IC220
Slide Set #12: Performance
(Chapter 1: 1.4, 1.7-1.9)

• Measure, Report, and Summarize
• Make intelligent choices
• See through the marketing hype
• Key to understanding underlying organizational motivation

Why is some hardware better than others for different programs? What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new operating system?)

How does the machine’s instruction set affect performance?

---

### Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation

Why is some hardware better than others for different programs?

What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new operating system?)

How does the machine’s instruction set affect performance?

---

#### Computer Performance:

- **Execution / Response Time (latency)**
  - How long does it take for my job to run?
  - How long does it take to execute a job?
  - How long must I wait for the database query?

- **Throughput**
  - How many jobs can the machine run at once?
  - What is the average execution rate?
  - How much work is getting done?

- **If we upgrade a machine with a new processor what do we improve?**

- **If we add a new machine to the lab what do we improve?**

---

### Performance

- **Throughput**
  - How many jobs can the machine run at once?
  - What is the average execution rate?
  - How much work is getting done?

---

**Which of these airplanes has the best performance?**

<table>
<thead>
<tr>
<th>Airplane</th>
<th>Passengers</th>
<th>Range (mi)</th>
<th>Speed (mph)</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 777</td>
<td>375</td>
<td>4630</td>
<td>610</td>
<td>228,750</td>
</tr>
<tr>
<td>Boeing 747</td>
<td>470</td>
<td>4150</td>
<td>610</td>
<td>286,700</td>
</tr>
<tr>
<td>BAC/Sud Concorde</td>
<td>132</td>
<td>4000</td>
<td>1350</td>
<td>178,200</td>
</tr>
<tr>
<td>Douglas DC-8-50</td>
<td>146</td>
<td>8720</td>
<td>544</td>
<td>79,424</td>
</tr>
</tbody>
</table>

- What percentage faster is the Concorde compared to the 747?

- To the DC-8?

- How does throughput of Concorde compare to 747?
Execution Time

- Elapsed Time =
  - a useful number, but often not good for comparison purposes

- CPU time =
  - doesn’t count I/O or time spent running other programs
  - can be broken up into system time, and user time

- Our focus is ?

Book’s Definition of Performance

- For some program running on machine X,
  \[ \text{Performance}_X = \]

  - ”X is n times faster than Y”

- Example:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds
  - How much faster is A than B?
  - (always use “times faster” NOT “10 sec faster”)

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

  \[
  \frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}
  \]

  \[ \text{CPUtime} = \text{CPUClockCycles} \times \text{ClockCycleTime} \]

- Clock “ticks” indicate when to start activities (one abstraction):

- Clock Cycle time =

- Clock rate (frequency) =

What is the clock cycle time for a 200 Mhz. clock rate?

Measuring Execution Time

Example: Some program requires 100 million cycles. CPU A runs at 2.0 GHz. CPU B runs at 3.0 GHz. Execution time on CPU A? CPU B?
How to Improve Performance

- To improve performance (everything else being equal) you can either
  - ______ the # of required cycles for a program, or
  - ______ the clock cycle time or, said another way,
  - ______ the clock rate.

Performance / Clock Cycle Review

1. Performance = 1 / Execution Time = 1 / CPU time

2. How do we compute CPU Time?
   - CPU Time = CPU Clock Cycles * Clock Cycle Time
     \[ \text{seconds} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}} \]

3. How do we get these?
   - Clock Cycle Time = time between ticks (seconds per cycle)
     - Usually a given
     - Or compute from Clock Rate
   - CPU Clock Cycles = # of cycles per program
     - Where does this come from?

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

Cycles Per Instruction (CPI)

- CPI Clock Cycles
  = Total # of clock cycles
  = avg # of clock cycles per instruction * program instruction count
  = CPI * IC

What is CPI?
- Average cycle count of all the instruction executed in the program
- CPI provides one way of comparing 2 different implementations of the same ISA, since the instruction count for a program will be the same

New performance equation:
- Time = Instruction count * CPI * ClockCycleTime
Now that we understand cycles

- A given program will require
  - some number of
  - some number of
  - some number of

- We have a vocabulary that relates these quantities:
  - Instruction count
  - CPU clock cycles (cycles/program)
  - Clock cycle time
  - Clock rate
  - CPI

Performance

- Performance is determined by _____!
- Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second?

