Chapter One, Again (1/2)

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?

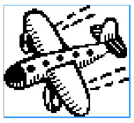
  What factors of system performance are hardware related?
  (e.g., Do we need a new machine, or a new operating system?)

  How does the machine's instruction set affect performance?

Which of these airplanes has the best performance?

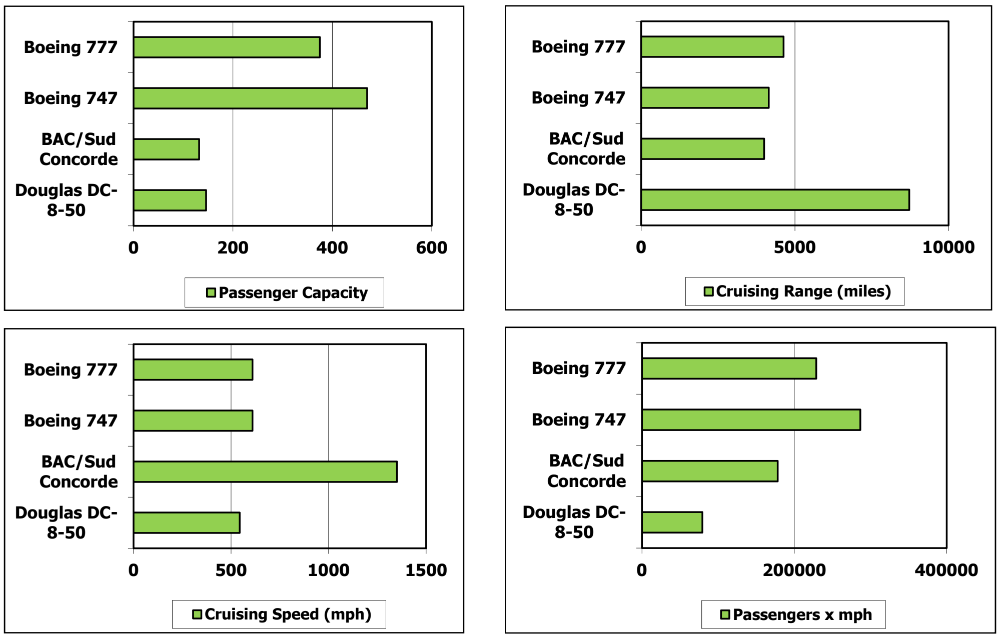
Airplane	Passengers (n)	Range (mi)	Speed (mph)	n x mph
Boeing 777	375	4630	610	228750
Boeing 747	470	4150	610	286700
BAC/Sud Concorde	132	4000	1350	178200
Douglas DC-8-50	146	8720	544	79424

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?



Defining Performance

- Which airplane has the best performance?

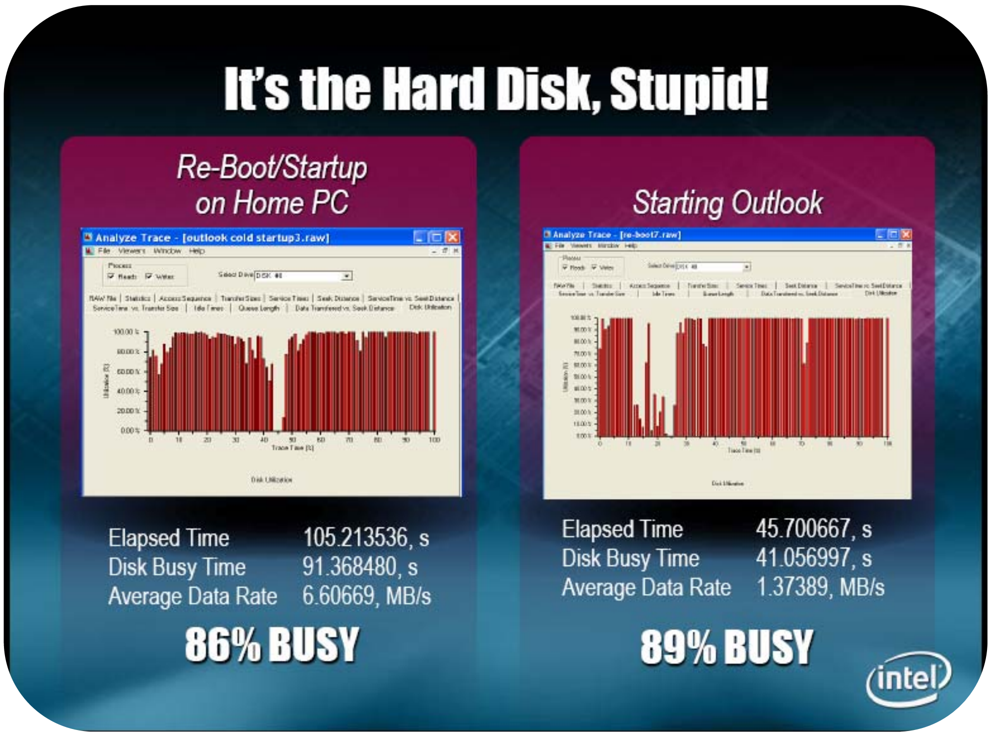


Computer Performance: TIME, TIME, TIME

- Response Time (latency)
  - — The time between the start and completion of a task
  - — 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
  - — The total amount of work done in a given time
  - — 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?*

