Performance

ICS 233

Computer Architecture and Assembly Language

Prof. Muhamed Mudawar

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

What is Performance?

- How can we make intelligent choices about computers?
- Why is some computer hardware performs better at some programs, but performs less at other programs?
- How do we measure the performance of a computer?
- What factors are hardware related? software related?
- How does machine's instruction set affect performance?
- Understanding performance is key to understanding underlying organizational motivation

Response Time and Throughput

- ❖ Response Time
  - ✧ Time between start and completion of a task, as observed by end user
  - ✧ Response Time = CPU Time + Waiting Time (I/O, OS scheduling, etc.)
- ❖ Throughput
  - ✧ Number of tasks the machine can run in a given period of time
- ❖ Decreasing execution time improves throughput
  - ✧ Example: using a faster version of a processor
  - ✧ Less time to run a task ⇒ more tasks can be executed
- ❖ Increasing throughput can also improve response time
  - ✧ Example: increasing number of processors in a multiprocessor
  - ✧ More tasks can be executed in parallel
  - ✧ Execution time of individual sequential tasks is not changed
  - ✧ But less waiting time in scheduling queue reduces response time

Book's Definition of Performance

- ❖ For some program running on machine *X*

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

- ❖ *X* is *n* times faster than *Y*

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

What do we mean by Execution Time?

❖ Real Elapsed Time

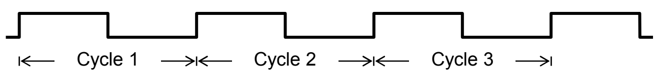
- ✧ Counts everything:
  - Waiting time, Input/output, disk access, OS scheduling, … etc.
- ✧ Useful number, but often not good for comparison purposes

❖ Our Focus: CPU Execution Time

- ✧ Time spent while executing the program instructions
- ✧ Doesn't count the waiting time for I/O or OS scheduling
- ✧ Can be measured in seconds, or
- ✧ Can be related to number of CPU clock cycles

Clock Cycles

❖ Clock cycle = Clock period = 1 / Clock rate



❖ Clock rate = Clock frequency = Cycles per second

- ✧ 1 Hz = 1 cycle/sec   1 KHz = 10^3 cycles/sec
- ✧ 1 MHz = 10^6 cycles/sec   1 GHz = 10^9 cycles/sec
- ✧ 2 GHz clock has a cycle time = $1/(2\times10^9)$ = 0.5 nanosecond (ns)

❖ We often use clock cycles to report CPU execution time

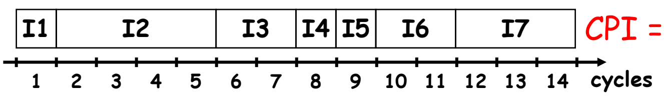
$$\text{CPU Execution Time} = \text{CPU cycles} \times \text{cycle time} = \frac{\text{CPU cycles}}{\text{Clock rate}}$$

Improving Performance

- To improve performance, we need to
  - Reduce number of clock cycles required by a program, or
  - Reduce clock cycle time (increase the clock rate)
- Example:
  - A program runs in 10 seconds on computer *X* with 2 GHz clock
  - What is the number of CPU cycles on computer *X* ?
  - We want to design computer *Y* to run same program in 6 seconds
  - But computer *Y* requires 10% more cycles to execute program
  - What is the clock rate for computer *Y* ?
- Solution:
  - CPU cycles on computer *X* = 10 sec × 2 × 10^9 cycles/s = 20 × 10^9
  - CPU cycles on computer *Y* = 1.1 × 20 × 10^9 = 22 × 10^9 cycles
  - Clock rate for computer *Y* = 22 × 10^9 cycles / 6 sec = 3.67 GHz

Clock Cycles per Instruction (CPI)

- Instructions take different number of cycles to execute
  - Multiplication takes more time than addition
  - Floating point operations take longer than integer ones
  - Accessing memory takes more time than accessing registers
- CPI is an average number of clock cycles per instruction



CPI = 14/7 = 2

- Important point

Changing the cycle time often changes the number of cycles required for various instructions (more later)

Performance Equation

- To execute, a given program will require …
  - Some number of machine instructions
  - Some number of clock cycles
  - Some number of seconds
- We can relate CPU clock cycles to instruction count

