**Parallel Processing Paradox: A Comparison of Amdahl’s and Gustafson’s Laws**

Zak Kastl

30 March 2018

# Introduction

Processing power has advanced exponentially in the last 25 years. What once required room-sized supercomputers are now computable on the personal mobile phone. Processing speed has advanced so far that creating smaller, faster processor beginning to become physically impossible.

So, what can be done? If there is a physical limit to how fast we can make one processor, are we approaching the limits to computing? The answer is no. If we cannot make a faster processor, how about using two, three, or more processors at the same time? This answer is where Parallel Computing begins.

What is the benefit of this parallel processing. If we consider a program that is perfectly parallelizable, where all the processing of the program is done in parallel, then we can reasonably conclude that we want an increase in speed proportional to that of the number of processors we add to the program. This excited people. Simply by adding processors, could we get linear (or even near linear) speedup of our software?

This paper will explore two seemingly conflicting responses to this question. Amdahl’s and Gustafson’s Laws. Both are mathematically sound but contradict each other. This paper will explore each of the laws and examines its answers to this question. Finally, the paper will examine whether each law truly answers the question at hand.

# Amdahl’s Law

In 1967, Gene Amdahl formulated the basis for what would be called Amdahl’s Law. While Amdahl did not give a specific formula in his paper, the literal description of the law gave rise to this law:

Where

* is the theoretical speedup of the execution of the whole tasks;
* is the number of processors used;
* is the amount of time performing tasks that do not benefit from improved system resources;
* is the proportion of the execution time that the part benefiting from improved resources originally occupied. [1] [2]

Amdahl concluded that because of the extra hardware and connections required by two processors instead of one, compared to a processor twice as fast, “The net result (of two processors) is a price performance degradation to 0.8 rather than an improvement to 2.0 for the single larger processor.”

This is worrisome. If multiple processors do not increase the performance of a system, then we will eventually run into diminishing returns regarding performance. Indeed, Amdahl’s paper was a splash of cold water into the growing field of computer science.

# Gustafson’s Law

The pall that Amdahl’s Law hung over the growing field of parallel computation lasted until 1988. That year, John L. Gustafson, working at Sandia National Laboratories (SNL), published a paper entitled *Reevaluating Amdahl’s Law*. This short paper described their research into massively-processing and addressed the skepticism surrounding it. Gustafson worked on a 1024-processor system and they performed timing tests on that system. Using Amdahl’s Law, as simplified speedup of the problem would be close to 1/1024, a value very close to 0.

However, Gustafson’s paper described 3 programs run on their 1024-processor machine that averaged a speedup of 1019 for the experiment described in his paper, a nearly linear improvement. Gustafson would later describe his estimation of speedup as follows:

where:

* S is the theoretical speedup of the execution of the task;
* N is the number of processors used for the task;
* s is a serial fraction of the task which does not benefit from parallelism.

Both men prove their arguments in their individual papers that gave rise to their laws, and Gustafson empirically proved his law by running the programs and experiencing their near-linear speedup. This begs the question on how both can be correct.

# Comparing the Laws

If we assume for the moment that both men cannot be right, we must counter the arguments. Gustafson has the advantage in this case because he provided both mathematical and empirical proof of his theory. The fact that SNL could tasks on their 1024-processor Tesseract system at nearly linear speedup leads to the idea that he is correct. Therefore, Amdahl must have a flaw in his argument.

Gustafson himself provides a possible explanation. In his paper, Gustafson informs us of a crucial detail, “The expression...(contains) the implicit assumption that p (the portion of the program that benefits from increased processors) is independent of N (the number of processors in the system), *which is virtually never the case*…. In practice, *the problem size scales with the number of processors.”* [2]

This is important, for it shows that rather than contradicting or disagreeing with one another, both men are in fact correct. The difference lies in the *question* being asked. Amdahl is correct when he says that tasks with a fixed problem size do not benefit greatly from the increased performance of parallel processing. Much like the adage, *build it and they will come*, Gustafson realizes that the problem size of the parallel portion can be increased with only a minimal increase in the time to perform the same task.

Amdahl, rather, operated from the assumption that parallel processing would decrease the time it takes to perform the same task on the same data. In this, he is correct. Increasing the number of processors in a system and tasking that system to perform the same function on the same data does not decrease the time to run the task linearly because the *not all functions of a task are parallelizable.* In addition, the portions that are parallelizable do not scale with the cost of multiple processors for running the same task.

# Takeaway

Now we must ask ourselves, “What does all of this mean?” We have seen that the question of parallelization requires a reexamination of the problem size of a task, rather than the simple speedup of said task. Running the tasks associated with booting an operating system, for example, will not benefit much from parallel processing as those tasks are serial in nature: “boot the OS, load external I/O devices, restore previous states”. This is a linear series of tasks that cannot be placed in parallel because they possibly require the status of the previous task to influence the next task.

On the other hand, this is fantastic news for different processing tasks. If we consider the case where a serial task takes a short time to run, say 10 seconds, we have a starting point for additional features that we could potentially run in parallel. If we consider Microsoft’s operating system (let us consider DOS/Windows as one operating system rather than two closely linked ones), released as DOS 1.0 on 7 May 1981, the same basic tasks apply today as they did 36 years ago: read OS from disk, load external I/O devices, restore states. [3]

Today, those tasks are supplemented with Graphical User Interfaces, program windows, network connectivity, real-time file searching, AI assistants, and other tasks that have increased the amount of source lines of code (SLOC) from around 4,000 for DOS 1.0 [4] to somewhere in the order of 50-60 million SLOC for Windows 10 [5], an increase of around *15,000%.* If we suppose that DOS 1.0 took around 10 seconds to boot, a modern Windows 10 system, might require 60 seconds to boot using a traditional hard disk as opposed to a solid-state drive, an increase of 600%.

This is a far cry from the 15,000% increase in potential complexity that SLOC counts might suggest. One cannot ignore the Moore’s Law level of increased performance of the hardware Windows runs on today, but the staggering number of things that Windows 10 can manage, not which the least is dozens of windowed programs running simultaneously and potentially hundreds of non-static internet applications running within a web browser, could only have been brought about with parallel processing.

Amdahl reminds us that there is a limit to how fast a program, a task, a system can run. Gustafson allows us to make the most of what we have. This will be especially important with the rise of machine learning (ML). Gustafson relies on increasing datasets to make full benefit of the performance increases brought about by multiple processors. ML relies primarily on increased datasets to make better predictions about future behaviors based on that data.

It is in this field that multiple processors will truly shine in the future. We are getting a glimpse of this today with the data centers employed by companies that rely on data for their functions: Amazon Google, and Facebook. If we keep in mind the question that parallel processing truly answers, we will reap the benefits they provide.

# List of References

[1] <http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf>

[2] <http://www.johngustafson.net/pubs/pub13/amdahl.htm>

[3] [https://en.wikipedia.org/wiki/Gustafson’s\_law/](https://en.wikipedia.org/wiki/Gustafson's_law/)

[4] <http://www.os2museum.com/wp/dos/dos-1-0-and-1-1/>

[5] <https://www.quora.com/How-man-lines-of-code-does-Windows-10-contain/>