PERFORMANCE METRICS

Mahdi Nazm Bojnordi

Assistant Professor
School of Computing
University of Utah
Overview

- Announcement
  - Aug. 28th: Homework 1 release (due on Sept. 4th)
    - Verify your uploaded files before deadline

- This lecture
  - Technology trends
  - Measuring performance
  - Principles of computer design
  - Power and energy
  - Cost and reliability
Technology Trends (Historical Data)

- **IC Logic Technology**: on-chip transistor count doubles every 18-24 months (Moore’s Law)
  - Transistor density increases by 35% per year
  - Die size increases 10-20% per year

- **DRAM Technology**
  - Chip capacity increases 25-40% per year

- **Flash Storage**
  - Chip capacity increases 50-60% per year
Recent Microprocessor Trends

- Transistor count (1.43x/yr)
- Core count (1.2-1.43x/yr)
- Frequency (1.05x/yr)
- Power (1.04x/yr)

Source: Micron University Symposium
Recent Microprocessor Trends

- Transistor count (1.43x/yr)
- Core count (1.2-1.43x/yr)
- Performance (1.15x/yr)
- Frequency (1.05x/yr)
- Power (1.04x/yr)

Source: Micron University Symposium
Measuring Performance

- How to measure performance?
Measuring Performance

- How to measure performance?
  - **Latency or response time**
    - The time between start and completion of an event (e.g., milliseconds for disk access)
  - **Bandwidth or throughput**
    - The total amount of work done in a given time (e.g., megabytes per second for disk transfer)
Measuring Performance

- How to measure performance?
  - Latency or response time
    - The time between start and completion of an event (e.g., milliseconds for disk access)
  - Bandwidth or throughput
    - The total amount of work done in a given time (e.g., megabytes per second for disk transfer)
- Which one is better? latency or throughput?
Measuring Performance

Which one is better (faster)?

- **Car**
  - Delay = 10m
  - Capacity = 4p

- **Bus**
  - Delay = 30m
  - Capacity = 30p
Measuring Performance

Which one is better (faster)?

- **Car**
  - Delay=10m
  - Capacity=4p
  - Throughput=0.4PPM

- **Bus**
  - Delay=30m
  - Capacity=30p
  - Throughput=1PPM

It really depends on your needs (goals).
Measuring Performance

- What program to use for measuring performance?
- Benchmarks Suites
  - A set of representative programs that are likely relevant to the user
  - Examples:
    - SPEC CPU 2017: CPU-oriented programs (for desktops)
    - SPECweb: throughput-oriented (for servers)
    - EEMBC: embedded processors/workloads
Summarizing Performance Numbers

- How to capture the behavior of multiple programs with a single number

<table>
<thead>
<tr>
<th></th>
<th>Comp-A</th>
<th>Comp-B</th>
<th>Comp-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog-1</td>
<td>10</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>Prog-2</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Prog-3</td>
<td>25</td>
<td>10</td>
<td>25</td>
</tr>
</tbody>
</table>
Summarizing Performance Numbers

- How to capture the behavior of multiple programs with a single number

<table>
<thead>
<tr>
<th></th>
<th>Comp-A</th>
<th>Comp-B</th>
<th>Comp-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog-1</td>
<td>10</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>Prog-2</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Prog-3</td>
<td>25</td>
<td>10</td>
<td>25</td>
</tr>
</tbody>
</table>

❖ AM: Arithmetic Mean (good for times and latencies)

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]
Summarizing Performance Numbers

- How to capture the behavior of multiple programs with a single number

<table>
<thead>
<tr>
<th></th>
<th>Comp-A</th>
<th>Comp-B</th>
<th>Comp-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog-1</td>
<td>1/10</td>
<td>1/5</td>
<td>1/25</td>
</tr>
<tr>
<td>Prog-2</td>
<td>1/5</td>
<td>1/10</td>
<td>1/20</td>
</tr>
<tr>
<td>Prog-3</td>
<td>1/25</td>
<td>1/10</td>
<td>1/25</td>
</tr>
</tbody>
</table>
Summarizing Performance Numbers

- How to capture the behavior of multiple programs with a single number

<table>
<thead>
<tr>
<th></th>
<th>Comp-A</th>
<th>Comp-B</th>
<th>Comp-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog-1</td>
<td>1/10</td>
<td>1/5</td>
<td>1/25</td>
</tr>
<tr>
<td>Prog-2</td>
<td>1/5</td>
<td>1/10</td>
<td>1/20</td>
</tr>
<tr>
<td>Prog-3</td>
<td>1/25</td>
<td>1/10</td>
<td>1/25</td>
</tr>
</tbody>
</table>

❖ HM: Harmonic Mean (good for rates and throughput)

\[
\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}
\]
How to capture the behavior of multiple programs with a single number

