Lecture: Benchmarks, Pipelining Intro

• Topics: Performance equations wrap-up, Intro to pipelining
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?

<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>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>
</tr>
<tr>
<td>P2</td>
<td>1000 secs</td>
<td>100 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-C compare against that of system-B (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-C compare against that of system-B (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 B, C provides a speedup of 0.97 (S.W.E.T) or 0.99 (GM) or 1.07 (S.E.T)
➢ Relative to B, C improves 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

• A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  ▪ What is the speedup?  \( \frac{1}{70} / \frac{1}{100} = 1.42 \)
  ▪ What is the percentage increase in performance?  \( \frac{1/70 - 1/100}{1/100} = 42\% \)
  ▪ 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

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

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

• 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 throughput 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. Could have also used GM. Speedup with AM would be 1.3.
Building a Car

Unpipelined

Start and finish a job before moving to the next
The Assembly Line

Pipelined

Break the job into smaller stages

A B C
A B C
A B C
A B C

Jobs

Time
Clocks and Latches

Stage 1

Stage 2
Clocks and Latches

Stage 1 \( \rightarrow \) L \( \rightarrow \) Stage 2

Clk
Title

• Bullet