1.10 Fallacies and pitfalls

This section has been set as optional by your instructor.

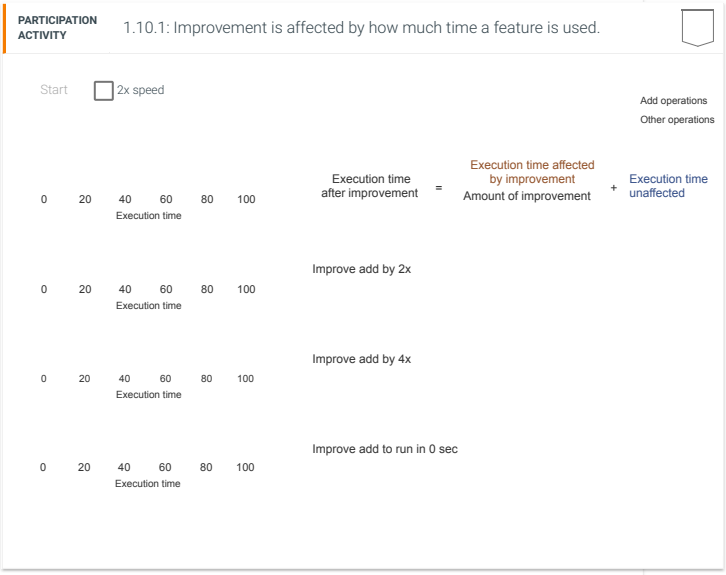
> "Science must begin with myths, and the criticism of myths."
> *Sir Karl Popper, The Philosophy of Science, 1957*

The purpose of a section on fallacies and pitfalls, which will be found in every chapter, is to explain some commonly held misconceptions that you might encounter. We call them fallacies. When discussing a fallacy, we try to give a counterexample. We also discuss pitfalls, or easily made mistakes. Often pitfalls are generalizations of principles that are true in a limited context. The purpose of these sections is to help you avoid making these mistakes in the computers you may design or use. Cost/performance fallacies and pitfalls have ensnared many a computer architect, including us. Accordingly, this section suffers no shortage of relevant examples. We start with a pitfall that traps many designers and reveals an important relationship in computer design.

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

The great idea of making the *common case fast* has a demoralizing corollary that has plagued designers of both hardware and software. It reminds us that the opportunity for improvement is affected by how much time the event consumes.

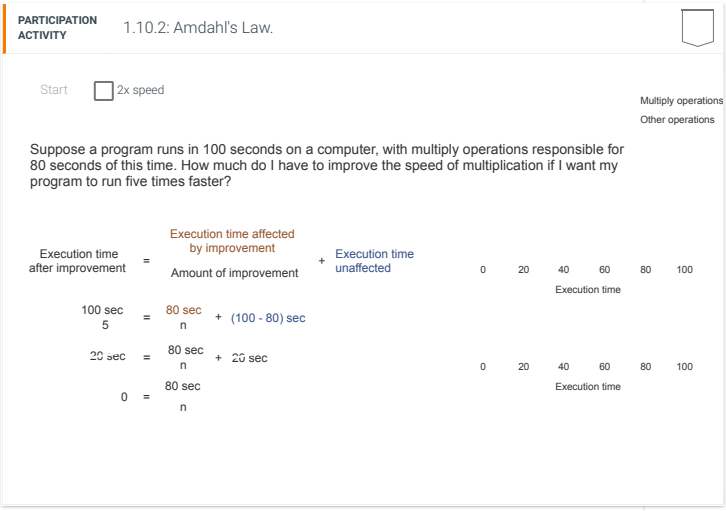
PARTICIPATION ACTIVITY 1.10.1: Improvement is affected by how much time a feature is used.



A simple design problem illustrates it well. 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?

The execution time of the program after making the improvement is given by the following simple equation known as *Amdahl's Law*:

PARTICIPATION ACTIVITY 1.10.2: Amdahl's Law.



That is, there is no amount by which we can enhance multiply to achieve a fivefold increase in performance, if multiply accounts for only 80% of the workload. The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. In everyday life this concept also yields what we call the law of diminishing returns.

Amdahl's Law: A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. It is a quantitative version of the law of diminishing returns.

We can use Amdahl's Law to estimate performance improvements when we know the time consumed for some function and its potential speedup. Amdahl's Law, together with the CPU performance equation, is a handy tool for evaluating possible enhancements. Amdahl's Law is explored in more detail in the exercises.