<table>
<thead>
<tr>
<th></th>
<th>Comp-A</th>
<th>Comp-B</th>
<th>Comp-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog-1</td>
<td>10/10</td>
<td>10/5</td>
<td>10/25</td>
</tr>
<tr>
<td>Prog-2</td>
<td>5/5</td>
<td>5/10</td>
<td>5/20</td>
</tr>
<tr>
<td>Prog-3</td>
<td>25/25</td>
<td>25/10</td>
<td>25/25</td>
</tr>
</tbody>
</table>
Summarizing Performance Numbers

How to capture the behavior of multiple programs with a single number

<table>
<thead>
<tr>
<th></th>
<th>Comp-A</th>
<th>Comp-B</th>
<th>Comp-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog-1</td>
<td>10/10</td>
<td>10/5</td>
<td>10/25</td>
</tr>
<tr>
<td>Prog-2</td>
<td>5/5</td>
<td>5/10</td>
<td>5/20</td>
</tr>
<tr>
<td>Prog-3</td>
<td>25/25</td>
<td>25/10</td>
<td>25/25</td>
</tr>
</tbody>
</table>

GM: Geometric Mean (good for speedups)

\[
\left( \prod_{i=1}^{n} x_i \right)^{1/n}
\]
Processor Performance

- Clock cycle time (CT = 1/clock frequency)
  - Influenced by technology and pipeline
- Cycles per instruction (CPI)
  - Influenced by architecture
  - IPC may be used instead (IPC = 1/CPI)
- Instruction count (IC)
  - Influenced by ISA and compiler
- CPU time = IC x CPI x CT
Find the average CPI of a load/store machine when running an application that results in the following statistics:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>ALU</td>
<td>40%</td>
<td>1</td>
</tr>
</tbody>
</table>
Example Problem

Find the average CPI of a load/store machine when running an application that results in the following statistics

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>ALU</td>
<td>40%</td>
<td>1</td>
</tr>
</tbody>
</table>

$$CPI = 0.2 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.4 \times 1 = 1.6$$
Example Problem

Find the average CPI of a load/store machine when running an application that results in the following statistics:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>ALU</td>
<td>40%</td>
<td>1</td>
</tr>
</tbody>
</table>

50% of the branches can be combined with ALU instructions and executed as Branch-ALU fused in 2 cycles. What is the new average CPI?
Example Problem

- Find the average CPI of a load/store machine when running an application that results in the following statistics:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>~22%</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>~22%</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>~11%</td>
<td>2</td>
</tr>
<tr>
<td>ALU</td>
<td>~33%</td>
<td>1</td>
</tr>
<tr>
<td>Branch-ALU</td>
<td>~12%</td>
<td>2</td>
</tr>
</tbody>
</table>

- 50% of the branches can be combined with ALU instructions and executed as Branch-ALU fused in 2 cycles. What is the new average CPI? \( \text{CPI} = 1.67 \)
Points to note

- Performance $= \frac{1}{\text{execution time}}$
- AM(IPCs) $= \frac{1}{\text{HM(CPIs)}}$
- GM(IPCs) $= \frac{1}{\text{GM(CPIs)}}$

\[
\frac{1}{n} \sum_{i=1}^{n} x_i \quad \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \quad \left( \prod_{i=1}^{n} x_i \right)^{1/n}
\]
Speedup vs. Percentage

- Speedup = old execution time / new execution time
- Improvement = (new performance - old performance) / old performance
- My old and new computers run a particular program in 80 and 60 seconds; compute the followings
  - speedup
  - percentage increase in performance
  - percentage reduction in execution time
Speedup vs. Percentage

- Speedup = old execution time / new execution time

- Improvement = (new performance - old performance)/old performance

My old and new computers run a particular program in 80 and 60 seconds; compute the followings

- speedup = 80/60 = ~1.33
- percentage increase in performance = 33%
- percentage reduction in execution time = 20/80 = 25%
Example Problem

- The IPC of a new computer is 20% worse than the old one. Its clock speed is 30% higher than the old one. If running the same binaries on both machines. What speedup is the new computer providing?
The IPC of a new computer is 20% worse than the old one. Its clock speed is 30% higher than the old one. If running the same binaries on both machines. What speedup is the new computer providing?

<table>
<thead>
<tr>
<th></th>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPC</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>Frequency</td>
<td>1</td>
<td>1.3</td>
</tr>
<tr>
<td>IC</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CPI</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CT</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CPU Time</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Example Problem

The IPC of a new computer is 20% worse than the old one. Its clock speed is 30% higher than the old one. If running the same binaries on both machines. What speedup is the new computer providing?

Speedup = 1/0.96 = 1.04
Principles of Computer Design

- Designing better computer systems requires better utilization of resources

  - **Parallelism**
    - Multiple units for executing partial or complete tasks

  - **Principle of locality (temporal and spatial)**
    - Reuse data and functional units

  - **Common Case**
    - Use additional resources to improve the common case