Lecture: Metrics to Evaluate Performance

- Topics: Benchmark suites, Performance equation,
  Summarizing performance with AM, GM, HM

  - Video 1: Using AM as a performance summary
  - Video 2: GM, Performance Equation
  - Video 3: AM vs. HM vs. GM

Measuring Performance

- Two primary metrics: wall clock time (response time for a program) and throughput (jobs performed in unit time)

- To optimize throughput, must ensure that there is minimal waste of resources

Benchmark Suites

- Performance is measured with benchmark suites: a collection of programs that are likely relevant to the user
  - SPEC CPU 2006: cpu-oriented programs (for desktops)
  - SPECweb, TPC: throughput-oriented (for servers)
  - EEMBC: for embedded processors/workloads

Summarizing Performance

- Consider 25 programs from a benchmark set – how do we capture the behavior of all 25 programs with a single number?

	P1	P2	P3
Sys-A	10	8	25
Sys-B	12	9	20
Sys-C	8	8	30

  - Sum of execution times (AM)
  - Sum of weighted execution times (AM)
  - Geometric mean of execution times (GM)

Sum of Weighted Exec Times – Example

- We fixed a reference machine X and ran 4 programs A, B, C, D on it such that each program ran for 1 second

- The exact same workload (the four programs execute the same number of instructions that they did on machine X) is run on a new machine Y and the execution times for each program are 0.8, 1.1, 0.5, 2

- With AM of normalized execution times, we can conclude that Y is 1.1 times slower than X – perhaps, not for all workloads, but definitely for one specific workload (where all programs run on the ref-machine for an equal #cycles)

Summarizing Performance

- Consider 25 programs from a benchmark set – how do we capture the behavior of all 25 programs with a single number?

	P1	P2	P3
Sys-A	10	8	25
Sys-B	12	9	20
Sys-C	8	8	30

  - Sum of execution times (AM)
  - Sum of weighted execution times (AM)
  - Geometric mean of execution times (GM) (may find inconsistencies here)

GM Example

	Computer-A	Computer-B	Computer-C
P1	1 sec	10 secs	20 secs
P2	1000 secs	100 secs	20 secs

Conclusion with GMs: (i) A=B
(ii) C is ~1.6 times faster

- For (i) to be true, P1 must occur 100 times for every occurrence of P2
- With the above assumption, (ii) is no longer true

Hence, GM can lead to inconsistencies

Summarizing Performance

- GM: does not require a reference machine, but does not predict performance very well
  - So we multiplied execution times and determined that sys-A is 1.2x faster…but on what workload?

- AM: does predict performance for a specific workload, but that workload was determined by executing programs on a reference machine
  - Every year or so, the reference machine will have to be updated

CPU Performance Equation

- Clock cycle time = 1 / clock speed

- CPU time = clock cycle time x cycles per instruction x
  number of instructions

- Influencing factors for each:
  - clock cycle time: technology and pipeline
  - CPI: architecture and instruction set design
  - instruction count: instruction set design and compiler

- CPI (cycles per instruction) or IPC (instructions per cycle) can not be accurately estimated analytically

An Alternative Perspective - I

- Each program is assumed to run for an equal number of cycles, so we're fair to each program

- The number of instructions executed per cycle is a measure of how well a program is doing on a system

- The appropriate summary measure is sum of IPCs or AM of IPCs = $\frac{1.2 \text{ instr}}{\text{cyc}} + \frac{1.8 \text{ instr}}{\text{cyc}} + \frac{0.5 \text{ instr}}{\text{cyc}}$

- This measure implicitly assumes that 1 instr in prog-A has the same importance as 1 instr in prog-B

An Alternative Perspective - II

- Each program is assumed to run for an equal number of instructions, so we're fair to each program

- The number of cycles required per instruction is a measure of how well a program is doing on a system

- The appropriate summary measure is sum of CPIs or AM of CPIs = $\frac{0.8 \text{ cyc}}{\text{instr}} + \frac{0.6 \text{ cyc}}{\text{instr}} + \frac{2.0 \text{ cyc}}{\text{instr}}$

- This measure implicitly assumes that 1 instr in prog-A has the same importance as 1 instr in prog-B

AM and HM

- Note that AM of IPCs = 1 / HM of CPIs and
  AM of CPIs = 1 / HM of IPCs

- So if the programs in a benchmark suite are weighted such that each runs for an equal number of cycles, then AM of IPCs or HM of CPIs are both appropriate measures

- If the programs in a benchmark suite are weighted such that each runs for an equal number of instructions, then AM of CPIs or HM of IPCs are both appropriate measures

AM vs. GM

- GM of IPCs = 1 / GM of CPIs
- AM of IPCs represents thruput for a workload where each program runs sequentially for 1 cycle each; but high-IPC programs contribute more to the AM
- GM of IPCs does not represent run-time for any real workload (what does it mean to multiply instructions?); but every program’s IPC contributes equally to the final measure

Speedup Vs. Percentage

- “Speedup” is a ratio = old exec time / new exec time

- “Improvement”, “Increase”, “Decrease” usually refer to percentage relative to the baseline
  = (new perf – old perf) / old perf

- A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  - What is the speedup?
  - What is the percentage increase in performance?
  - What is the reduction in execution time?

Title

- Bullet