech recognition A widely-known speech recognition system from Carnegie Mellon University.

All individual benchmarks share that file-I/O only occurs when input files are read. Computing uses only cache and RAM. However, this requires that the system has a minimum memory capacity of (1 * number of processor cores) GB on 32-bit systems and (2 * number of processor cores) GB on 64-bit systems, not including space for the operating system and other non-SPECcpu2006-specific tasks.

Measurement results and their interpretation

CPU2006 can carry out eight different measurements that are distinguished in terms of means and methodology:

Benchmark	Number of single benchmarks	Arithmetics	Type	Compiler optimization	Measurement result	Application
SPECint2006	12	integer	peak	aggressive	Speed	single-threaded
SPECint_base2006	12	integer	base	conservative		
SPECint_rate2006	12	integer	peak	aggressive	Throughput	multi-threaded
SPECint_rate_base2006	12	integer	base	conservative		
SPECfp2006	17	floating point	peak	aggressive	Speed	single-threaded
SPECfp_base2006	17	floating point	base	conservative		
SPECfp_rate2006	17	floating point	peak	aggressive	Throughput	multi-threaded
SPECfp_rate_base2006	17	floating point	base	conservative		

The benchmarks are provided by SPEC as source code to allow the executability of the benchmarks on various hardware models. Therefore, they have to be compiled first. However, the compilation of programs is not a standardized procedure. There are a couple of optional optimization possibilities that may be used by a programmer and maybe ignored by another one. Considering this problem SPEC allows both: Measuring with conservative (base) optimization where strict guidelines exist – essentially the use of the same optimization flags in identical sequence for all individual benchmarks - as well as measuring with aggressive (peak) optimization where the guidelines are more generous, so each single benchmark can be optimized individually. The first measurement is mandatory, the second one is optional only.

SPEC provides two different methods of performance measuring: One method (speed) consists in measuring how much time is required to execute a single task. The second method (rate) measures the throughput, which means how much time is required to execute a pre-defined number of multiple tasks in parallel.

All benchmark results represent the geometric mean of normalized ratios determined for the individual benchmarks. “Normalized“ means measuring how fast the test system runs in comparison to a reference system. SPEC chose the Sun “Ultra Enterprise 2” system with a 296 MHz UltraSPARC II processor as the reference system. Each benchmark was executed on this system and the reference times were determined. For every base benchmark a value of “1“ was determined for the reference system. The result of a test system means, that it is able to do the same work as an equivalent number of reference systems in the same time as the reference system needs. For example, if a test system has a SPECint_base2006 value of 2; this means that it has executed this benchmark at about twice as fast as the reference system. Or, if a test system has a SPECfp_rate_base2006 value of 7.8; this means that it has executed this benchmark at about 7.8 / [number of users] as fast as the reference system and that the test system has executed [number of users] users in parallel. In general, if for two systems the number of users is the same, the system with the higher result is faster. If the number of users is different, the system with the higher result does more. The reason for it may be that it is faster or that it can do more in parallel.

The geometric mean is used to calculate the overall benchmark result from the single benchmark results. The effect of this method is a weighting in favor of the lower single results compared with the arithmetical-mean method. The intention behind this is to prevent any attempts to optimize compilers for an individual benchmark and thus to give this benchmark a disproportionately high weight. Moreover, this causes a shift towards the component that has remained unimproved. Better processors do not have such a strong impact on those single benchmarks that require many RAM accesses relative to cache accesses. Their worse results compared to other single benchmarks are emphasized by the use of the geometric mean.

Since minor fluctuations in performance are completely natural between measuring runs, all SPEC measurement results are deliberately rounded to three significant digits. For example, both a measured SPECint_rate value of 1234.999 and a measured SPECint_rate value of 1225.000 result in a rounded SPECint_rate value of 1230. The result files in html, pdf, ps and txt format contain such rounded values. Even requests with SPEC for results (e.g. greater than 1230) always refer to rounded values.

The interpretation of benchmark results depends on whether a complete system or a single component (processor, memory or compiler) is to be compared with a competitive product. If the focus is on a single component it has to be considered that the other components are identical as otherwise values cannot be clearly assigned to single components. Therefore, the system configuration is documented with great precision in all publications.

Usually we submit our SPECcpu2006 measurements for publication at SPEC. But not all of our results appear at SPEC's web sites. As we archive the log data for all measurements, we are able to prove the correct realization of the measurements – even the not submitted ones - any time.

Literature

PRIMERGY Servers

http://www.fujitsu.com/fts/products/computing/servers/primergy/

PRIMEQUEST Servers

http://www.fujitsu.com/fts/products/computing/servers/mission-critical/

CELSIUS Workstations

http://www.fujitsu.com/fts/products/computing/pc/workstations/

PRIMERGY & PRIMEQUEST Performance

http://www.fujitsu.com/fts/x86-server-benchmarks

Benchmark descriptions

http://www.fujitsu.com/fts/products/computing/servers/primergy/benchmarks/benchmark-descriptions.html

This White Paper:

- http://docs.ts.fujitsu.com/dl.aspx?id=1a427c16-12bf-41b0-9ca3-4cc360ef14ce
- http://docs.ts.fujitsu.com/dl.aspx?id=04351fd2-8a69-42a3-ba1c-4342dcc89b89
- http://docs.ts.fujitsu.com/dl.aspx?id=00b0bf10-8f75-435f-bb9b-3eceb5ce0157

SPECcpu2006

http://www.spec.org/osg/cpu2006

Contact

FUJITSU

Website: http://www.fujitsu.com/

PRIMERGY & PRIMEQUEST Product Marketing

mailto:Primergy-PM@ts.fujitsu.com

PRIMERGY Performance and Benchmarks

mailto:primergy.benchmark@ts.fujitsu.com