

Chapter 4

Assessing and Understanding Performance

Fall 2005

Department of Computer Science

Kent State University

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?

Defining Performance

Case Study: Airplane

Airplane Passengers Range (mi) Speed (mph) Throughput

Boeing 777	375	4630	610	288,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

- The 747 carries the most Passengers
- DC-8 has the longest range
- Concorde has the highest speed
- The 777 has the highest throughput

Which airplane performs the best?

The answer depends on how performance is measured

Defining Performance

Computer Systems

- Case 1: Individual Computer Users
 - Response Time (latency)
 - How long does it take to execute my job (**Executing Time**)?
 - How long must I wait for the database query?
- Case 2: Data Center, Switching Systems
 - Throughput – (Total amount work done in a given **Time**)
 - How many concurrent jobs can the machine run in a given Time period?
 - How many subscriber calls can the switch handle without dropping the
- *Need different performance metrics*

Computer Performance: TIME, TIME, TIME

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

Execution Time

- Elapsed Time
 - counts everything (*disk and memory accesses, I/O , etc.*)
 - 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: user CPU time
 - time spent executing the lines of code that are "in" our program

Computer Performance Definition

Response Time

Current Focus : Response Time

- To Maximize Performance →

Minimize Response Time (**Execution Time**):

$$\text{Computer X:} \quad \text{Performance}_x = \frac{1}{\text{Executing Time}_x}$$

$$\text{Computer Y:} \quad \text{Performance}_y = \frac{1}{\text{Executing Time}_y}$$

What if $\text{Performance}_x > \text{Performance}_y$?

Relative Performance

For some program executing on computer X:

If "X is n times faster than Y"

$$\rightarrow \frac{\text{Performance}_x}{\text{Performance}_y} = n$$

Assume computer X runs a program in 10 seconds while computer Y takes 15 seconds to run the same program. Then computer X has better performance than computer Y.

How much better?

$$\frac{\text{Performance}_x}{\text{Performance}_y} = \frac{\text{Execution Time}_y}{\text{Execution Time}_x} = \frac{15 \text{ seconds}}{10 \text{ seconds}} = 1.5$$

Performance Metrics

End-User Perspective

- Elapsed Time (Response Time or Wall-Clock Time)
 - Total time to complete a task
 - Disk and Memory Access, I/O OS overhead, CPU execution time etc)
- CPU Execution Time
 - Actual time CPU spends computing for a specific task
 - System CPU Time (Time spent in OS on behalf of your program)
 - User CPU Time (Time spent in executing lines of code inside your program)
 - Does not count I/O or time spent running other programs

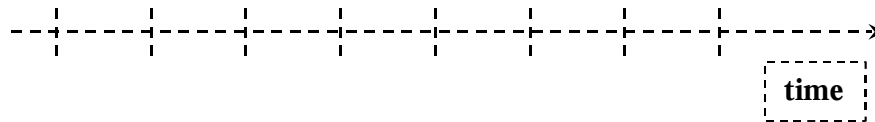
System Performance ~ Refers to Elapsed Time for an unloaded system

CPU Performance ~ Refers to **User CPU time** (This is our primary focus)

Performance Metrics

Hardware Perspective

- Designers measure hardware performance via clock **cycles**
- Clock “ticks” indicate when to schedule events:



- Clock runs at a constant rate
- **cycle time** = time between ticks = **seconds per cycle**
- clock rate (frequency) = **cycles per second** (1 Hz. = 1 cycle/sec)
- Therefore: cycle time = 1/clock rate

A 4 Ghz. clock has a $\frac{1}{4 \times 10^9} \times 10^{12} = 250$ picoseconds (ps) **cycle time**

How to Improve Performance

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

So, to improve performance (everything else being equal) you can either (increase or decrease?)

_____ the # of required cycles for a program, or

_____ the clock cycle time or, said another way,

_____ the clock rate.

Improving Performance

Example: Favorite program

Our favorite program runs in 10 seconds on computer A, which has a 4 GHz. 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?"

Improving Performance

Example: Favorite program

$$\text{CPU Time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{4 \times 10^9 \frac{\text{cycles}}{\text{sec}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 4 \times 10^9 \frac{\text{cycles}}{\text{sec}} = 40 \times 10^9 \frac{\text{cycles}}{\text{sec}}$$

