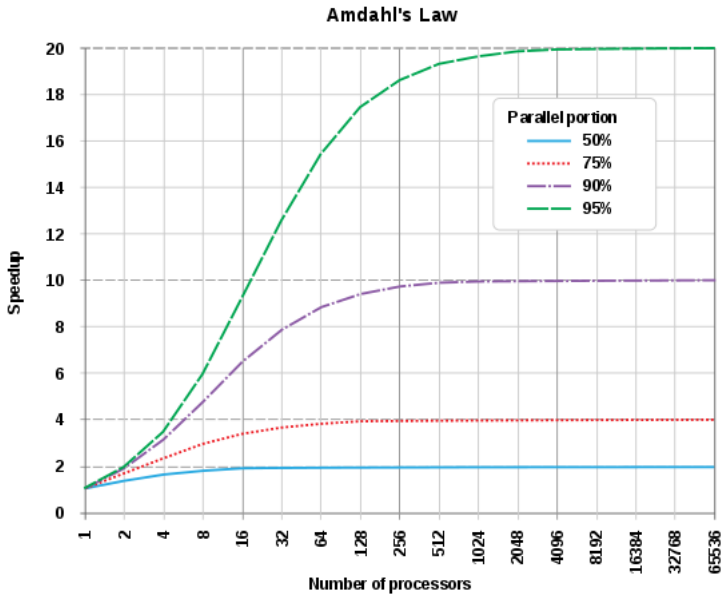
WIKIPEDIA

Amdahl's law

In computer architecture, **Amdahl's law** (or **Amdahl's argument**[1]) is a formula which gives the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved. It states that "the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used".[2] It is named after computer scientist Gene Amdahl, and was presented at the American Federation of Information Processing Societies (AFIPS) Spring Joint Computer Conference in 1967.



The theoretical speedup of the latency (via a reduction of latency, ie: latency as a metric is elapsed time between an input and output in a system) of the execution of a program as a function of the number of processors executing it, according to Amdahl's law. The speedup is limited by the serial part of the program. For example, if 95% of the program can be parallelized, the theoretical maximum speedup using parallel computing would be 20 times.

Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors. For example, if a program needs 20 hours to complete using a single thread, but a one-hour portion of the program cannot be parallelized, therefore only the remaining 19 hours' (p = 0.95) execution time can be parallelized, then regardless of how many threads are devoted to a parallelized execution of this program, the minimum execution time cannot be less than one hour. Hence, the theoretical speedup is limited to at most 20 times the single thread performance, $\left(\frac{1}{1-p} = 20\right)$.

Contents

Definition

Amdahl's law can be formulated in the following way:[3]

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

where

- S_{latency} is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

Furthermore,

$$\begin{cases} S_{\text{latency}}(s) \leq \dfrac{1}{1-p} \\ \lim\limits_{s \to \infty} S_{\text{latency}}(s) = \dfrac{1}{1-p}. \end{cases}$$

shows that the theoretical speedup of the execution of the whole task increases with the improvement of the resources of the system and that regardless of the magnitude of the improvement, the theoretical speedup is always limited by the part of the task that cannot benefit from the improvement.

Amdahl's law applies only to the cases where the problem size is fixed. In practice, as more computing resources become available, they tend to get used on larger problems (larger datasets), and the time spent in the parallelizable part often grows much faster than the inherently serial work. In this case, Gustafson's law gives a less pessimistic and more realistic assessment of the parallel performance.[4]

Derivation

A task executed by a system whose resources are improved compared to an initial similar system can be split up into two parts:

- a part that does not benefit from the improvement of the resources of the system;
- a part that benefits from the improvement of the resources of the system.

An example is a computer program that processes files. A part of that program may scan the directory of the disk and create a list of files internally in memory. After that, another part of the program passes each file to a separate thread for processing. The part that scans the directory and creates the file list cannot be sped up on a parallel computer, but the part that processes the files can.

