Amdahl's Law

If you are only able to speed up a portion of your program, the overall improvement is going to be a lot less than you might think! For example, if we can make half of a program run 10 times faster, does this mean that the program overall will run 5 times faster? Unfortunately not.

Suppose we have i instructions, each taking t time units to execute. Then, the execution time is i*t.

Next, suppose that for a proportion, p, of the program (0 < p < 1), we can reduce the execution time by a factor of n. In other words, that part of the program experiences a speedup of n.

What is the new execution time? Let's split up the program into two parts, the part that improves, and the part that does not.

$$exec\ time = improving\ part + static\ part$$

$$= i * p * \left(\frac{t}{n}\right) + i * (1 - p) * t$$

$$= i * t * (\frac{p}{n} + 1 - p)$$

We can now calculate the speedup of the program as a whole. It is the old execution time divided by the new execution time.

$$speedup = \frac{i * t}{i * t * (\frac{p}{n} + 1 - p)} = \frac{1}{\frac{p}{n} + 1 - p}$$

Let's look at a numerical example. Assume we have i = 1,000 instructions, p = 0.75, and t = 10 nanoseconds. Note that the speedup formula does not use i or t. In this case, the speedup can be written in terms of n:

$$speedup = \frac{1}{\frac{0.75}{n} + 0.25} = \frac{4n}{n + 3}$$

So, if n = 8, then the speedup is 32/11, which is about 2.9. This is nowhere near 8!

1. What is the theoretical maximum speedup? Does this answer make intuitive sense to you?
2. It turns out that the theoretical maximum speedup is a function of p. What is it?
3. For example, if you aim to improve 10% of the code, what is the best possible speedup you could achieve?
4. If you could improve 25% of the code, what is the best possible speedup?
5. If you are able to double the speed of 10% of the code, what is the speedup of the program overall?
6. Going back to the example at the top of the page, what is the overall speedup if it's not 5x?