Amdahl's Law is also used to argue for practical limits to the number of parallel processors. We examine this argument in the Fallacies and Pitfalls section of COD Chapter 6 (Parallel Processor from Client to Cloud).

PARTICIPATION ACTIVITY 1.10.3: Amdahl's Law.

1) A program runs in 100 seconds. Multiply operations are responsible for 30 of those seconds. If extensive designer effort is applied such that multiply operations are made to run 2 times faster, what is the program's new execution time?

sec

Check Show answer

2) If some aspect of a computer accounts for 50% of program execution time, what is the limit on how many times faster programs can run if engineers focus on improving that aspect?

times

Check Show answer

Fallacy: Computers at low utilization use little power.

Power efficiency matters at low utilizations because server workloads vary. Utilization of servers in Google's warehouse scale computer, for example, is between 10% and 50% most of the time and at 100% less than 1% of the time. Even given 5 years to learn how to run the SPECpower benchmark well, the specially configured computer with the best results in 2012 still uses 33% of the peak power at 10% of the load. Systems in the field that are not configured for the SPECpower benchmark are surely worse.

Since servers' workloads vary but use a large fraction of peak power, Luiz Barroso and Urs Hölzle [2007] argue that we should redesign hardware to achieve "energy-proportional computing." If future servers used, say, 10% of peak power at 10% workload, we could reduce the electricity bill of datacenters and become good corporate citizens in an era of increasing concern about CO2 emissions.

Fallacy: Designing for performance and designing for energy efficiency are unrelated goals.

Since energy is power over time, it is often the case that hardware or software optimizations that take less time save energy overall even if the optimization takes a bit more energy when it is used. One reason is that all the rest of the computer is consuming energy while the program is running, so even if the optimized portion uses a little more energy, the reduced time can save the energy of the whole system.

PARTICIPATION ACTIVITY 1.10.4: Fallacies.

1) Google's warehouse scale computer uses 5% of the peak power when running at 10% utilization.

- True
- False

2) Designers must choose between energy and performance. If a computer is designed for improved performance, then the computer's energy consumption will increase.

- True
- False

Pitfall: Using a subset of the performance equation as a performance metric.

We have already warned about the danger of predicting performance based on simply one of clock rate, instruction count, or CPI. Another common mistake is to use only two of the three factors to compare performance. Although using two of the three factors may be valid in a limited context, the concept is also easily misused. Indeed, nearly all proposed alternatives to the use of time as the performance metric have led eventually to misleading claims, distorted results, or incorrect interpretations.

One alternative to time is *MIPS (million instructions per second)*. For a given program, MIPS is simply

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

Since MIPS is an instruction execution rate, MIPS specifies performance inversely to execution time; faster computers have a higher MIPS rating. The good news about MIPS is that it is easy to understand, and quicker computers mean bigger MIPS, which matches intuition.

Million instructions per second (***MIPS***): A measurement of program execution speed based on the number of millions of instructions. MIPS is computed as the instruction count divided by the product of the execution time and 10^6.

There are three problems with using MIPS as a measure for comparing computers. First, MIPS specifies the instruction execution rate but does not take into account the capabilities of the instructions. We cannot compare computers with different instruction sets using MIPS, since the instruction counts will certainly differ. Second, MIPS varies between programs on the same computer; thus, a computer cannot have a single MIPS rating. For example, by substituting for execution time, we see the relationship between MIPS, clock rate, and CPI:

$$\text{MIPS} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

The CPI varied by a factor of 5 for SPEC CPU2006 on an Intel Core i7 computer in COD Figure 1.18 (SPECINTC2006 benchmarks running on a 2.66GHz Intel Core i7 920), so MIPS does as well. Finally, and most importantly, if a new program executes more instructions but each instruction is faster, MIPS can vary independently from performance!

PARTICIPATION ACTIVITY 1.10.5: Check yourself: MIPS.

Consider the following performance measurements for a program:

Measurement	Computer A	Computer B
Instruction count	10 billion	8 billion
Clock rate	4 GHz	4 GHz
CPI	1.0	1.1

1) Which computer has the higher MIPS rating?
   - ○ Computer A
   - ○ Computer B

2) Which computer is faster for that program?
   - ○ Computer A
   - ○ Computer B

Provide feedback on this section