Response Time Matters


It's the Hard Disk, Stupid!
Re-Boot/Startup on Home PC
Starting Outlook
Analyze Trace - [outlook cold startup3.raw]
Analyze Trace - [re-boot7.raw]
86% BUSY
89% BUSY
intel

	Re-Boot/Startup on Home PC	Starting Outlook
Elapsed Time	105.213536, s	45.700667, s
Disk Busy Time	91.368480, s	41.056997, s
Average Data Rate	6.60669, MB/s	1.37389, MB/s

Justin Rattner's ISCA'08 Keynote (VP and CTO of Intel)

Execution Time

- Elapsed Time (Response 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
- For example, a program may have a system CPU time of 22 sec., a user CPU time of 90 sec., a CPU execution time of 112 sec., and a response time of 162 sec..

Simple Definition of Performance

- Performance is in units of things-per-second; bigger is better
- For some program running on machine X,

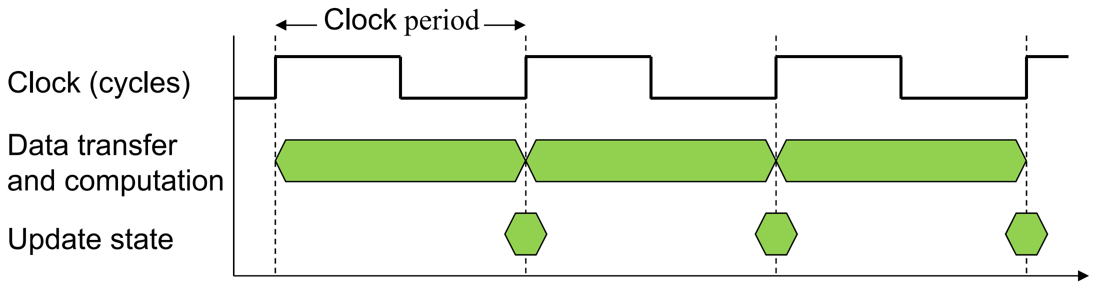
  $\text{Performance}_X = 1 / \text{Execution time}_X$
- "X is n times faster than Y"

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

- When is throughput more important than execution time?
- When is execution time more important than throughput?
- Example: Performance?
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock
- A computer clock runs at a constant rate and determines when to start activities in hardware.



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

How to Improve Performance

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

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

- To improve performance (everything else being equal) you can either

$\underline{\quad\downarrow\quad}$ the # of required cycles for a program, or
$\underline{\quad\downarrow\quad}$ the clock cycle time or, said another way,
$\underline{\quad\uparrow\quad}$ the clock rate.

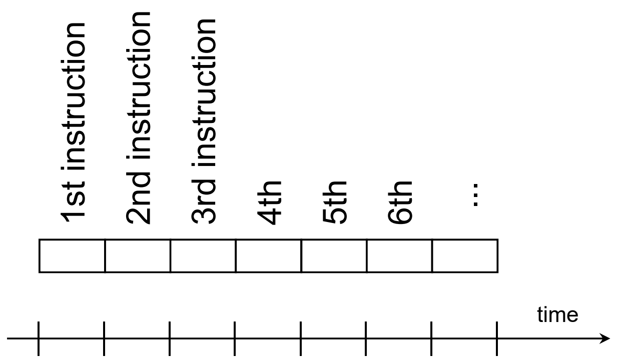
Understanding Performance

- How are the following likely to effect response time and throughput?
  - Increasing the clock speed of a given processor.
  - Increasing the number of jobs in a system (e.g., having a single computer service multiple users).
  - Increasing the number of processors in a system that uses multiple processors (e.g., a network of ATM machines).
- If an Pentium III runs a program in 8 seconds and a PowerPC runs the same program in 10 seconds, how many times faster is the Pentium III?

  n = 10 / 8 = 1.25 times faster (or 25% faster)
- Why might someone chose to buy the PowerPC in this case?

How many cycles are required for a program?

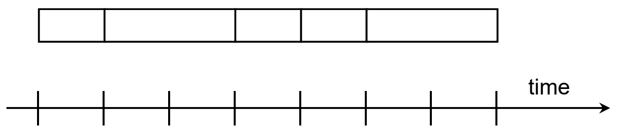
- Could assume that # of cycles = # of instructions?



