

Benchmark:

Program used to evaluate performance.

Uses

- Guide computer design.
- Guide purchasing decisions.
- Marketing tool.

Using Benchmarks to Guide Computer Design

Measure overall performance.

Determine characteristics of programs.

E.g. /, frequency of floating-point operations.

Determine effect of design options.

Choosing Benchmark Programs

Important Question: *We need some benchmarks. Which programs should we choose?*

My Answer:

Fedora Linux, Emacs, Firefox, the GNU development toolchain, the CUDA toolkit, Inkscape, Okular, Amarok, Digikam, Gimp, Zoom.

Hello, Intel. Please use the programs above to develop your next-generation processor. Thank you.

Unrealistic (in most cases) answer:

The exact set of programs customer will run.

Unrealistic because different customers may run different programs.

Therefore, choice of benchmarks will rarely make everyone happy.

Benchmark Types

A benchmark program can be...

... an ordinary program that someone runs (*e.g.*, Firefox) ...

... or a program specifically written to test a system (*e.g.*, our `pi` program)...

... or something in between.

The following types describe where a benchmark falls in this range.

Benchmark Types

Real Programs:

Programs chosen using surveys, for example.

Example: Photoshop (Image editing program.)

- + Measured performance improvements apply to customer.
- Large programs hard to run on simulator. (Before system built.)

Kernels:

Use part of program responsible for most execution time.

Example: Photoshop code for shrinking an image.

- + Easier to study.
- Not all program have small kernels.

Benchmark Types

Microbenchmarks:

Code written to test a specific feature of a system.

Example: Measure maximum number of FP divisions per second.

- + Useful for tuning specific features during implementation development.
- One might become too fixated on a narrow feature.

Toy Benchmarks:

Programs written casually, without insuring that they measure something useful.

Example: The pi program used in class.

- + Easier to write.
- Not realistic.

Benchmark Types

Commonly Used Benchmark Types

Overall performance: real programs

Test specific features: microbenchmarks.

Benchmark Suites

Benchmark Suite:

A named set of programs used to evaluate a system.

Typically:

- Developed and managed by a publication or non-profit organization.
E.g. /, Standard Performance Evaluation Corp., PC Magazine.
- Tests clearly delineated aspects of system.
E.g. /, CPU, graphics, I/O, application.
- Specifies a set of programs and inputs for those programs.
- Specifies reporting requirements for results.

What Suites Might Measure

- Application Performance
E.g., productivity (office) applications, database programs.
Usually tests entire system.
- CPU and Memory Performance
Ignores effect of I/O.
- Graphics Performance
- Database Transaction Throughput
Perhaps a datacenter workload.

SPEC CPU Benchmark Suites

Introduction

Measure CPU and memory performance on *integer* and *FP* programs.

Respected measure of CPU performance.

Managed by *Standard Performance Evaluation Corporation* ...

... a non-profit organization funded by computer companies and other interested parties.

SPEC CPU Updated Every Few Years

Latest Version (AOTW) 2017, previous version 2006.

Each SPEC CPU suite (*e.g.*, SPEC 2017)...

... consists of two sets of programs, *SPECint* and *SPECfp* ...

... each can be prepared and run multiple ways.

SPEC CPU Suite Goals

Measure **CPU** and **memory** system.

Avoid benchmarks making lots of disk I/O, etc.

Measure **potential of newest** implementations and ISAs.

Tester compiles benchmark using own tools.

Trustworthiness of **Suite**.

Suite developed by competitors, and other interested parties.

Trustworthiness of **Results**.

Easy for anyone to duplicate test results, so erroneous results quickly exposed.

SPEC CPU2017 Benchmark Programs

List of programs at: <https://www.spec.org/cpu2017/Docs/overview.html#benchmarks>.

Typical Integer Programs (SPECint*2017)

- Old-School AI: (deepsjeng [chess], leela [go]).
- Compression. (xz).
- Programming: (gcc, perl).

Typical Floating-Point Programs (SPECfp*2017)

- Finite-difference scientific computation. (cactuBSSN)
- Weather forecasting (wrf).

SPEC CPU2017 Suites and Measures

The Three SPEC CPU2017 “Axes”

Integer v. Floating Point (int v. fp):

Refers to two suites of programs.

Execution Time v. Throughput (speed v. rate):

Execution Time: One program running, measure its run time.

Throughput: Multiple copies of same program running, measure N/t .

Untuned v. Tuned (base v. peak):

Untuned (Base): Prepared by skilled and conscientious programmer.

Tuned (Peak): Prepared by hyper-motivated expert.

Suite of **integer programs** run to determine:

- *SPECspeed2017_int_peak*, execution time of tuned code.
- *SPECspeed2017_int_base*, execution time of untuned code.
- *SPECrate2017_int_peak*, throughput of tuned code.
- *SPECrate2017_int_base*, throughput of untuned code.

Suite of **floating programs** run to determine:

- *SPECspeed2017_fp_peak*, execution time of tuned code.
- *SPECspeed2017_fp_base*, execution time of untuned code.
- *SPECrate2017_fp_peak*, throughput of tuned code.
- *SPECrate2017_fp_base*, throughput of untuned code.

Integer v. Floating Point

SPECcpu programs divided into two sets, *integer* and *floating-point*.

Neither set is affected much by:

Disk access.

Other I/O, including graphics.

Floating-Point Programs

Have many floating point operations. (Of course.)

Have loops that iterate for many iterations.

Have fewer branch instructions.

SPEC CPU Benchmark Suites » SPEC Testing Procedure

SPEC Testing Procedure

Defined by *Run & Reporting Rules*.

See: <https://www.spec.org/cpu2017/Docs/runrules.html>

SPEC benchmarks are run by a *tester* (not SPEC).

Testing Steps

Get:

System Under Test (SUT):

The computer on which benchmarks are to be run.

A copy of the SPECcpu benchmark suite.

Compilers and other build tools for your system.

Prepare a *config file*:

Name of system, build tools, etc.

Location of compiler.

Portability switches.

Optimization switches.

Run the SPEC script:

Script will..

Compile benchmarks, profile, compile again.

Run benchmarks three times, verify outputs.

Generate reports.

Evaluate results:

If not satisfied

Try different optimization switches.

Substitute different compilers, libraries, etc.

Convince customers that for them SPECcpu results are irrelevant.