CSE 141: Introduction to Computer Architecture

Performance
Thought Experiment

- What is the fastest way to send a picture of a black hole to Boston?
- What is the fastest way to send 5 petabytes of data to Boston?

\[
\text{(5 petabytes)} / \text{(940 Megabits/second)} = 1.35 \text{ years}
\]
Graphs that go up and to the right are good, but what do they mean?
The bottom line: Performance

- Time to do the task
  - *execution time*, response time, latency
- Tasks per day, hour, week, sec, ns...
  - *throughput*, bandwidth

<table>
<thead>
<tr>
<th></th>
<th>Time to Bay Area</th>
<th>Speed</th>
<th>Passengers</th>
<th>Throughput (pmph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ferrari</td>
<td>3.1 hours</td>
<td>160 mph</td>
<td>2</td>
<td>320</td>
</tr>
<tr>
<td>Bus</td>
<td>7.7 hours</td>
<td>65 mph</td>
<td>60</td>
<td>3900</td>
</tr>
</tbody>
</table>
Measures of “Performance”

• Execution Time
• Throughput (operations/time)
  – Transactions/sec, queries/day, etc.
• Frame Rate
• Responsiveness
• Performance / Cost
• Performance / Power
• Performance / Energy
There are many ways to measure program execution time

```bash
$ time make  # cargo build
Compiling hail v0.1.0 (/tock/boards/hail)
   Finished release [optimized + debuginfo] target(s) in **19.96s**
```

- Program-reported time?
- Wall-clock time?
- user CPU time?
- user + kernel CPU time?
Our definition of Performance

\[
\text{Performance}_X = \frac{1}{\text{Execution Time}_X}, \text{ for program } X
\]

• Only has meaning in the context of a **program** or **workload**
• Not very intuitive as an absolute measure, but most of the time we’re more interested in **relative performance**
Relative Performance

• Can be confusing...
  A runs in 12 seconds
  B runs in 20 seconds
  – A/B = .6, so A is 40% faster, or 1.4X faster, or B is 40% slower
  – B/A = 1.67, so A is 67% faster, or 1.67X faster, or B is 67% slower

• Needs a precise definition
Relative Performance (Speedup), the Definition

\[
\text{Speedup } (X/Y) = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n
\]
Example

• Machine A runs program C in 9 seconds.
• Machine B runs the same program in 6 seconds.
• What is the speedup we see if we move to Machine B from Machine A?
Poll Question: What is the speedup?

- Machine A runs program C in 9 seconds.
- Machine B runs the same program in 6 seconds.
- Machine B gets a new compiler, and can now run the program in 3 seconds.
- What is the speedup from the new compiler?

When you have your answer, type it into the chat box in Zoom, but do not hit enter.
What is Time?

CPU Execution Time = CPU clock cycles * Clock cycle time

- Every conventional processor has a clock with an associated clock cycle time or clock rate
- Every program runs in an integral number (whole number) of clock cycles

Cycle Time

MHz = millions of cycles/second, GHz = billions of cycles/second
X MHz = 1000/X nanoseconds cycle time
Y GHz = 1/Y nanoseconds cycle time
How many clock cycles?

Number of CPU clock cycles =

\[\text{Instruction count} \times \text{Average Clock Cycles per Instruction (CPI)}\]

Exercise:

Computer A runs program C in 3.6 billion cycles.
Program C requires 2 billion dynamic instructions.
What is the CPI?
Poll Question: How many clock cycles?

A computer is running a program with CPI = 2.0. It executes 24 million instructions. How long will it run?

<table>
<thead>
<tr>
<th>Selection</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2.4 seconds</td>
</tr>
<tr>
<td>B</td>
<td>12 million cycles</td>
</tr>
<tr>
<td>C</td>
<td>48 million seconds</td>
</tr>
<tr>
<td>D</td>
<td>48 million cycles</td>
</tr>
<tr>
<td>E</td>
<td>None of the above</td>
</tr>
</tbody>
</table>
Putting it all together

CPU Execution Time = [CPU clock cycles] * [Clock cycle time]

CPU clock cycles = [Instruction count] * [Average Clock Cycles per Instruction (CPI)]
Poll Question: All Together Now

- Instruction Count = 4 billion
- 2 GHz processor
- Execution time of 3 seconds

What is the CPI for this program?

When you have your answer, type it into the chat box in Zoom, but do not hit enter
Cycle Time/Clock Rate is no longer fixed

- Increasingly, modern processors can execute at multiple clock rates (cycle times).
- Why?
- However, the granularity at which we can change the cycle time tends to be fairly coarse, so all of these principles and formulas still apply.
Who Affects Performance? How?

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

- programmer
- compiler
- instruction-set architect
- machine architect
- hardware designer
- materials scientist/physicist/silicon engineer
Performance Variation: What affects what?

CPU Execution Time \[=\] Instruction Count \(\times\) CPI \(\times\) Clock Cycle Time

<table>
<thead>
<tr>
<th></th>
<th>Number of Instructions</th>
<th>CPI</th>
<th>Clock Cycle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same machine, different programs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sam programs, different machine, same ISA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Same programs, different machines</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MIPS
(the performance measure, not the architecture...)

MIPS – “Millions of Instructions Per Second”

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution Time} \times 10^6}$$

$$= \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

• Program-independent
• Deceptive!

Some also discuss [M]FLOPS
“Floating point operations per second”
Which programs are best, are “most fair”, to run when measuring performance?

- peak throughput measures (simple programs)?
- synthetic benchmarks (whetstone, dhrystone,...)?
- Real applications
- SPEC (best of both worlds, but with problems of their own)
  - System Performance Evaluation Cooperative
  - Provides a common set of real applications
    - Along with strict guidelines for how to run them
  - Provides a relatively unbiased means to compare machines.
Amdahl's Law

- The impact of a performance improvement is limited by the percent of execution time affected by the improvement

\[
\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}
\]

- Make the common case fast!!
Amdahl’s Law and Massive Parallelism

0.9

0.1
Amdahl’s Law and Massive Parallelism

**Speedup**

\[
\frac{1}{0.55} = 1.82
\]
Amdahl’s Law and Massive Parallelism

\[ \text{Speedup} = \frac{1}{0.9 + 0.1} = 1.82 \]

\[ \text{Speedup} = \frac{1}{0.45 + 0.1} = 3.07 \]

\[ \text{Speedup} = \frac{1}{0.225 + 0.1} \]
Amdahl’s Law and Massive Parallelism

\[ \text{Speedup} = \frac{1}{0.55} = 1.82 \]

\[ \text{Speedup} = \frac{1}{0.325} = 3.07 \]

\[ \text{Speedup} < 10 \]
Key Points

• Be careful how you specify performance
• Execution time = instructions * CPI * cycle time
• Use real applications
• Use standards, if possible
• Make the common case fast