CPU cycles = Instruction Count × CPI

- Performance Equation: (related to instruction count)

Time = Instruction Count × CPI × cycle time

Understanding Performance Equation

Time = Instruction Count × CPI × cycle time

	I-Count	CPI	Cycle
Program	X		
Compiler	X	X	
ISA	X	X	X
Organization		X	X
Technology			X

Using the Performance Equation

- Suppose we have two implementations of the same ISA
- For a given program
  - Machine A has a clock cycle time of 250 ps and a CPI of 2.0
  - Machine B has a clock cycle time of 500 ps and a CPI of 1.2
  - Which machine is faster for this program, and by how much?
- Solution:
  - Both computer execute same count of instructions = I
  - CPU execution time (A) = I × 2.0 × 250 ps = 500 × I ps
  - CPU execution time (B) = I × 1.2 × 500 ps = 600 × I ps
  - Computer A is faster than B by a factor = $\frac{600 \times I}{500 \times I} = 1.2$

Determining the CPI

- Different types of instructions have different CPI

  Let CPI_i = clocks per instruction for class *i* of instructions

  Let C_i = instruction count for class *i* of instructions

$$\text{CPU cycles} = \sum_{i=1}^{n} (CPI_i \times C_i)$$

$$CPI = \frac{\sum_{i=1}^{n} (CPI_i \times C_i)}{\sum_{i=1}^{n} C_i}$$

- Designers often obtain CPI by a detailed simulation
- Hardware counters are also used for operational CPUs

Example on Determining the CPI

❖ Problem

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 per instruction, 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

Compute the CPU cycles for each sequence. Which sequence is faster?

What is the CPI for each sequence?

❖ Solution

CPU cycles (1st sequence) = (2×1) + (1×2) + (2×3) = 2+2+6 = 10 cycles

CPU cycles (2nd sequence) = (4×1) + (1×2) + (1×3) = 4+2+3 = 9 cycles

Second sequence is faster, even though it executes one extra instruction

CPI (1st sequence) = 10/5 = 2 CPI (2nd sequence) = 9/6 = 1.5

Second Example on CPI

Given: instruction mix of a program on a RISC processor

What is average CPI?

What is the percent of time used by each instruction class?

$Class_i$	$Freq_i$	CPI_i	$CPI_i \times Freq_i$	%Time
ALU	50%	1	0.5×1 = 0.5	0.5/2.2 = 23%
Load	20%	5	0.2×5 = 1.0	1.0/2.2 = 45%
Store	10%	3	0.1×3 = 0.3	0.3/2.2 = 14%
Branch	20%	2	0.2×2 = 0.4	0.4/2.2 = 18%

Average CPI = 0.5+1.0+0.3+0.4 = 2.2

How faster would the machine be if load time is 2 cycles?

What if two ALU instructions could be executed at once?

MIPS as a Performance Measure

- MIPS: Millions Instructions Per Second
- Sometimes used as performance metric
  - Faster machine ⇒ larger MIPS
- MIPS specifies instruction execution rate

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

- We can also relate execution time to MIPS

$$\text{Execution Time} = \frac{\text{Inst Count}}{\text{MIPS} \times 10^6} = \frac{\text{Inst Count} \times \text{CPI}}{\text{Clock Rate}}$$

Drawbacks of MIPS

Three problems using MIPS as a performance metric

1. Does not take into account the capability of instructions
   - Cannot use MIPS to compare computers with different instruction sets because the instruction count will differ
2. MIPS varies between programs on the same computer
   - A computer cannot have a single MIPS rating for all programs
3. MIPS can vary inversely with performance
   - A higher MIPS rating does not always mean better performance
   - Example in next slide shows this anomalous behavior

MIPS example

- Two different compilers are being tested on the same program for a 4 GHz machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2, and 3 cycles, respectively.
- The instruction count produced by the first compiler is 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- The second compiler produces 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Which compiler produces a higher MIPS?
- Which compiler produces a better execution time?

