

## Amdahl's Law

The performance gain that can be obtained by improving some portion of a computer can be calculated using Amdahl's Law. Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Amdahl's Law defines the *speedup* that can be gained by using a particular feature. What is speedup? Suppose that we can make an enhancement to a computer that will improve performance when it is used. Speedup is the ratio

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

Alternatively

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

Speedup tells us how much faster a task will run using the computer with the enhancement as opposed to the original computer.

Amdahl's Law gives us a quick way to find the speedup from some enhancement, which depends on two factors:

- *The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement--For example, if 20 seconds of the execution time of a program that takes 60 seconds in total can use an enhancement, the fraction is 20/60. This value, which we will call  $\text{Fraction}_{\text{enhanced}}$ , is always less than or equal to 1.*
- *The improvement gained by the enhanced execution mode; that is, how much faster the task would run if the enhanced mode were used for the entire program- This value is the time of the original mode over the time of the enhanced mode. If the enhanced mode takes, say, 2 seconds for a portion of the program, while it is 5 seconds in the original mode, the improvement is 5/2. We will call this value, which is always greater than 1,  $\text{Speedup}_{\text{enhanced}}$ .*

The execution time using the original computer with the enhanced mode will be the time spent using the unenhanced portion of the computer plus the time spent using the enhancement:

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left( (1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

The overall speedup is the ratio of the execution times:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Amdahl's Law expresses the law of diminishing returns: The incremental improvement in speedup gained by an improvement of just a portion of the computation diminishes as improvements are added. An important corollary of Amdahl's Law is that if an enhancement is only usable for a fraction of a task, we can't speed up the task by more than the reciprocal of 1 minus that fraction.

A common mistake in applying Amdahl's Law is to confuse "fraction of time converted to use an enhancement" and "fraction of time after enhancement is in use." If, instead of measuring the time that we *could use* the enhancement in a computation, we measure the time *after* the enhancement is in use, the results will be incorrect!

Amdahl's Law can serve as a guide to how much an enhancement will improve performance and how to distribute resources to improve cost performance. The goal, clearly, is to spend resources proportional to where time is spent. Amdahl's Law is particularly useful for comparing the overall system performance of two alternatives, but it can also be applied to compare two processor design alternatives, as the following example shows.

**Problem** Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

**Answer**

$$\text{Fraction}_{\text{enhanced}} = 0.4$$

$$\text{Speedup}_{\text{enhanced}} = 10$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

**Problem** A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10. The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

**Answer**

We can compare these two alternatives by comparing the speedups:

$$\text{Speedup}_{\text{FPSQR}} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

Improving the performance of the FP operation overall is slightly better because of the higher frequency.