Benchmarks

02-1

Benchmark:
Program used to evaluate performance.

Uses

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

02-2

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

02-3

Important: Choice of programs for evaluation.

Optimal but unrealistic:

The exact set of programs customer will run.

Problem: computers used for different applications.

Therefore, must model typical users' workload.

Benchmark Types

02-4

Benchmark Classifications

Based on how benchmark is to be used.

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.

02-5

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 get too fixated on 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.

Commonly Used Benchmark Categories

Overall performance: real programs

Test specific features: microbenchmarks.

Benchmark Suites

02-6

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.

02-7

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

02-8

Example, SPEC CPU2006 Suites

Measures 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.

Uses common programs such as perl, gcc, gzip.

Requires that results on each program be reported.

Programs compiled with publicly available compilers and libraries.

Programs compiled with and without expert tuning.

02-9

SPEC CPU2006 Suites and Measures

Suite of integer programs run to determine:

- SPECint2006, execution time of tuned code.
- SPECint_base2006, execution time of untuned code.
- SPECint_rate2006, throughput of tuned code.
- SPECint_rate_base2006, throughput of untuned code.

Suite of floating programs run to determine:

- SPECfp2006, execution time of tuned code.
- SPECfp_base2006, execution time of untuned code.
- SPECfp_rate2006, throughput of tuned code.
- SPECfp_rate_rate2006, throughput of untuned code.

SPECcpu Benchmarks

02-10

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.

02-11

SPEC Testing Procedure

Defined by *Run & Reporting Rules*.

Carried out by tester (not SPEC).

Given:

A computer to test.

A copy the SPEC benchmarks.

Software to compile.

02-12

Test Procedure

Get:

System Under Test (SUT):
The computer on which benchmarks are to be run.

A copy of the SPECcpu benchmark suite.

Prepare a config file:

Name of system, build tools, etc.

Location of compiler.

Portability switches.

Optimization switches.

02-13

Run 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.