SI232
Slide Set #12: Performance
(Chapter 4)

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?

Which of these airplanes has the best performance?

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

- **What percentage faster is the Concorde compared to the 747?**
  - **To the DC-8?**
- **How does throughput of Concorde compare to 747?**

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?***

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,

  Performance_X =
- "X is n times faster than Y"
- Problem:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds
  - How much faster is A than B?

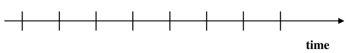
Clock Cycles

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

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

CPUtime = CPUClockCycles x 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

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

CPUtime = CPUClockCycles x ClockCycleTime

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?

Exercise

- **1. Program A runs in 10 seconds on a machine with a 100 MHz clock. How many clock cycles does program A require?**

(extra space)

Exercise

- **2.) Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target?"**
- **3.) Why might machine B need more clock cycles to run the program?**

How to Improve Performance

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

So, to improve performance (everything else being equal) you can either

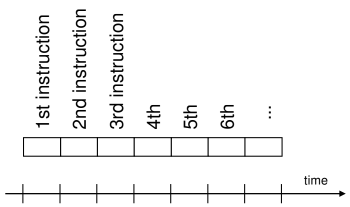
________ the # of required cycles for a program, or

________ the clock cycle time or, said another way,

________ the clock rate.

How many cycles are required for a program?

- **Could assume that # of cycles = # of instructions**



This assumption is...
Why?

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**

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

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?

Cycles Per Instruction (CPI)

CPU 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?

Exercise #1: MIPS

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.
  Compiler #1: code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.
  Compiler #2: code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.
- Which sequence will be faster according to execution time?
- Which sequence will be faster according to MIPS?
  MIPS = Inst. Count / (ExecutionTime * 10^6)

(extra space)

Exercise #2

- Program A runs in 0.34 seconds on a 500 Mhz machine. You know that this program requires 100 million instructions of which:
  - 10% are mult. instructions that take an unknown number of cycle
  - 60% are other arithmetic instructions taking 1 cycle
  - 30% are memory instructions taking 2 cycles
- How many cycles does a multiplication take on this machine?

(extra space)

Exercise #3

- Program A runs in 2 seconds on a certain machine. You know that this program requires 500 million instructions of which:
  - 30% are multiplication instructions that take 10 cycles
  - 40% are other arithmetic instructions taking 1 cycle
  - 30% are memory instructions taking 2 cycles
- Suppose multiplication could be improved to take just 1 cycle. How much faster would the new machine be compared to the old?

(extra space)

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

Benchmarks

Types of Benchmarks used depend on position of development cycle

- Small benchmarks
  - Nice for architects and designers
  - Very small code segments
  - Easy to standardize
  - Can be abused
- SPEC (System Performance Evaluation Cooperative)
  - http://www.specbench.org/
  - Companies have agreed on a set of real program and inputs
  - Valuable indicator of performance (and compiler technology)

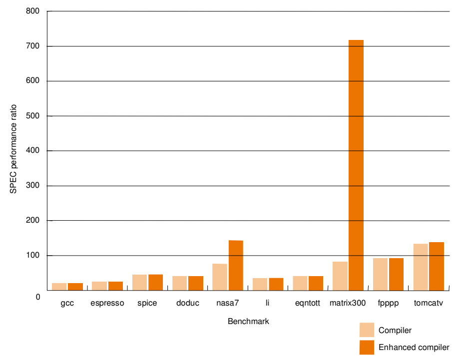
Benchmark Games

- *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 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…*

  Saturday, January 6, 1996 New York Times

SPEC ‘89

- **Compiler “enhancements” and performance**



SPEC CPU2000

Integer benchmarks		FP benchmarks	
Name	**Description**	**Name**	**Type**
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition using neural networks
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and rote simulator	fma3d	Crash simulation using finite-element method
		sixtrack	High-energy nuclear physics accelerator design
		apsi	Meteorology: pollutant distribution

FIGURE 4.5 The SPEC CPU2000 benchmarks. The 12 integer benchmarks in the left half of the table are written in C and C++, while the floating-point benchmarks in the right half are written in Fortran (77 or 90) and C. For more information on SPEC and on the SPEC benchmarks, see www.spec.org. The SPEC CPU benchmarks use wall clock time as the metric, but because there is little I/O, they measure CPU performance.

Amdahl's Law

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 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?**

- ***Corollary: Make the common case fast***

Exercise #1

- **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?**

- **Formula:**
  Time = Exec. Time Unaffected +(Exe. Time Affected / Amount of Improvement)

Exercise #2

- 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?

Remember

- 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!