Lecture: Metrics, Benchmarks, Performance

• Topics: benchmark suites, summarizing performance, performance equations

• HW1 due Wednesday 1:25pm (+ 1.5 day auto extension)
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 2017: 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?

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys-A</td>
<td>10</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>Sys-B</td>
<td>12</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>Sys-C</td>
<td>8</td>
<td>8</td>
<td>30</td>
</tr>
</tbody>
</table>

- 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)
GM Example

<table>
<thead>
<tr>
<th></th>
<th>Computer-A</th>
<th>Computer-B</th>
<th>Computer-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1 sec</td>
<td>10 secs</td>
<td>20 secs</td>
</tr>
<tr>
<td>P2</td>
<td>1000 secs</td>
<td>100 secs</td>
<td>20 secs</td>
</tr>
</tbody>
</table>

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
Problem 4

- Consider 3 programs from a benchmark set. Assume that system-A is the reference machine. How does the performance of system-B compare against that of system-C (for all 3 metrics)?

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys-A</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Sys-B</td>
<td>6</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>Sys-C</td>
<td>7</td>
<td>9</td>
<td>14</td>
</tr>
</tbody>
</table>

- Sum of execution times (AM)
- Sum of weighted execution times (AM)
- Geometric mean of execution times (GM)
Problem 4

• Consider 3 programs from a benchmark set. Assume that system-A is the reference machine. How does the performance of system-B compare against that of system-C (for all 3 metrics)?

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>S.E.T</th>
<th>S.W.E.T</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys-A</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>35</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Sys-B</td>
<td>6</td>
<td>8</td>
<td>18</td>
<td>32</td>
<td>2.9</td>
<td>9.5</td>
</tr>
<tr>
<td>Sys-C</td>
<td>7</td>
<td>9</td>
<td>14</td>
<td>30</td>
<td>3</td>
<td>9.6</td>
</tr>
</tbody>
</table>

➢ Relative to C, B provides a speedup of 1.03 (S.W.E.T) or 1.01 (GM) or 0.94 (S.E.T)
➢ Relative to C, B reduces execution time by 3.3% (S.W.E.T) or 1% (GM) or -6.7% (S.E.T)
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
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

• Note that performance is proportional to 1 / exec time

• A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  ▪ What is the speedup? \( (1/70) / (1/100) = 1.42 \)
  ▪ What is the percentage increase in performance? \( (1/70 - 1/100) / (1/100) = 42\% \)
  ▪ What is the reduction in execution time? 30\%
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
Problem 5

- My new laptop has an IPC that is 20% worse than my old laptop. It has a clock speed that is 30% higher than the old laptop. I’m running the same binaries on both machines. What speedup is my new laptop providing?
Problem 5

• My new laptop has an IPC that is 20% worse than my old laptop. It has a clock speed that is 30% higher than the old laptop. I’m running the same binaries on both machines. What speedup is my new laptop providing?

Exec time = cycle time * CPI * instrs
Perf = clock speed * IPC / instrs
Speedup = new perf / old perf
    = new clock speed * new IPC / old clock speed * old IPC
    = 1.3 * 0.8 = 1.04
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 = $1.2 \frac{\text{instr}}{\text{cyc}} + 1.8 \frac{\text{instr}}{\text{cyc}} + 0.5 \frac{\text{instr}}{\text{cyc}}$

• 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 = \(0.8 \text{ cyc} + 0.6 \text{ cyc} + 2.0 \text{ cyc}\)

  \[
  \text{instr} \quad \text{instr} \quad \text{instr}
  \]

• 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
Problem 6

- My new laptop has a clock speed that is 30% higher than the old laptop. I’m running the same binaries on both machines. Their IPCs are listed below. I run the binaries such that each binary gets an equal share of CPU time. What speedup is my new laptop providing?

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old-IPC</td>
<td>1.2</td>
<td>1.6</td>
<td>2.0</td>
</tr>
<tr>
<td>New-IPC</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
</tbody>
</table>
Problem 6

• My new laptop has a clock speed that is 30% higher than the old laptop. I’m running the same binaries on both machines. Their IPCs are listed below. I run the binaries such that each binary gets an equal share of CPU time. What speedup is my new laptop providing?

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>AM</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old-IPC</td>
<td>1.2</td>
<td>1.6</td>
<td>2.0</td>
<td>1.6</td>
<td>1.57</td>
</tr>
<tr>
<td>New-IPC</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
</tbody>
</table>

AM of IPCs is the right measure.
Speedup with AM would be 1.3.
Performance Metrics Summary

• Performance summaries: AM of weighted exec times, GM
• AM of IPCs, HM of IPCs (AM of CPIs), GM of IPCs
• Speedup (ratio), performance improvement (ratio – 1)
• CPU time = cycle time x CPI x #instructions