$$\text{CPU Time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

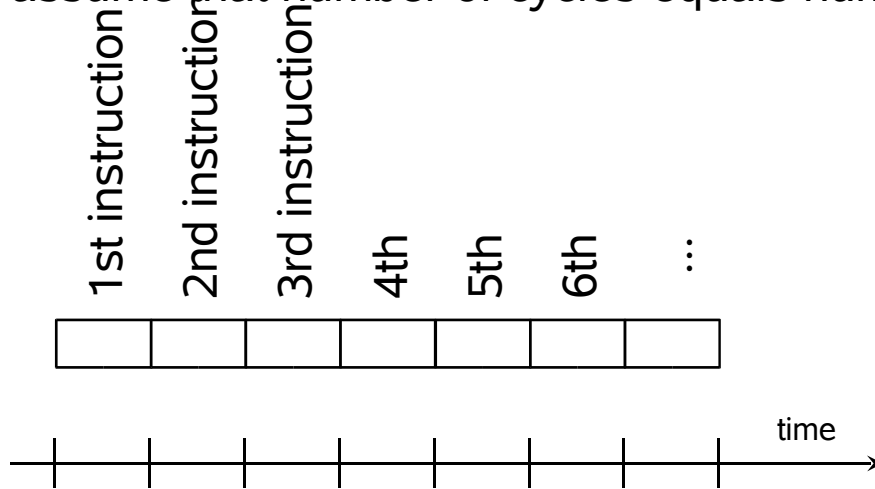
$$6 \text{ seconds} = \frac{1.2 \times 40 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 40 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{8 \times 10^9 \text{ cycles}}{\text{seconds}} = 8 \text{ GHz}$$

Computer B must have twice clock rate of A to run program in 6 seconds

How many cycles are required for a program?

- Could assume that number of cycles equals number of instructions

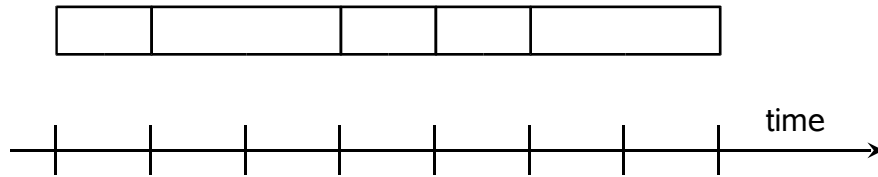


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- *Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)*

Now that we understand cycles

- A given program will require
 - some number of instructions (machine instructions)
 - some number of cycles
 - some number of seconds
- We have a vocabulary that relates these quantities:
 - cycle time (seconds per cycle)
 - clock rate (cycles per second)
 - CPI (cycles per instruction)
 - a floating point intensive application might have a higher CPI*
 - MIPS (millions of instructions per second)
 - this would be higher for a program using simple instructions*

Performance

- Performance is determined by execution time
- 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: thinking one of the variables is indicative of performance when it really isn't.

More on Clock cycles in a prog.

CPI

$$\begin{aligned} \text{CPU time} &= \text{Instructions per program} \times \text{Average Clock cycles Per instruction} \\ &= \text{Instructions per program} \times \text{“Clock cycles Per Instruction” (CPI)} \\ &= \text{Instruction Count} \times \text{CPI} \end{aligned}$$

CPI is an average number of clock cycles for all instructions executed in a program

CPU Performance Equation

CPI

Recall:

$$\text{End-User CPU time in program} = \text{Clock cycle time} \times \text{Clock cycle in program}$$

$$\text{End-User CPU time in program} = \text{Clock cycle time} \times \text{Instruction Count} \times \text{CPI}$$

or

$$\text{End-User CPU time in program} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}}$$

CPU Performance and its Factors

... so far: new units of measure

$$\begin{aligned} \text{Time} &= \frac{\text{seconds}}{\text{program}} = \frac{\text{seconds}}{\text{cycle}} \times \frac{\text{cycle}}{\text{program}} \\ &= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} \end{aligned}$$

CPU Execution time
for a program

Instruction count

Avg. CPI

Clock cycle time

How do we measure these performance factors?
(We will talk about this shortly)

CPU Performance Factors

..Measure

CPU Execution time
for a program

By running the program

Clock cycle time

Published as part of the computer documentation
(clock rate)

Instruction count

Using Simulators of architecture, Hardware counters

Avg. CPI

Using detailed simulation of the implementation,
hardware counters

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 250 ps and a CPI of 2.0

Machine B has a clock cycle time of 500 ps and a CPI of 1.2

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

- *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

CPI Example

Same ISA

- Each computer executes same number of instructions (I) for the program

$$\begin{aligned} \text{CPU time in A} &= \text{Clock cycle time} \times \text{Instruction Count} \times \text{CPI} \\ \bullet \text{ CPU time}_A &= 250\text{ps} \times I \times 20 = 500 \times I \text{ ps} \\ \bullet \text{ CPU time}_B &= 500\text{ps} \times I \times 1.2 = 600 \times I \text{ ps} \end{aligned}$$

- Hence Computer A is faster by:

$$\begin{aligned} \frac{\text{CPU performance}_A}{\text{CPU performance}_B} &= \frac{\text{Execution time}_B}{\text{Execution time}_A} \\ &= \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2 \end{aligned}$$

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?

MIPS example

- Two different compilers are being tested for a 4 GHz. 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.

The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

The second compiler's 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 MIPS?
- Which sequence will be faster according to execution time?

Remember

- Performance is specific to a particular program/s
 - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
 - increases in clock rate (without adverse CPI affects)
 - improvements in processor organization that lower CPI
 - compiler enhancements that lower CPI and/or instruction count
 - Algorithm/Language choices that affect instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance