Chapter 1
Computer Abstractions and Technology

1.6 Performance

- **Measure, Report, and Summarize**
- **Make intelligent choices**
- **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

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)	pass * speed
Boeing 737-100	101	630	598	60398
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?**

Continue

Computer Performance: TIME, TIME, TIME

- **Response Time (latency) also referred as execution time**
  - **— 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?**

Question?

- ***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?***

Continue

- **To maximize performance ➔ minimize response/execution time:**

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

- **If the performance of computer X is greater than Y, we have:**

$$\text{Performace}_X > \text{Performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

- **We will use phrase: "X is *n* times faster than Y" to mean:**

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

- **If X is *n* times faster than Y, then execution of Y is *n* longer than X:**

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

Example: Relative Performance

Computer A runs a program in 10 seconds and B runs it in 15 seconds, how much faster is A than B?

A is *n* times faster than B if:

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Thus the performance ratio is:

$$\frac{15}{10} = 1.5$$

and A is therefore 1.5 times faster than B

We could also say that computer B is 1.5 times slower than computer A.

Example: Relative Performance

Computer C's performance is 4 times better than the performance of B, which runs a given application in 28 seconds. How long will computer C take to run that application?

$$\frac{\text{Performance}_C}{\text{Performance}_B} = 4 = \frac{\text{Execution time}_B}{\text{Execution time}_C} = \frac{28}{x} = 4$$

$$x = \frac{28}{4} = 7$$

Measuring Performance

- **Response time**, or **Elapsed time**
  - counts everything *(disk and memory accesses, I/O , etc.)*
  - a useful number, but often not good for comparison purposes

However, a processor may work on several programs simultaneously, in such cases, the system may try to optimize throughput rather than elapsed. Thus use CPU time instead of Elapsed time.

- **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
- How fast the **hardware**?
  - **clock cycles**
  - **clock period**

2.2 CPU Performance and Its Factors

CPU execution time = CPU Clock cycles × Clock cycle time
for a program for a program

Clock cycle and clock rate are inverses, thus:

$$\text{CPU execution time} = \frac{\text{CPU Clock cycles for a program}}{\text{Clock rate}}$$

- **Improve performance by reducing either the length of the clock cycle or the number of clock cycles.**
- **However, there is a trade-off between the number of clock cycles needed and the length of each cycle.**

Example: Improving Performance

- **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?"**

Continue

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

$$10\,\text{seconds} = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

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

CPU time for B can be found using this equation:

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

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

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

Basic Performance Equation

CPU execution time = CPU Clock cycles × Clock cycle time
for a program for a program

CPU Clock cycles = Instruction count (I_c) × clock cycles per instruction (CPI)

CPU time = I_c × CPI × Clock cycle time

or

$$\text{CPU time} = \frac{I_c \times \text{CPI}}{\text{Clock rate}}$$

Example: Using the Performance Equation

We have two implementations:

Computer A: clock cycle time = 250 ps & CPI = 2.0

Computer B: clock cycle time = 500 ps & CPI = 1.2

Which one is faster, and by how much?

For Computer A:

CPU time$_A$ = CPU clock cycles$_A$ × Clock cycle time$_A$

= I_c × 2.0 ×250 ps = 500 × I_c ps

For Computer B:

CPU time$_B$ = I_c × 1.2 ×500 ps = 600 × I_c ps

➔ **Clearly, computer A is faster, The amount faster is:**

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I_c}{500 \times I_c} = 1.2$$

Continue

- **How can we determine the value of these factors:**

  CPU execution time, clock cycle time, I_c, and CPI

- **It is possible to compute the CPU clock cycles by looking at the different class of instructions:**

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

Example: Comparing Code Segments

A compiler designer is trying to decide between two code sequences, given by the hardware designers the following facts:

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

The compiler writer is considering two code that require the following I_c:

Code sequence	Instructions counts for instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code executes the most instruction? Which will be faster? What is CPI?

continue

Sequence 1 executes 2+1+2= 5 instructions, Sequence 2 execute 4+1+1= 6 instructions. So sequence 1 executes fewer instructions.

We can use the following equation to find the total number of clock cycles:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

This yields

CPU clock cycles$_1$ = (2×1)+(1 ×2)+(2 ×3)=2+2+6 = 10 cycles

CPU clock cycles$_2$ = (4×1)+(1 ×2)+(1 ×3)=4+2+3 = 9 cycles

➔ sequence 2 is faster.

The CPI values can be computed by:

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

1.3 Evaluating Performance

- Workload
- Benchmarks
- Benchmarks ➔ small code ➔ encourage optimization
- Benchmarks ➔ highly optimized code ➔ erroneous optimizations

- Suitable benchmarks ➔ obtained performance measurement ➔ write a performance report (reproducibility)

- SPEC benchmarks (system performance evaluation cooperative)

Comparing and Summarizing Performance

- How to summarize a set of measurements results?

	Computer A	Computer B
Program 1 (seconds)	1	10
Program 2 (seconds)	1000	100
Total time (seconds)	1001	110

- The following statements are true:
  - A is 10 times faster than B for program 1.
  - B is 10 times faster than A for programs 2.

