Amdahl's Law

Blindly adding threads to speed up program execution may not always be a good idea. Find out what Amdahl's Law says about parallelizing a program

Definition

No text on concurrency is complete without mentioning the ***Amdahl's Law***. The law specifies the cap on the maximum speedup that can be achieved when parallelizing the execution of a program.

If you have a poultry farm where a hundred hens lay eggs each day, then no matter how many people you hire to process the laid eggs, you still need to wait an entire day for the 100 eggs to be laid. Increasing the number of workers on the farm can't shorten the time it takes for a hen to lay an egg. Similarly, software programs consist of parts which can't be sped up even if the number of processors is increased. These parts of the program must execute serially and aren't amenable to parallelism.

Amdahl's law describes the theoretical speedup a program can achieve at best by using additional computing resources. We'll skip the mathematical derivation and go straight to the simplified equation expressing Amdahl's law:

$$S(n) = \frac{1}{(1 - P) + \frac{P}{n}}$$

- ***S(n)*** is the speed-up achieved by using ***n*** cores or threads.
- ***P*** is the fraction of the program that is parallelizable
- ***(1 - P)*** is the fraction of the program that must be executed serially.

Example

Say our program has a parallelizable portion of **P = 90% = 0.9**. Now let's see how the speed-up occurs as we increase the number of processes

- ***n = 1 processor*** $S(1) = \frac{1}{(1 - P) + \frac{P}{1}} = \frac{1}{1 - P + P} = 1$
- ***n = 2 processors*** $S(2) = \frac{1}{(1 - 0.9) + \frac{0.9}{2}} = \frac{1}{0.1 + 0.45} = 1.81$
- ***n = 5 processors*** $S(5) = \frac{1}{(1 - 0.9) + \frac{0.9}{5}} = \frac{1}{0.1 + 0.18} = 3.57$
- ***n = 10 processors*** $S(10) = \frac{1}{(1 - 0.9) + \frac{0.9}{10}} = \frac{1}{0.1 + 0.09} = 5.26$
- ***n = 100 processors*** $S(100) = \frac{1}{(1 - 0.9) + \frac{0.9}{100}} = \frac{1}{0.1 + 0.009} = 9.17$
- ***n = 1000 processors*** $S(1000) = \frac{1}{(1 - 0.9) + \frac{0.9}{1000}} = \frac{1}{0.1 + 0.0009} = 9.91$
- ***n = infinite processors*** $S(\infty) = \frac{1}{(1 - 0.9) + \frac{0.9}{\infty}} = \frac{1}{0.1 + 0} = 10$

The speed-up steadily increases as we increase the number of processors or threads. However, as you can see the theoretical maximum speed-up for our program with 10% serial execution will be 10. We can't speed-up our program execution more than 10 times compared to when we run the same program on a single CPU or thread. To achieve greater speed-ups than 10 we must optimize or parallelize the serially executed portion of the code.

Another important aspect to realize is that when we speed-up our program execution by roughly 5 times, we do so by employing 10 processors. The utilization of these 10 processors, in turn, decreases by roughly 50% because now the 10 processors remain idle for the rest of the time that a single processor would have been busy. Utilization is defined as the ***speedup divided by the number of processors***.

As an example say the program runs in 10 minutes using a single core. We assumed the parallelizable portion of the program is 90%, which implies 1 minute of the program time must execute serially. The speedup we can achieve with 10 processors is roughly 5 times which comes out to be 2 minutes of total program execution time. Out of those 2 minutes, 1 minute is of mandatory serial execution and the rest can all be parallelized. This implies that 9 of the processors will complete 90% of the non-serial work in 1 minute while 1 processor remains idle and then one out of the 10 processors, will execute the serial portion for another minute. The rest of the 9 processors are idle for that 1 minute during which the serial execution takes place. In effect, the combined utilization of the ten processors drops by 50%.

As ***N*** approaches infinity, the Amdahl's law takes the following form:

$$S(n) = \frac{1}{(1 - P)} = \frac{1}{fraction\ of\ program\ serially\ executed}$$

One should take the calculations using Amdahl's law with a grain of salt. If the formula spits out a speed-up of 5x it doesn't imply that in reality one would observe a similar speed-up. There are other factors such as the memory architecture, cache misses, network and disk I/O etc that can affect the execution time of a program and the actual speed-up might be less than the calculated one.

The Amdahl's law works on a problem of fixed size. However as computing resources are improved, algorithms run on larger and even larger datasets. As the dataset size grows, the parallelizable portion of the program grows faster than the serial portion and a more realistic assessment of performance is given by **Gustafson's law**, which we won't discuss here as it is beyond the scope of this text.

← Back lesson
Semaphore vs Monitor

Completed Next →
Moore's Law