- Common pitfall:

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).

  For some program,

  Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

  What machine is faster for this program, and by how much?

# of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

  The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
  The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

  Which sequence will be faster? How much?
  What is the CPI for each sequence?
Evaluating Performance

- Best scenario is head-to-head
  - Two or more machines running the same programs (workload), over an extended time
  - Compare execution time
  - Choose your machine

- Fallback scenario: BENCHMARKS
  - Packaged in ‘sets’
  - Programs specifically chosen to measure performance
    - Programs typical of ___________
      - Composed of real applications
        - Specific to workplace environment
        - Minimizes ability to speed up execution

#### Benchmark Games

- 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 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...

  Saturday, January 6, 1996 New York Times

#### SPEC “results”

- Compiler “enhancements” and performance
SPEC CPU2006 (Integer)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Programming Language</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>480: perfbench</td>
<td>C</td>
<td>Uses the SPEC Perfmark benchmark suite.</td>
</tr>
<tr>
<td>495: djpeg</td>
<td>C+</td>
<td>Uses the SPEC Cjpeg benchmark suite.</td>
</tr>
<tr>
<td>496: gcc</td>
<td>C</td>
<td>Uses the SPEC C/gcc benchmark suite.</td>
</tr>
<tr>
<td>497: gcc</td>
<td>C</td>
<td>Uses the SPEC C/gcc benchmark suite.</td>
</tr>
<tr>
<td>498: gfortran</td>
<td>C</td>
<td>Uses the SPEC C/gfortran benchmark suite.</td>
</tr>
<tr>
<td>499: mips</td>
<td>C</td>
<td>Uses the SPEC C/mips benchmark suite.</td>
</tr>
<tr>
<td>500: mips</td>
<td>C</td>
<td>Uses the SPEC C/mips benchmark suite.</td>
</tr>
<tr>
<td>501: mips</td>
<td>C</td>
<td>Uses the SPEC C/mips benchmark suite.</td>
</tr>
<tr>
<td>502: mips</td>
<td>C</td>
<td>Uses the SPEC C/mips benchmark suite.</td>
</tr>
</tbody>
</table>

Execution Time After Improvement = Execution Time Unaffected + (Execution Time Affected / Amount of Improvement)

**Example:**

Suppose a program runs in 100 seconds on a machine, with multiplication 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?**

**Corollary:** Make the common case fast

SPEC CPU2006 (Floating point)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Programming Language</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>410: bwaves</td>
<td>Fortran</td>
<td>Fluid Dynamics</td>
</tr>
<tr>
<td>416: gessos</td>
<td>Fortran</td>
<td>Quantum Chemistry</td>
</tr>
<tr>
<td>433: mpc</td>
<td>C</td>
<td>Physics/Quantum Chemodynamics</td>
</tr>
<tr>
<td>434: swave</td>
<td>Fortran</td>
<td>Physics/CFD</td>
</tr>
<tr>
<td>455: gromacs</td>
<td>C</td>
<td>Biochemistry/Molecular Dynamics</td>
</tr>
<tr>
<td>456: carbon</td>
<td>C</td>
<td>Physics/General Relativity</td>
</tr>
<tr>
<td>457: wake2d</td>
<td>Fortran</td>
<td>Fluid Dynamics</td>
</tr>
</tbody>
</table>

(plus 10 more...)

Amdahl’s Law

**EX: 1-71...**

**Performance is specific to**

- Only total execution time is a consistent summary of performance

**For a given architecture performance increases come from:**

**Pitfall:** expecting improvement in one aspect of a machine’s performance to proportionally affect the total performance

**You should not always believe everything you read! Read carefully!**