This assumption is incorrect, because 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)***

Example

- **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?"**

- **Don't Panic, can easily work this out from basic principles**

Solution

- **Old computer**
  - **Execution time**
    - T_{old} = second/program = cycle/program × second/cycle = 10
  - **Clock rate**
    - f_{old} = cycle/second = 400 Mhz
  - **Execution cycles**
    - cycle/program = 10 × 400M
- **New computer**
  - **Execution time**
    - $T_{new} = 1.2 \times \text{cycle/program} \times 1/f_{new}$
      $= 1.2 \times 10 \times 400\text{M} \times 1/f_{new} = 6$
  - **Clock rate**
    - $f_{new} = 12/6 \times 400\text{M} = 800\text{Mhz}$

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

Revised Performance Equation

Initially, the performance is represented by

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

Cycles / Program can be further divided into two components:

$$\frac{\text{Cycles}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}}$$

The performance is now represented by

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

Execution Time = Program Count × CPI × Clock Cycle Time

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$
$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$
$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- **Instruction Count for a program**
  - **Determined by program, ISA and compiler**
- **Average cycles per instruction**
  - **Determined by CPU hardware**
  - **If different instructions have different CPI**
    - Average CPI affected by instruction mix

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.

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?

- *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?*

Solution

- Execution Time = Program Count x CPI x Clock Cycle Time
  - T_A = Program Count x 2 x 10 ns = PC x 20 ns
  - T_B = Program Count x 1.2 x 20 ns = PC x 24 ns

- Same ISA implies same program count for a given program

- Performance Comparison
  - n = Performance A ÷ Performance B = T_B/T_A = 24/20 = 1.2
  - Computer A is 1.2 times faster than Computer B

Non-uniform CPI

- Instructions may require different CPI for execution.
- It is possible to compute the CPU clock cycles by looking at the different types of instructions and using their individual clock cycle counts.

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

CPU Clock Cycles with uniform CPI

$$\frac{\text{Seconds}}{\text{Program}} = \sum_{i=0}^{n} \left(\frac{\text{Instructions}(i)}{\text{Program}} \times \frac{\text{Cycles}(i)}{\text{Instruction}} \right) \times \frac{\text{Seconds}}{\text{Cycle}}$$

CPU Clock Cycles with non-uniform CPI

# 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?

Solution

- 1st code
  - # of cycles: $2\times1 + 1\times2 + 2\times3 = 10$
  - # of instructions: $2 + 1 + 2 = 5$
  - CPI = 10/5 = 2
- 2nd code
  - # of cycles: $4\times1 + 1\times2 + 1\times3 = 9$
  - # of instructions: $4 + 1 + 1 = 6$
  - CPI = 9/6 = 1.5
- Performance comparison
  - $N = T_1/T_2 = 10/9 = 1.1$
  - Code 2 is 1.1 time faster than Code 1

Factors affecting CPU Performance

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- **Which factors are affected by each of the following?**

	instr. count	CPI	clock rate
Program	O	O	
Compiler	O	O	
Instr. Set Arch.	O	O	O
Organization		O	O
Technology		O	O

Execution Time = Program Count × CPI × Clock Cycle Time

Poor Performance Metrics

- **Marketing metrics for computer performance included MIPS and MFLOPS**
- **MIPS : millions of instructions per second**
  MIPS = instruction count / (execution time x 10^6)
  - **For example, a program that executes 3 million instructions in 2 seconds has a MIPS rating of 1.5**
  - **Advantage : Easy to understand and measure**
  - **Disadvantages : May not reflect actual performance, since simple instructions do better.**
- **MFLOPS : millions of floating point operations per second**
  MFLOPS = floating point operations / (execution time x 10^6)
  - **For example, s program that executes 4 million instructions in 5 seconds has a MFLOPS rating of 0.8**
  - **Advantage : Easy to understand and measure**
  - **Disadvantages : Same as MIPS,only measures floating point**

MIPS example

- **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.**

  The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 2 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?**

Solution

- **1st compiler**
  - **# of cycles: $5\times1 + 1\times2 + 2\times3 = 13$ Millions**
  - **# of instructions: $5 + 1 + 2 = 8$ Millions**
  - **MIPS = $8/13 \times 100$ M = 61.5**
- **2nd compiiler**
  - **# of cycles: $10\times1 + 1\times2 + 1\times3 = 15$ Millions**
  - **# of instructions: $10 + 1 + 1 = 12$ Millions**
  - **MIPS = $12/15 \times 100$ M = 80**
- **Performance comparison**
  - **By MIPS: Compiler 1 < Compiler 2**
  - **By Execution time: Compiler 1 > Compiler 2**