

20MCA103 DFCA

PERFORMANCE

- Understanding how best to measure performance and the limitations of performance measurements is important in selecting a computer.
- If you were running a program on two different desktop computers, the faster one is the desktop computer that gets the job done first.
- If you were running a datacenter that had several servers running jobs submitted by many users, the faster computer was the one that completed the most jobs during a day.
- As an individual computer user, you are interested in reducing computer user, you are interested in reducing
- **response time**—the time between the start and

- **response time**—the time between the start and completion of a task—also referred to as **execution time**
- **throughput or bandwidth**—the total amount of work done in a given time.

- Do the following changes to a computer system increase throughput, decrease response time, or both?
- 1. Replacing the processor in a computer with a faster version
- 2. Adding additional processors to a system that uses multiple processors
- Decreasing response time almost always improves throughput. Hence, in case
- 1, both response time and throughput are improved. In case 2, no one task gets work done faster, so only throughput increases.
- But it would reduce the waiting time in the queue. Thus, in many real computer systems, changing either execution time or throughput often affects the other.

- To maximize performance, we want to minimize response time or execution time for some task.
- Thus, we can relate performance and

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

This means that for two computers X and Y, if the performance of X is greater than the performance of Y, we have

$$\begin{aligned} \text{Performance}_X &> \text{Performance}_Y \\ \frac{1}{\text{Execution time}_X} &> \frac{1}{\text{Execution time}_Y} \\ \text{Execution time}_Y &> \text{Execution time}_X \end{aligned}$$

That is, the execution time on Y is longer than that on X, if X is faster than Y.

In discussing a computer design, we often want to relate the performance of two different computers quantitatively. We will use the phrase “X is n times faster than Y”—or equivalently “X is n times as fast as Y”—to mean

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

If X is n times as fast as Y, then the execution time on Y is n times as long as it is on X:

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

Relative Performance

If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

We know that A is n times as fast as 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 as fast as B.

- Because performance and execution time are reciprocals, increasing performance requires decreasing execution time.
- To avoid the potential confusion between the terms *increasing* and *decreasing*, we usually
say “improve performance” or “improve execution time” when we mean “increase performance” and “decrease execution time.”

- **Program *execution time*** is measured in seconds per program. However, time can be defined in different ways, depending on what we count.
- The most straightforward definition of time is called ***wall clock time, response time, or elapsed time***. These terms mean the total time to complete a task, including disk accesses, memory accesses, *input/output* (I/O) activities, operating system overhead everything.
- Computers are often shared and a processor may work on several programs simultaneously. In such cases, the system may try to optimize throughput rather than attempt to minimize the elapsed time for one program.
- **CPU execution time** or simply **CPU time**, is the time the CPU spends computing for this task and does not include time spent waiting for I/O or running other programs.
- CPU time can be further divided into the CPU time spent in the program, called **user CPU time**, and the CPU time spent in the operating system performing

- *System performance* is used to refer to elapsed time on an unloaded system and *CPU performance* to refer to user CPU time
- All computers are constructed using a clock that determines when events take place in the hardware. These discrete time intervals are called **clock cycles** (or **clock periods**).
- Designers refer to the length of a **clock period** both as the time for a complete *clock cycle* (e.g., 250 picoseconds, or 250 ps) and as the *clock rate* (e.g., 4 gigahertz, or 4 GHz), which is the inverse of the clock period

- **CPU Performance and Its Factors.**
- Users and designers often examine performance using different metrics.
- CPU performance measure is CPU execution time. A simple formula relates the most basic metrics (clock cycles and clock cycle time) to CPU time:

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

Alternatively, because clock rate and clock cycle time are inverses,

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

Improving Performance

Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

Let's first find the number of clock cycles required for the program on A:

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

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \frac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

CPU time for B can be found using this equation:

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

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Instruction Performance

- The performance equations above did not include any reference to the number of instructions needed for the program.
- The execution time must depend on the number of instructions in a

program. One way to think about execution time is that it equals the

number of instructions executed multiplied by the average time per instruction.

- Therefore, the number of clock cycles required for a program can be written as

CPU clock cycles = Instructions for a program x Average clock cycles

per instruction

- The term **clock cycles per instruction**, which is the average number of clock cycles each instruction takes to execute, is often abbreviated as **CPI**.
- Since different instructions may take different amounts of time depending on what they do, CPI is an average of all the instructions executed in the program.
- CPI provides one way of comparing two different implementations

- **Using the Performance Equation**
- Suppose we have two implementations of the same instruction set architecture.
- Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program,
- and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?

•

We know that each computer executes the same number of instructions for the program; let's call this number I . First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$

$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time} \\ &= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}\end{aligned}$$

Likewise, for B:

$$\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

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

We can conclude that computer A is 1.2 times as fast as computer B for this program.

•

The Classic CPU Performance Equation

We can now write this basic performance equation in terms of **instruction count** (the number of instructions executed by the program), CPI, and clock cycle time:

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

or, since the clock rate is the inverse of clock cycle time:

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

These formulas are particularly useful because they separate the three key factors that affect performance. We can use these formulas to compare two different implementations or to evaluate a design alternative if we know its impact on these three parameters.

Figure 1.15 shows the basic measurements at different levels in the computer and what is being measured in each case. We can see how these factors are combined to yield execution time measured in seconds per program:

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

Always bear in mind that the only complete and reliable measure of computer performance is time. For example, changing the instruction set to lower the instruction count may lead to an organization with a slower clock cycle time or higher CPI that offsets the improvement in instruction count. Similarly, because CPI depends on type of instructions executed, the code that executes the fewest number of instructions may not be the fastest.

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

FIGURE 1.15 The basic components of performance and how each is measured.

•

The performance of a program depends on the algorithm, the language, the compiler, the architecture, and the actual hardware. The following table summarizes how these components affect the factors in the CPU performance equation.

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more divides, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.