The execution time of the whole task before the improvement of the resources of the system is denoted as T. It includes the execution time of the part that would not benefit from the improvement of the resources and the execution time of the one that would benefit from it. The fraction of the execution time of the task that would benefit from the improvement of the resources is denoted by p. The one concerning the part that would not benefit from it is therefore $1 - p$. Then:

$$T = (1 - p)T + pT.$$

It is the execution of the part that benefits from the improvement of the resources that is accelerated by the factor s after the improvement of the resources. Consequently, the execution time of the part that does not benefit from it remains the same, while the part that benefits from it becomes:

$$\frac{p}{s}T.$$

The theoretical execution time $T(s)$ of the whole task after the improvement of the resources is then:

$$T(s) = (1 - p)T + \frac{p}{s}T.$$

Amdahl's law gives the theoretical speedup in latency of the execution of the whole task *at fixed workload* W, which yields

$$S_{\text{latency}}(s) = \frac{TW}{T(s)W} = \frac{T}{T(s)} = \frac{1}{1 - p + \frac{p}{s}}.$$

Parallel programs

If 30% of the execution time may be the subject of a speedup, p will be 0.3; if the improvement makes the affected part twice as fast, s will be 2. Amdahl's law states that the overall speedup of applying the improvement will be:

$$S_{\text{latency}} = \frac{1}{1 - p + \frac{p}{s}} = \frac{1}{1 - 0.3 + \frac{0.3}{2}} = 1.18.$$

For example, assume that we are given a serial task which is split into four consecutive parts, whose percentages of execution time are $p1 = 0.11$, $p2 = 0.18$, $p3 = 0.23$, and $p4 = 0.48$ respectively. Then we are told that the 1st part is not sped up, so $s1 = 1$, while the 2nd part is sped up 5 times, so $s2 = 5$, the 3rd part is sped up 20 times, so $s3 = 20$, and the 4th part is sped up 1.6 times, so $s4 = 1.6$. By using Amdahl's law, the overall speedup is

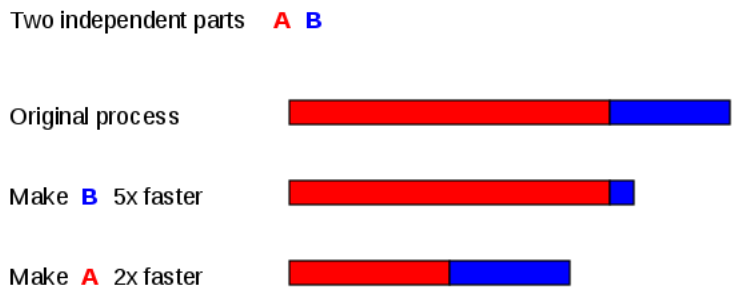
$$S_{\text{latency}} = \frac{1}{\frac{p1}{s1} + \frac{p2}{s2} + \frac{p3}{s3} + \frac{p4}{s4}} = \frac{1}{\frac{0.11}{1} + \frac{0.18}{5} + \frac{0.23}{20} + \frac{0.48}{1.6}} = 2.19.$$

Notice how the 5 times and 20 times speedup on the 2nd and 3rd parts respectively don't have much effect on the overall speedup when the 4th part (48% of the execution time) is accelerated by only 1.6 times.

Serial programs

For example, with a serial program in two parts A and B for which $T_A = 3$ s and $T_B = 1$ s,

- if part B is made to run 5 times faster, that is $s = 5$ and $p = T_B/(T_A + T_B) = 0.25$, then



Assume that a task has two independent parts, *A* and *B*. Part *B* takes roughly 25% of the time of the whole computation. By working very hard, one may be able to make this part 5 times faster, but this reduces the time of the whole computation only slightly. In contrast, one may need to perform less work to make part *A* perform twice as fast. This will make the computation much faster than by optimizing part *B*, even though part *B*'s speedup is greater in terms of the ratio, (5 times versus 2 times).

$$S_{\text{latency}} = \frac{1}{1 - 0.25 + \frac{0.25}{5}} = 1.25;$$

- if part A is made to run 2 times faster, that is $s = 2$ and $p = T_A/(T_A + T_B) = 0.75$, then

$$S_{\text{latency}} = \frac{1}{1 - 0.75 + \frac{0.75}{2}} = 1.60.$$

Therefore, making part A to run 2 times faster is better than making part B to run 5 times faster. The percentage improvement in speed can be calculated as

$$\text{percentage improvement} = 100\left(1 - \frac{1}{S_{\text{latency}}}\right).$$

- Improving part A by a factor of 2 will increase overall program speed by a factor of 1.60, which makes it 37.5% faster than the original computation.
- However, improving part B by a factor of 5, which presumably requires more effort, will achieve an overall speedup factor of 1.25 only, which makes it 20% faster.

Optimizing the sequential part of parallel programs

If the non-parallelizable part is optimized by a factor of O, then

$$T(O, s) = (1-p)\frac{T}{O} + \frac{p}{s}T.$$

It follows from Amdahl's law that the speedup due to parallelism is given by

$$S_{\text{latency}}(O, s) = \frac{T(O)}{T(O, s)} = \frac{(1-p)\frac{1}{O} + p}{\frac{1-p}{O} + \frac{p}{s}}.$$

When $s = 1$, we have $S_{\text{latency}}(O, s) = 1$, meaning that the speedup is measured with respect to the execution time after the non-parallelizable part is optimized.

When $s = \infty$,

$$S_{\text{latency}}(O, \infty) = \frac{T(O)}{T(O, s)} = \frac{(1-p)\frac{1}{O} + p}{\frac{1-p}{O} + \frac{p}{s}} = 1 + \frac{p}{1-p}O.$$

If $1 - p = 0.4$, $O = 2$ and $s = 5$, then:

$$S_{\text{latency}}(O, s) = \frac{T(O)}{T(O, s)} = \frac{0.4\frac{1}{2} + 0.6}{\frac{0.4}{2} + \frac{0.6}{5}} = 2.5.$$

Transforming sequential parts of parallel programs into parallelizable

Next, we consider the case wherein the non-parallelizable part is reduced by a factor of O', and the parallelizable part is correspondingly increased. Then

$$T'(O', s) = \frac{1-p}{O'}T + \left(1 - \frac{1-p}{O'}\right)\frac{T}{s}.$$

It follows from Amdahl's law that the speedup due to parallelism is given by

$$S'_{\text{latency}}(O', s) = \frac{T'(O')}{T'(O', s)} = \frac{1}{\frac{1-p}{O'} + \left(1 - \frac{1-p}{O'}\right)\frac{1}{s}}.$$

The derivation above is in agreement with Jakob Jenkov's analysis of the execution time vs. speedup tradeoff.[5]

Relation to the law of diminishing returns

Amdahl's law is often conflated with the law of diminishing returns, whereas only a special case of applying Amdahl's law demonstrates law of diminishing returns. If one picks optimally (in terms of the achieved speedup) what is to be improved, then one will see monotonically decreasing improvements as one improves. If, however, one picks non-optimally, after improving a sub-optimal component and moving on to improve a more optimal component, one can see an increase in the

return. Note that it is often rational to improve a system in an order that is "non-optimal" in this sense, given that some improvements are more difficult or require larger development time than others.

Amdahl's law does represent the law of diminishing returns if one is considering what sort of return one gets by adding more processors to a machine, if one is running a fixed-size computation that will use all available processors to their capacity. Each new processor added to the system will add less usable power than the previous one. Each time one doubles the number of processors the speedup ratio will diminish, as the total throughput heads toward the limit of $1/(1 - p)$.

This analysis neglects other potential bottlenecks such as memory bandwidth and I/O bandwidth. If these resources do not scale with the number of processors, then merely adding processors provides even lower returns.

An implication of Amdahl's law is that to speed up real applications which have both serial and parallel portions, heterogeneous computing techniques are required.[6] There are novel speedup and energy consumption models based on a more general representation of heterogeneity, referred to as the normal form heterogeneity, that support a wide range of heterogeneous many-core architectures. These modelling methods aim to predict system power efficiency and performance ranges, and facilitates research and development at the hardware and system software levels.[7][8]

See also

- Gustafson's law
- Analysis of parallel algorithms
- Critical path method
- Moore's law

References

1. Rodgers, David P. (June 1985). "Improvements in multiprocessor system design". *ACM SIGARCH Computer Architecture News*. New York, NY, USA: ACM. **13** (3): 225–231 [p. 226]. doi:10.1145/327070.327215 (https://doi.org/10.1145%2F327070.327215). ISBN 0-8186-0634-7. ISSN 0163-5964 (https://www.worldcat.org/issn/0163-5964). S2CID 7083878 (https://api.semanticscholar.org/CorpusID:7083878).
2. Reddy, Martin (2011). *API Design for C++*. Burlington, Massachusetts: Morgan Kaufmann Publishers. doi:10.1016/C2010-0-65832-9 (https://doi.org/10.1016%2FC2010-0-65832-9). ISBN 978-0-12-385003-4. LCCN 2010039601 (https://lccn.loc.gov/2010039601). OCLC 666246330 (https://www.worldcat.org/oclc/666246330).
3. Bryant, Randal E.; David, O'Hallaron (2016), *Computer Systems: A Programmer's Perspective* (3 ed.), Pearson Education, p. 58, ISBN 978-1-488-67207-1
4. McCool, Michael; Reinders, James; Robison, Arch (2013). *Structured Parallel Programming: Patterns for Efficient Computation*. Elsevier. p. 61. ISBN 978-0-12-415993-8.
5. "Amdahl's Law" (http://tutorials.jenkov.com/java-concurrency/amdahls-law.html).
6. Hill, Mark D.; Marty, Michael R. (2008). "Amdahl's Law in the Multicore Era". *Computer*. **41** (7): 33–38. doi:10.1109/MC.2008.209 (https://doi.org/10.1109%2FMC.2008.209).