Solution to MIPS Example

- First, we find the CPU cycles for both compilers
  - CPU cycles (compiler 1) = $(5\times1 + 1\times2 + 1\times3)\times10^9 = 10\times10^9$
  - CPU cycles (compiler 2) = $(10\times1 + 1\times2 + 1\times3)\times10^9 = 15\times10^9$
- Next, we find the execution time for both compilers
  - Execution time (compiler 1) = 10×10^9 cycles / 4×10^9 Hz = 2.5 sec
  - Execution time (compiler 2) = 15×10^9 cycles / 4×10^9 Hz = 3.75 sec
- Compiler1 generates faster program (less execution time)
- Now, we compute MIPS rate for both compilers
  - MIPS = Instruction Count / (Execution Time $\times 10^6$)
  - MIPS (compiler 1) = $(5+1+1) \times 10^9 / (2.5 \times 10^6) = 2800$
  - MIPS (compiler 2) = $(10+1+1) \times 10^9 / (3.75 \times 10^6) = 3200$
- So, code from compiler 2 has a higher MIPS rating !!!

Amdahl's Law

- Amdahl's Law is a measure of Speedup
  - How a computer performs after an enhancement E
  - Relative to how it performed previously

$$\text{Speedup(E)} = \frac{\text{Performance with E}}{\text{Performance before}} = \frac{\text{ExTime before}}{\text{ExTime with E}}$$

- Enhancement improves a fraction *f* of execution time by a factor *s* and the remaining time is unaffected

$$\text{ExTime with E} = \text{ExTime before} \times (f / s + (1 - f))$$

$$\text{Speedup(E)} = \frac{1}{(f / s + (1 - f))}$$

Example on Amdahl's Law

- 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?
- Solution: suppose we improve multiplication by a factor *s*

  25 sec (4 times faster) = 80 sec / *s* + 20 sec

  s = 80 / (25 – 20) = 80 / 5 = 16

  Improve the speed of multiplication by *s* = 16 times
- How about making the program 5 times faster?

  20 sec (5 times faster) = 80 sec / *s* + 20 sec

  s = 80 / (20 – 20) = ∞ Impossible to make 5 times faster!

Benchmarks

- ❖ Performance best obtained by running a real application
  - ✧ Use programs typical of expected workload
  - ✧ Representatives of expected classes of applications
  - ✧ Examples: compilers, editors, scientific applications, graphics, ...
- ❖ SPEC (System Performance Evaluation Corporation)
  - ✧ Funded and supported by a number of computer vendors
  - ✧ Companies have agreed on a set of real program and inputs
  - ✧ Various benchmarks for …

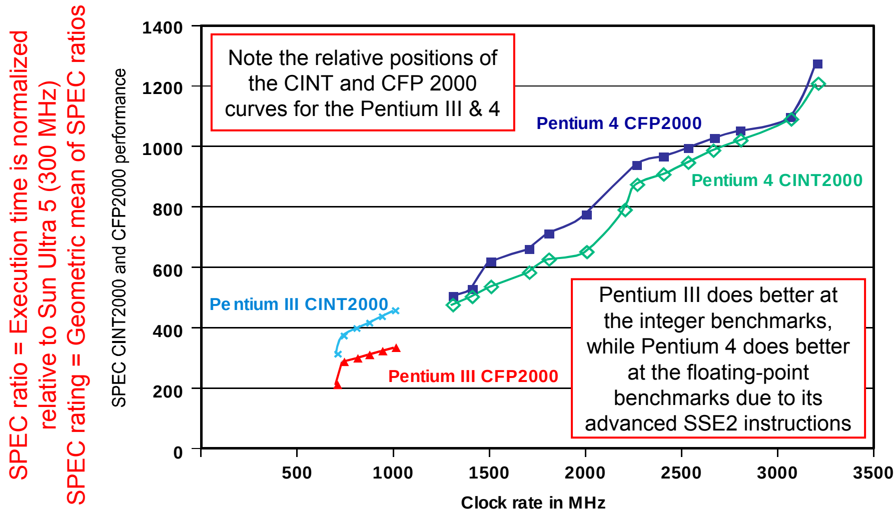
    CPU performance, graphics, high-performance computing, client-server models, file systems, Web servers, etc.
  - ✧ Valuable indicator of performance (and compiler technology)

The SPEC CPU2000 Benchmarks

12 Integer benchmarks (C and C++)		14 FP benchmarks (Fortran 77, 90, and C)	
Name	Description	Name	Description
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA placement and routing	swim	Shallow water model
gcc	GNU C compiler	mgrid	Multigrid solver in 3D potential field
mcf	Combinatorial optimization	applu	Partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Neural networks image recognition
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 route simulator	fma3d	Crash simulation using finite elements
		sixtrack	High-energy nuclear physics
		apsi	Meteorology: pollutant distribution

