PERFORMANCE METRICS

Mahdi Nazm Bojnordi
Assistant Professor
School of Computing
University of Utah
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)
- Performance (1.15x/yr)
- Frequency (1.05x/yr)
- Power (1.04x/yr)

Source: Micron University Symposium
Performance Trends

- 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 grows faster?
  - Bandwidth, by at least the square of latency improvement rate.

- Which one is better? latency or throughput?
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 2006: 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>

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

\[
\frac{1}{n} \sum_{i=1}^{n} x_i
\]
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>

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

\[
\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}
\]
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>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)

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

- **Clock cycle time** \((CT = 1 / \text{clock frequency})\)
  - Influenced by technology and pipeline

- **Cycles per instruction** (CPI)
  - Influenced by architecture
  - IPC may be used instead \((\text{IPC} = 1 / \text{CPI})\)

- **Instruction count** (IC)
  - Influenced by ISA and compiler

- **CPU time** \(= \text{IC} \times \text{CPI} \times CT\)
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>

\[ \text{CPI} = 0.2 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.4 \times 1 = 1.6 \]
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>

80% of the branches can be combined with ALU instructions and executed as Branch-ALU fused in 2 cycles. What is the new average CPI? CPI = 1.67
The Processor Performance

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

\[
\frac{1}{n} \sum_{i=1}^{n} x_i, \quad \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}, \quad \left( \prod_{i=1}^{n} x_i \right)^{1/n}
\]
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 = 80/60
  - percentage increase in performance = 33%
  - reduction in execution time = 20/80 = 25%
A new computer has an IPC that is 20% worse than the old one. However, it has a clock speed that is 30% higher than the old one. If running the same binaries on both machines. What speedup is the new computer providing? 

\[ \text{Speedup} = \frac{1}{0.96} = 1.04 \]

<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>1/1</td>
<td>1/0.8 = 1.25</td>
</tr>
<tr>
<td>CT</td>
<td>1/1</td>
<td>1/1.3 ~ 0.77</td>
</tr>
<tr>
<td>CPU Time</td>
<td>1</td>
<td>~0.96</td>
</tr>
</tbody>
</table>
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
Amdahl’s Law

- The law of diminishing returns

\[
\text{Execution time}_{new} = \text{Execution time}_{old} \times \left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \\
\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{old}}{\text{Execution time}_{new}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}
\]
Example Problem

- Our new processor is 10x faster on computation than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for IO 60% of the time, what is the overall speedup?

\[
f = 0.4 \quad s = 10
\]

\[
\text{Speedup} = \frac{1}{(0.6 + 0.4/10)} = \frac{1}{0.64} = 1.5625
\]
Power and Energy
Power and Energy

- Power = Voltage x Current (P = VI)
  - Instantaneous rate of energy transfer (Watt)
- Energy = Power x Time (E = PT)
  - The cost of performing a task (Joule)
Power and Energy

- Power = Voltage x Current \( (P = VI) \)
  - Instantaneous rate of energy transfer (Watt)
- Energy = Power x Time \( (E = PT) \)
  - The cost of performing a task (Joule)

**Graph:**
- Peak Power = 3W
- Average Power = 1.66W
- Total Energy = 5J
CPU Power and Energy

- All consumed energy is converted to heat
  - CPU power is the rate of heat generation
  - Excessive peak power may result in burning the chip
- Static and dynamic energy components
  - Energy = (Power_{Static} + Power_{Dynamic}) x Time
  - Power_{Static} = Voltage x Current_{Static}
  - Power_{Dynamic} = Activity x Capacitance x Voltage^2 x Frequency

\[ \text{Energy} = (\text{Power}_{\text{Static}} + \text{Power}_{\text{Dynamic}}) \times \text{Time} \]
\[ \text{Power}_{\text{Static}} = \text{Voltage} \times \text{Current}_{\text{Static}} \]
\[ \text{Power}_{\text{Dynamic}} = \text{Activity} \times \text{Capacitance} \times \text{Voltage}^2 \times \text{Frequency} \]
Power Reduction Techniques

- Reducing voltage (V)
  - Negative effect on frequency
  - Opportunity to power gate (wake up time)
  - Dynamic voltage and frequency scaling

- Reducing frequency (f)
  - Negative effect on CPU time
  - Clock gating in unused resources

- Points to note
  - Utilization directly effects dynamic power
  - Lowering power does NOT mean lowering energy
Example Problem

For a processor running at 100% utilization at 100W, 30% of the power is attributed to leakage. What is the total power dissipation when the processor is running at 50% utilization?
Example Problem

For a processor running at 100% utilization at 100W, 30% of the power is attributed to leakage. What is the total power dissipation when the processor is running at 50% utilization?

@100%
- Power = 30W + 70W = 100W

@50%
- Power = 30W + 35W = 65W
A processor consumes 80W of dynamic power and 20W of static power at 3GHz. It completes a program in 20 seconds. What is the energy consumption if frequency scales down by 20%?
Example Problem

- A processor consumes 80W of dynamic power and 20W of static power at 3GHz. It completes a program in 20 seconds. What is the energy consumption if frequency scales down by 20%?

- @3GHz
  - Energy = (80W + 20W) x 20s = 2000J

- @2.4GHz
  - Energy = (0.8x80W + 20W) x 20/0.8 = 2100J
Example Problem

- A processor consumes 80W of dynamic power and 20W of static power at 3GHz. It completes a program in 20 seconds. What is the energy consumption if frequency scales down by 20%?
- What is the energy consumption if voltage and frequency scale down by 20%?
Example Problem

- A processor consumes 80W of dynamic power and 20W of static power at 3GHz. It completes a program in 20 seconds. What is the energy consumption if frequency scales down by 20%?

- What is the energy consumption if voltage and frequency scale down by 20%?

- @ 80%V and 80%f
  - Energy = (80x0.8^2x0.8+20x0.8) x 20/0.8 = 1424J
Cost and Reliability
Cost of Integrated Circuit

- **Cost of die**
  \[
  \frac{\text{wafer cost}}{\text{dies per wafer} \times \text{die yield}}
  \]

- **Yield of die**
  \[
  \frac{\text{wafer yield}}{(1+\text{defect per unit area} \times \text{die area})^N}
  \]

- **N**: process-complexity factor
  - Specified by chip manufacturer
Defect rate for a 144mm\(^2\) die is 0.5 per cm\(^2\). Assuming that we use a 40nm technology node (N=11), find the die yield.
Example Problem

- Defect rate for a 144mm$^2$ die is 0.5 per cm$^2$. Assuming that we use a 40nm technology node (N=11), find the die yield.

- Die yield = $1/(1 + 0.5 \times 1.44)^{11}$
Dependability

- A measure of system's reliability and availability

- System reliability
  - A measure of continuous service (time-to-failure)
  - Mean Time To Failure (MTTF)
  - Mean Time To Repair (MTTR)

- System availability
  \[
  \text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}
  \]
Dependability

- A measure of system's reliability and availability

- System reliability
  - A measure of continuous service (time-to-failure)
  - Mean Time To Failure (MTTF) = \( \frac{3+2+1}{3} = 2 \)
  - Mean Time To Repair (MTTR) = \( \frac{0.75+1+1.25}{3} = 1 \)

- System availability
  - \( \frac{2}{2+1} = 0.67 \)