7. Rafiev, Ashur; Al-Hayanni, Mohammed A. N.; Xia, Fei; Shafik, Rishad; Romanovsky, Alexander; Yakovlev, Alex (2018-07-01). "Speedup and Power Scaling Models for Heterogeneous Many-Core Systems" (https://ieeexplore.ieee.org/document/8255653/). *IEEE Transactions on Multi-Scale Computing Systems*. **4** (3): 436–449. doi:10.1109/TMSCS.2018.2791531 (https://doi.org/10.1109%2FTMSCS.2018.2791531). ISSN 2332-7766 (https://www.worldcat.org/issn/2332-7766).
8. Al-hayanni, Mohammed A. Noaman; Xia, Fei; Rafiev, Ashur; Romanovsky, Alexander; Shafik, Rishad; Yakovlev, Alex (July 2020). "Amdahl's law in the context of heterogeneous many-core systems – a survey" (https://onlinelibrary.wiley.com/doi/10.1049/iet-cdt.2018.5220). *IET Computers & Digital Techniques*. **14** (4): 133–148. doi:10.1049/iet-cdt.2018.5220 (https://doi.org/10.1049%2Fiet-cdt.2018.5220). ISSN 1751-8601 (https://www.worldcat.org/issn/1751-8601).

Further reading

- Amdahl, Gene M. (1967). "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities" (https://inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf) (PDF). *AFIPS Conference Proceedings* (30): 483–485. doi:10.1145/1465482.1465560 (https://doi.org/10.1145%2F1465482.1465560).

External links

- "Parallel Programming: When Amdahl's law is inapplicable?" (https://archive.today/20130414224506/http://www.futurechips.org/thoughts-for-researchers/parallel-programming-gene-amdahl-said.html). 2011-06-25. Archived from the original (http://www.futurechips.org/thoughts-for-researchers/parallel-programming-gene-amdahl-said.html) on 2013-04-14. Retrieved 2011-06-26.
- Gene M. Amdahl (1989), *Oral history interview with Gene M. Amdahl*, Charles Babbage Institute, University of Minnesota, hdl:11299/104341 (https://hdl.handle.net/11299%2F104341). Amdahl discusses his graduate work at the University of Wisconsin and his design of WISC. Discusses his role in the design of several computers for IBM including the STRETCH, IBM 701, and IBM 704. He discusses his work with Nathaniel Rochester and IBM's management of the design process. Mentions work with Ramo-Wooldridge, Aeronutronic, and Computer Sciences Corporation
- Amdahl's Law: Not all performance improvements are created equal (http://www.julianbrowne.com/article/viewer/amdahls-law) (2007)
- "Amdahl's Law" (http://demonstrations.wolfram.com/AmdahlsLaw/) by Joel F. Klein, Wolfram Demonstrations Project (2007)
- Amdahl's Law in the Multicore Era (https://research.cs.wisc.edu/multifacet/amdahl/) (July 2008)
- What the $#@! is Parallelism, Anyhow? (http://www.cprogramming.com/parallelism.html) (Charles Leiserson, May 2008)
- Evaluation of the Intel Core i7 Turbo Boost feature (https://web.archive.org/web/20160304000551/https://www.cs.sfu.ca/~fedorova/papers/TurboBoostEvaluation.pdf), by James Charles, Preet Jassi, Ananth Narayan S, Abbas Sadat and Alexandra Fedorova (2009)
- Calculation of the acceleration of parallel programs as a function of the number of threads (https://www.researchgate.net/publication/228569958), by George Popov, Valeri Mladenov and Nikos Mastorakis (January 2010)
- Danny Hillis - Proving Amdahl's Law wrong, video recorded October 2016 (https://www.webofstories.com/play/danny.hillis/109)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Amdahl%27s_law&oldid=1123971320"