- ❖ Wall clock time is used as metric
- ❖ Benchmarks measure CPU time, because of little I/O

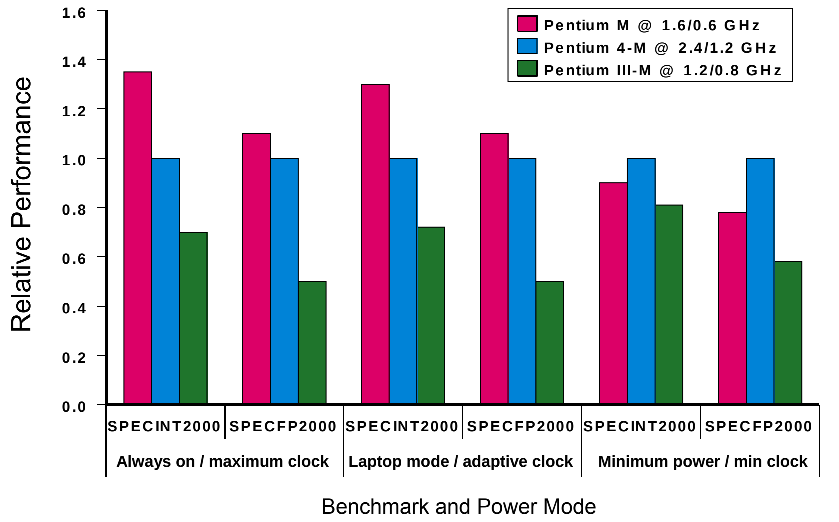
SPEC 2000 Ratings (Pentium III & 4)



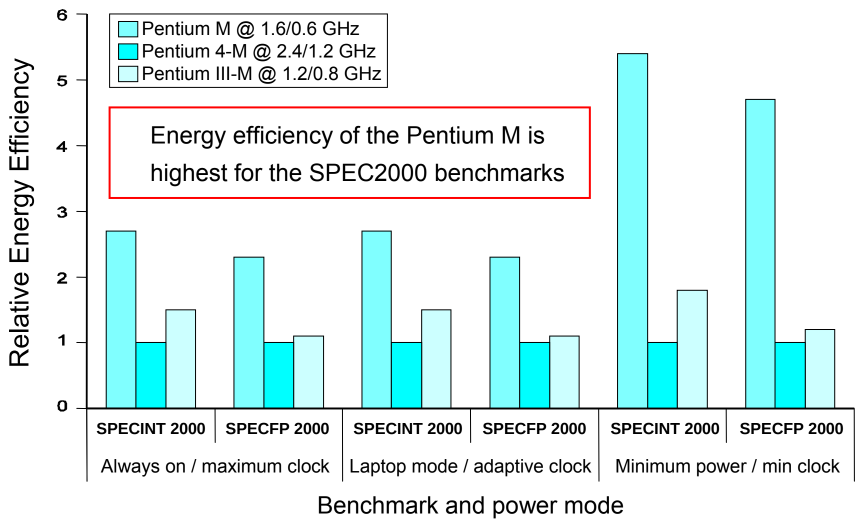
Performance and Power

- Power is a key limitation
  - Battery capacity has improved only slightly over time
- Need to design power-efficient processors
- Reduce power by
  - Reducing frequency
  - Reducing voltage
  - Putting components to sleep
- Energy efficiency
  - Important metric for power-limited applications
  - Defined as performance divided by power consumption

Performance and Power



Energy Efficiency



Things to Remember

- Performance is specific to a particular program
  - Any measure of performance should reflect execution time
  - Total execution time is a consistent summary of performance
- For a given ISA, performance improvements come from
  - Increases in clock rate (without increasing the CPI)
  - Improvements in processor organization that lower CPI
  - Compiler enhancements that lower CPI and/or instruction count
  - Algorithm/Language choices that affect instruction count
- Pitfalls (things you should avoid)
  - Using a subset of the performance equation as a metric
  - Expecting improvement of one aspect of a computer to increase performance proportional to the size of improvement