Amdahl's and Gustafson-Barsis's Laws Revisited

Andrzej Karbowski
Warsaw University of Technology, Faculty of Electronics and Information Technology, Institute of Control and Computation Engineering, Nowowiejska 15/19, 00-665 Warszawa, Poland

Abstract: Amdahl's and Gustafson-Barsis's laws, which belong to the most influential concepts in parallel and distributed processing, describe limits on the speedup obtained owing to parallelization of an application when the sequential fraction of the time of the execution of it on, respectively, sequential and parallel machines, are given. In the literature these two laws are derived separately or their connections are shown in rather complicated ways. In this paper it is shown that by simple arithmetic transformations in few lines Gustafson-Barsis's Law can be obtained from Amdahl's Law and vice-versa.

Keywords: parallel computing, distributed computing, speedup, performance measures

1. Introduction

Amdahl's Law [1] was formulated about six decades ago and has been one of the most influential concepts in parallel and distributed processing [24]. The law describes the limits on the speedup obtained from the execution of an application on a parallel machine with its relation to the single-processor[1], sequential machine. More precisely, Amdahl's Law connects the speedup of the application on the parallel machine with the sequential fraction of the program and the number of processors.

Although several alternative efficiency measures have been proposed [6, 13, 23, 19] Amdahl's Law is still the most powerful tool to assess the performance of parallel systems. Only small modifications are necessary to describe the speedup in current systems with multicore processors [14, 9, 22, 10, 16].

About a decade after Amdahl's paper, on the basis of some experiments, Gustafson claimed that "the assumptions underlying Amdahl's 1967 argument are inappropriate for the current approach to massive ensemble parallelism" [7]. Furthermore, he formulated, using his words, "an alternative to Amdahl's Law suggested by E. Barsis at Sandia [National Laboratories]", in which the size of the problem increases with the number of processors. The so-called Gustafson-Barsis's Law is said to vindicate the use of massively parallel processing [18].

However, in the author's opinion, when we analyze deeper both laws, we see that they are strongly connected, that is Gustafson-Barsis's Law can be directly derived from Amdahl's Law and vice-versa. To the author's knowledge, there is no publication showing this formally. Even in the latest works [2, 3, 11, 16, 12, 19, 5, 17] these two laws are derived separately and recognized as the alternative descriptions of the complicated world of parallel processing.

In 1996 Shi in an unpublished manuscript [21] stated that "there is really only one law but two different formulations". He showed calculations on sample data how to get the speedup due to the Amdahl's Law having serial percentage from the Gustafson's Law and that both speedups are equal. Unfortunately, there is no formal analysis, and the formula connecting serial percentages in both laws is given arbitrarily.

There were trials to show the connection between these two laws done by Sun and Ni [23] and recently by Schryen [20]. They treated Amdahl and Gustafson-Barsis's laws as particular, simplified forms of a more general law estimating speedup, considering uneven workload allocation and communication overhead.

In this paper the author will show that the connection between Amdahl's and Gustafson-Barsis's laws is more direct and simpler and points out the essence of this relationship.

2. Derivation of Amdahl's and Gustafson-Barsis's laws

Although in the original Amdahl's paper [1] there were no equations, basing on the verbal description one may present his concept formally. It is assumed in the model that the program consists of two parts: sequential and parallel. While the time $T_{seq}(n)$ of the execution of the sequential part for a given size n is the same on all machines, independently of the number of processors p, the parallel part is assumed to be perfectly scalable, that is the time $T_{par}(n,p)$ of its execution on a machine with p processors is one p-th fraction of the time of the execution on the machine with one processor

$$T_{par}(n,p) = \frac{T_{par}(n,1)}{p} \tag{1}$$

[1] In this paper the term "processor" is used in the abstract sense, that is understood as a core.

Autor korespondujący:
Andrzej Karbowski, andrzej.karbowski@pw.edu.pl

Artykuł recenzowany
nadesłany 13.08.2025 r., przyjęty do druku 08.12.2025 r.