Total Execution Time: A Consistent Summary Measure

$$\frac{Performanœ_B}{Performanœ_A} = \frac{Executiontime_A}{Executiontime_B} = \frac{1001}{110} = 9.1$$

That is, B is 9.1 times as fast as A for programs 1 and 2 together.

1.5 Fallacies and Pitfalls

- ***Pitfall: Expecting the improvement of one aspect of a computer to increase performance by an amount proportional to the size of the improvement.***

Example 1:

Suppose a program runs in 100 seconds on a computer, with multiply operations responsible for 80 seconds of this time. How much do I have to improve the speed of multiplication if I want my program to run five times faster?

Amdahl's law:

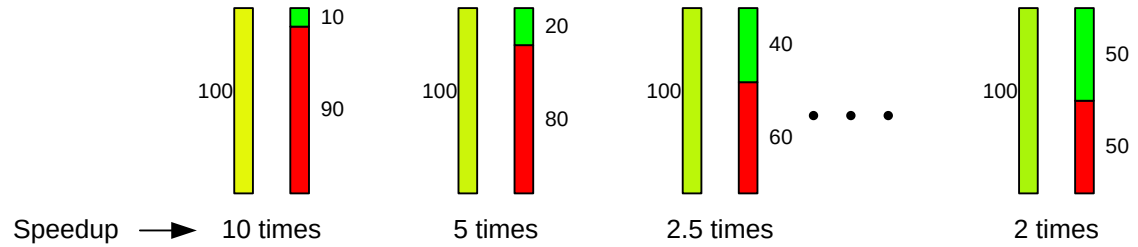
$$Execution\,time\,after\,improvement = \frac{Execution\,time\,affected\,by\,improvement}{amount\,of\,improvement} + Execution\,time\,unaffected$$

$$exe_{new} = \frac{exe_{affected}}{sp_{affected}} + exe_{unaffected}$$

$$20 = \frac{80}{sp_{affected}} + 20$$

$$0 = \frac{80}{sp_{affected}} \qquad \Rightarrow sp_{affected} \to \infty$$

Amdahl's law



Example 2:

Floating Point instructions improved to run twice faster; but only 10% of actual instructions are FP.

$$exe_{new} = \frac{0.1}{2} + 0.9 = 0.95$$

$$SP_{overall} = \frac{exe_{old}}{exe_{new}} = \frac{1}{0.95} = 1.053$$

$$exe_{new} = \frac{exe_{affected}}{sp_{affected}} + exe_{unaffected}$$

$$exe_{new} = exe_{old}\left[\frac{exe_{affected}}{sp_{affected}} + exe_{unaffected}\right]$$

$$SP_{overall} = \frac{exe_{old}}{exe_{new}} = \frac{1}{\frac{exe_{affected}}{sp_{affected}} + exe_{unaffected}}$$

Amdahl's law

Example 3:

Implementation of floating-point (FP) square root, where square root is responsible for 20% of execution time:

1. **Add FPSQR hardware → speed up this operation by factor of 10**
2. **Make all FP instruction faster; FP instructions responsible for 50% of execution time → make all FP instruction run 2 times faster**

What is your choice?

$$SP_{overall} = \frac{exe_{old}}{exe_{new}} = \frac{1}{\frac{exe_{affected}}{sp_{affected}} + exe_{unaffected}}$$

$$SP_{overall} = \frac{1}{\frac{0.2}{10} + 0.8} = \frac{1}{0.82} = 1.22$$

$$SP_{overall} = \frac{1}{\frac{0.5}{2} + 0.5} = \frac{1}{0.75} = 1.33$$

better

Fallacies and Pitfalls (Continue)

- ***Pitfall: Using a subset of the performance equation as a performance metric.***
- **MIPS: as performance metric**
  - **Microprocessor without Interlocking Pipeline Stages**
  - **Million instruction per second**

$$MIPS = \frac{Instruction\ count}{Execution\ time \times 10^6}$$

- **MIPS can fail to give a true picture of performance**

Example: suppose we obtain the following data:

Code from	Instructions counts (in billions) for each instruction class		
	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

Assume that the computer's clock rate is 4 GHz. Which code sequence will execute faster according to MIPS? According to execution time?

Continue

$$\text{Execution time} = \frac{\text{CPU clock cycles}}{\text{Clock rate}}$$

CPU clock cycles$_1$ = (5×1 + 1×2 + 1×3) ×10^9= 10 ×10^9

CPU clock cycles$_2$= (10×1 + 1×2 + 1×3) ×10^9= 15 ×10^9

Now, we find the execution time for the two compilers:

$$Execution\ time_1 = \frac{10\times10^9}{4\times10^9} = 2.5 \text{ seconds}$$

better

$$Execution\ time2 = \frac{15\times10^9}{4\times10^9} = 3.75 \text{ seconds}$$

Now, Let's computes the MIPS rate for each version using MIPS:

$$MIPS = \frac{Instruction\ count}{Execution\ time\ \times10^6}$$

$$MIPS_1 = \frac{(5+1+1)\times10^9}{2.5\times10^6} = 2800$$

$$MIPS_2 = \frac{(10+1+1)\times10^9}{3.75\times10^6} = 3200$$

better