Performance

Hwansoo Han
Defining Performance

Which airplane has the best performance?

- **Passenger Capacity**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50

- **Cruising Range (miles)**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50

- **Cruising Speed (mph)**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50

- **Passengers x mph**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50
Response Time and Throughput

- **Response time**
  - How long it takes to do a task

- **Throughput**
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour

- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?

- We’ll focus on response time for now…
Relative Performance

- Define Performance = \( 1 / \) Execution Time
- “\( X \) is \( n \) times faster than \( Y \)”

\[
\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{ExecutionTime}_Y}{\text{ExecutionTime}_X} = n
\]

- Example: Time taken to run a program
  - 10 seconds on Machine A, 15 seconds on Machine B

  \[
  \frac{\text{ExecutionTime}_B}{\text{ExecutionTime}_A} = \frac{15}{10} = 1.5
  \]

  - A is 1.5 times faster than B
Measuring Execution Time

- **Elapsed time**
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
  - Not good for comparison when computers are shared

- **CPU time**
  - Time spent processing a given job
    - Discounts I/O time, other jobs’ shares
  - Comprised of user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance
CPU Clocking

Operation of digital hardware governed by a constant-rate clock

- Clock period: duration of a clock cycle
  - e.g. 250 ps = 0.25 ns = 250 x 10^{-12} s

- Clock frequency (rate): cycles per second
  - e.g. 4.0 GHz = 4000 MHz = 4.0 x 10^9 Hz
CPU Time

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

\[
\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}
\]

\[
= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}
\]
CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - New technology can allow faster clock
  - But this causes $1.2 \times$ clock cycles
- How fast must Computer B clock be?

Clock Rate$_B$ = Clock Cycle$_B$ / CPU Time$_B$ = $1.2 \times$ Clock Cycle$_A$ / 6s

Clock Cycle$_A$ = CPU Time$_A$ x Clock Rate$_A$ = 10s x 2GHz = $20 \times 10^9$

Clock Rate$_B$ = $1.2 \times 20 \times 10^9$ / 6s = $24 \times 10^9$ / 6s = 4GHz
Instruction Count and CPI

- **Instruction Count for a program**
  - Determined by program, ISA and compiler

- **Average cycles per instruction**
  - Determined by CPU hardware
  - If different instructions have different CPI, average CPI affected by instruction mix

\[
\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}
\]

\[
\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}
\]

\[
= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}
\]
CPI Example

- Small CPI means faster computer?
  - Computer A: Cycle Time = 250ps, CPI = 2.0
  - Computer B: Cycle Time = 500ps, CPI = 1.2
  - Assume the same ISA $\Rightarrow$ the same instruction count = $I$

- Which is faster, and by how much?

  \[
  \text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A
  \]
  \[
  = I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}
  \]

  \[
  \text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B
  \]
  \[
  = I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}
  \]

  \[
  \frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2
  \]

  A is faster...

  ...by this much
CPI in More Detail

- If different instruction classes take different numbers of cycles

\[
\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)
\]

- Weighted average CPI

\[
\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left[ \text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right]
\]

Relative frequency
CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

<table>
<thead>
<tr>
<th>Class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI for class</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>IC in sequence 1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>IC in sequence 2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sequence 1: IC = 5
- Clock cycles
  = 2 x 1 + 1 x 2 + 2 x 3
  = 10
- Avg. CPI = 10/5 = 2.0

Sequence 2: IC = 6
- Clock cycles
  = 4 x 1 + 1 x 2 + 1 x 3
  = 9
- Avg. CPI = 9/6 = 1.5
Factors Affecting Performance

- Performance depends on
  - Algorithm – affects IC, possibly CPI
  - Programming language – affects IC, CPI
  - Compiler – affects IC, CPI
  - Instruction set architecture – affects IC, CPI, $T_c$

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$
Benchmarks

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications
    e.g., compilers/editors, scientific applications, graphics, etc.

- Small benchmarks
  - Nice for architects and designers
  - Easy to standardize
  - Can be abused

- SPEC (Standard Performance Evaluation Corporation)
  - Companies have agreed on a set of real program and inputs
  - Valuable indicator of performance (and compiler technology)
  - SPEC CPU2006, SPEC Web2005, …
An embarrassed Intel Corp. acknowledged Friday that a bug in a software program known as a compiler had led the company to overstate the speed of its microprocessor chips on an industry benchmark by 10 percent. However, industry analysts said the coding error…was a sad commentary on a common industry practice of “cheating” on standardized performance tests…The error was pointed out to Intel two days ago by a competitor, Motorola …came in a test known as SPECint92…Intel acknowledged that it had “optimized” its compiler to improve its test scores. The company had also said that it did not like the practice but felt to compelled to make the optimizations because its competitors were doing the same thing…At the heart of Intel’s problem is the practice of “tuning” compiler programs to recognize certain computing problems in the test and then substituting special handwritten pieces of code…
Other Benchmarks

- **EEMBC**
  - Applications on embedded systems such as
    - Networking/telecom. Devices, automobiles, consumer media devices, etc.

- **Mediabench**
  - Set of multimedia applications (i.e., codec, graphics, etc.)

- **NAS**
  - Parallel benchmarks from NASA

- **SPLASH, PARSEC**
  - Multithreaded benchmarks for multicores/multiprocessors

- **BioBench**
  - Gene sequence search algorithms

- **TPC benchmark**
  - Transaction processing performance
  - TPC-C, TPC-W, …
Amdahl's Law

- **Execution time after improvement** \( (\text{Exec\_time}_{\text{new}}) \)
  \[
  \text{Exec\_time}_{\text{new}} = \text{Exec\_time}_{\text{unaffected}} + \left( \frac{\text{Exec\_time}_{\text{affected}}}{\text{Amount\_of\_improvement}} \right) = \text{Exec\_time}_{\text{org}} \times (1 - f) + \frac{f}{S}
  \]

- **Example:**
  Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time.
  
  - How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?
  - How about making it 5 times faster?
Speedup and Amdahl’s Law

- **Speedup**
  \[
  \text{Speedup} = \frac{\text{Exec\_time}_\text{org}}{\text{Exec\_time}_\text{new}} = \frac{1}{(1 - f) + f / S}
  \]

- **Principles**
  - Make the common case fast
    - As \( f \to 1 \), speedup \( \to S \)
  - Speedup is limited by the fraction of code that can be optimized
    - As \( S \to \infty \), speedup \( \to 1 / (1 - f) \)
  - Uncommon case can become the common one after improvement
Example

- Suppose we enhance a machine making all floating-point instructions run five times faster.
  - If the execution time of some benchmark before the floating-point enhancement is 10 seconds
  - What will the speedup be, if half of the 10 seconds is spent executing floating-point instructions?

- We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3.
  - One benchmark we are considering runs for 100 seconds with the old floating-point hardware.
  - How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?
Summary

- Performance is specific to a particular program(s)
  - Total execution time is a consistent summary of the performance

- For a given architecture, performance increases come from:
  - Increases in clock rate (without adverse CPI affects)
  - Improvements in processor organization that lower CPI
  - Compiler enhancements that lower CPI and/or instruction count
  - Algorithm/Language choices that affect instruction count

- Pitfall:
  - Expecting improvement in one aspect of a machine’s performance to affect the total performance