CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Instructors:
Randy H. Katz
David A. Patterson

http://inst.eecs.Berkeley.edu/~cs61c/fa10
Agenda

• Review
• Defining Performance, Latency vs. Bandwidth
• Administrivia
• Technology Break
• Measuring Performance
Review

• Translate from text that easy for programmers to understand into code that machine executes efficiently: Compilers, Assemblers
• Linkers allow separate translation of modules
• Interpreters for debugging, but slow execution
• Hybrid (Java): Compiler + Interpreter to try to get best of both
Defining Performance

• What does it mean to say X is faster than Y?
• Ferrari vs. School Bus?
• 2009 Ferrari 599 GTB
  – 2 passengers, 11.1 secs in quarter mile
• 2009 Type D school bus
  – 54 passengers, quarter mile time?
  http://www.youtube.com/watch?v=KwyCoQuhUNA
• Response Time or Latency: time between start and completion of a task (time to move vehicle ¼ mile)
• Throughput or Bandwidth: total amount of work in a given time (passenger-miles in 1 hour)
Running Systems to 100% Utilization

- Implication of the graph at the right?

- Can you explain why this happens?
The Iron Law of Queues
(aka Little’s Law)

\[ L = \lambda W \]

Average number of customers in system (L) = average interarrival rate (\( \lambda \)) \times average service time (W)
Cloud Performance: Why Application Latency Matters

- Key figure of merit: application responsiveness
  - Longer the delay, the fewer the user clicks, the less the user happiness, and the lower the revenue per user

<table>
<thead>
<tr>
<th>Server Delay (ms)</th>
<th>Increased time to next click (ms)</th>
<th>Queries/user</th>
<th>Any clicks/user</th>
<th>User satisfaction</th>
<th>Revenue/User</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>200</td>
<td>500</td>
<td>--</td>
<td>-0.3%</td>
<td>-0.4%</td>
<td>--</td>
</tr>
<tr>
<td>500</td>
<td>1200</td>
<td>--</td>
<td>-1.0%</td>
<td>-0.9%</td>
<td>-1.2%</td>
</tr>
<tr>
<td>1000</td>
<td>1900</td>
<td>-0.7%</td>
<td>-1.9%</td>
<td>-1.6%</td>
<td>-2.8%</td>
</tr>
<tr>
<td>2000</td>
<td>3100</td>
<td>-1.8%</td>
<td>-4.4%</td>
<td>-3.8%</td>
<td>-4.3%</td>
</tr>
</tbody>
</table>

Figure 6.10 Negative impact of delays at Bing search server on user behavior [Brutlag and Schurman 2009].
Defining Relative Performance

• Performance\(_X\) = 1/Program Execution Time\(_X\)
• Performance\(_X\) > Performance\(_Y\) =>
  1/Execution Time\(_X\) > 1/Execution Time\(_Y\) =>
  Execution Time\(_Y\) > Execution Time\(_X\)
• Computer X is N times faster than Computer Y
  Performance\(_X\) / Performance\(_Y\) = N or
  Execution Time\(_Y\) / Execution Time\(_X\) = N
• Ferrari is 12/11.1 or 1.08 times faster than bus
Measuring Performance

• Computers use a clock to determine when events takes place within hardware

  • Clock cycles — discrete time intervals
    — a.k.a. clocks, cycles, clock periods, clock ticks

  • Clock rate or clock frequency — clock cycles per second (inverse of clock cycle time)

• 3 GigaHertz clock rate
  => clock cycle time = 1/3x10⁹ seconds
  clock cycle time = 333 picoseconds (ps)
CPU Performance Factors

• To distinguish between processor time and I/O, *CPU time* is time spent in processor

• CPU time/program
  = Clock cycles/program x clock cycle time

• Or CPU time/program
  = Clock cycles/program ÷ clock rate
CPU Performance Factors

• But a program executes instructions
• CPU time/program
  = Clock cycles/program x clock cycle time
  = Instructions/program
    x Average Clock cycles/Instruction
    x clock cycle time
• 1st term called Instruction Count
• 3rd term is 1 / Clock rate
• 2nd term abbreviated CPI for average Clock cycles Per Instruction
• Why CPI = 1? Why CPI > 1? Can CPI be < 1?
Restating Performance Equation

\[ \text{Time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}} \]
What affects each component?

Instruction Count, CPI, Clock Rate

<table>
<thead>
<tr>
<th>Hardware or software component?</th>
<th>Affects What?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td></td>
</tr>
<tr>
<td>Programming Language</td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td></td>
</tr>
<tr>
<td>Instruction Set Architecture</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

• Review
• Defining Performance, Latency vs. Bandwidth
• Administrivia
• Technology Break
• Measuring Performance
Breaking News: Google Instant Search

“Instant Efficiency”

Typical search takes 24 seconds, Google’s search algorithm is only 300 ms of this
“It’s not search ‘as you type’, but ‘search before you type’!”
“We can predict what you are likely to type and give you those results in real time”
Peer Instruction

• Computer A clock cycle time 250 ps, $\text{CPI}_A = 2$
• Computer B clock cycle time 500 ps, $\text{CPI}_B = 1.2$
• Assume A and B have same instruction set
• Which statement is true?
  A. Computer A is $\sim 1.2$ times faster than B
  B. Computer A is $\sim 4.0$ times faster than B
  C. Computer B is $\sim 1.7$ times faster than A
  D. Computer B is $\sim 3.4$ times faster than A
  E. None of the above
Workload and Benchmark

• *Workload*: A set of programs run on a computer
  – The actual collection of applications run or made from real programs to approximate such a mix
  – Specifies both programs and relative frequencies

• *Benchmark* A program selected for use in comparing computer performance
  – Benchmarks form a workload
  – Usually standardized so that many use them
SPEC (System Performance Evaluation Cooperative)

• Computer Vendor cooperative for benchmarks, started in 1989
• SPECCPU2006
  – 12 Integer Programs
  – 17 Floating-Point Programs
• Often turn into number where bigger is faster
• \textit{SPECratio}: reference execution time on old reference computer divide by execution time on new computer
# SPECINT2006 on AMD Barcelona

<table>
<thead>
<tr>
<th>Description</th>
<th>Instruction Count (B)</th>
<th>CPI</th>
<th>Clock cycle time (ps)</th>
<th>Execution Time (s)</th>
<th>Reference Time (s)</th>
<th>SPEC-ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreted string processing</td>
<td>2,118</td>
<td>0.75</td>
<td>400</td>
<td>637</td>
<td>9,770</td>
<td>15.3</td>
</tr>
<tr>
<td>Block-sorting compression</td>
<td>2,389</td>
<td>0.85</td>
<td>400</td>
<td>817</td>
<td>9,650</td>
<td>11.8</td>
</tr>
<tr>
<td>GNU C compiler</td>
<td>1,050</td>
<td>1.72</td>
<td>400</td>
<td>724</td>
<td>8,050</td>
<td>11.1</td>
</tr>
<tr>
<td>Combinatorial optimization</td>
<td>336</td>
<td>10.0</td>
<td>400</td>
<td>1,345</td>
<td>9,120</td>
<td>6.8</td>
</tr>
<tr>
<td>Go game</td>
<td>1,658</td>
<td>1.09</td>
<td>400</td>
<td>721</td>
<td>10,490</td>
<td>14.6</td>
</tr>
<tr>
<td>Search gene sequence</td>
<td>2,783</td>
<td>0.80</td>
<td>400</td>
<td>890</td>
<td>9,330</td>
<td>10.5</td>
</tr>
<tr>
<td>Chess game</td>
<td>2,176</td>
<td>0.96</td>
<td>400</td>
<td>837</td>
<td>12,100</td>
<td>14.5</td>
</tr>
<tr>
<td>Quantum computer simulation</td>
<td>1,623</td>
<td>1.61</td>
<td>400</td>
<td>1,047</td>
<td>20,720</td>
<td>19.8</td>
</tr>
<tr>
<td>Video compression</td>
<td>3,102</td>
<td>0.80</td>
<td>400</td>
<td>993</td>
<td>22,130</td>
<td>22.3</td>
</tr>
<tr>
<td>Discrete event simulation library</td>
<td>587</td>
<td>2.94</td>
<td>400</td>
<td>690</td>
<td>6,250</td>
<td>9.1</td>
</tr>
<tr>
<td>Games/path finding</td>
<td>1,082</td>
<td>1.79</td>
<td>400</td>
<td>773</td>
<td>7,020</td>
<td>9.1</td>
</tr>
<tr>
<td>XML parsing</td>
<td>1,058</td>
<td>2.70</td>
<td>400</td>
<td>1,143</td>
<td>6,900</td>
<td>6.0</td>
</tr>
</tbody>
</table>
Summary

• Time (seconds/program) is measure of performance

\[
\text{Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}
\]

• Algorithms, Programming Languages, Compilers, Instruction Set affect Instruction Count and CPI

• Instruction Set affects Clock Period/ Clock Rate

• Benchmarks stand in for real workloads to as standardized measure of relative performance