If we take into account, like Amdahl and Gustafson, only the structure of the application, neglecting communication and synchronization overheads and other factors influencing the real time of the execution of the application (e.g. process/thread creation time)[2] the total real-time $T(n,p)$ of the execution of the program on a machine which uses p processors will be equal to

$$T(n,p) = T_{seq}(n) + T_{par}(n,p) \tag{2}$$

We can define now the sequential fraction coefficient $\xi(n,p)$ of the application as:

$$\xi(n,p) = \frac{T_{seq}(n)}{T(n,p)} = \frac{T_{seq}(n)}{T_{seq}(n) + T_{par}(n,p)} \tag{3}$$

The sequential part time T_{seq} for a given problem size n and every $p = 1, 2, \ldots$ satisfies the equation

$$T_{seq}(n) = \xi(n,p) \cdot T(n,p) \tag{4}$$

In particular, for the sequential machine we express the sequential part time

$$T_{seq}(n) = \xi(n,1) \cdot T(n,1) \tag{5}$$

and the parallel part time

$$T_{par}(n,1) = \big(1 - \xi(n,1)\big) \cdot T(n,1) \tag{6}$$

The parallel part time T_{par} on the machine with p processors, owing to (1) and (6), may be calculated from the expression

$$T_{par}(n,p) = \frac{1}{p} \cdot T_{par}(n,1) = \frac{1}{p} \cdot \big(1 - \xi(n,1)\big) \cdot T(n,1) \tag{7}$$

The total time $T(n,p)$ on this machine due to (2), (5) and (7) is the sum of the sequential and the parallel parts time, that is

$$\begin{aligned} T(n,p) &= T_{seq}(n) + T_{par}(n,p) \\ &= \xi(n,1) \cdot T(n,1) + \frac{\big(1 - \xi(n,1)\big) \cdot T(n,1)}{p} \\ &= \left[\xi(n,1) + \frac{1 - \xi(n,1)}{p} \right] \cdot T(n,1) \end{aligned} \tag{8}$$

From (8) we get directly the formula for the speedup $S(n,p)$ obtained due to the parallelization of the application:

$$S(n,p) = \frac{T(n,1)}{T(n,p)} = \frac{1}{\xi(n,1) + \dfrac{1 - \xi(n,1)}{p}} \tag{9}$$

The formula (9) is called Amdahl's Law.

We may look at the parallel application from a different view. Gustafson was the first who noticed that "it is the parallel or vector part of a program that scales with the problem size" [7]. In some experiments described in [7, 8] it was assumed that the run time was constant, while the problem size scaled with the number of processors. More precisely, the time of the sequential part was independent, while the work to be done in parallel varied linearly with the number of processors.

Following Gustafson, the total time required by a serial processor to perform the task equals to[3]

$$T(n,1) = T_{seq}(n) + T_{par}(n,p) \cdot p \tag{10}$$

so, using this and (2), the scaled speedup of the process executed on the parallel system is equal to [15]:

$$\begin{aligned} S(n,p) = \frac{T(n,1)}{T(n,p)} &= \frac{T_{seq}(n) + T_{par}(n,p) \cdot p}{T_{seq}(n) + T_{par}(n,p)} \\ &= \frac{p\,T_{seq}(n) + p\,T_{par}(n,p) - (p-1)T_{seq}(n)}{T_{seq}(n) + T_{par}(n,p)} \\ &= p + (1-p)\frac{T_{seq}(n)}{T_{seq}(n) + T_{par}(n,p)} \\ &= p + (1-p) \cdot \xi(n,p) \end{aligned} \tag{11}$$

In short:

$$S(n,p) = p - (p-1)\xi(n,p) \tag{12}$$

The last equation is called the Gustafson-Barsis's Law.

3. The main result

We shall demonstrate now that by merely a simple substitution and elementary arithmetic operations Gustafson-Barsis's Law can be directly derived from Amdahl's Law. The derivation is based on the observation that for a given problem size n there is a constant in all executions of the program, on machines with different number of processors. This constant is the time of the execution of the sequential part $T_{seq}(n)$ for the given problem size n, as expressed in Eqn. (4). It is independent of the number of processors p, that is:

$$T_{seq}(n) = \xi(n,p) \cdot T(n,p) = \text{const.}, \quad p = 1,2,3,\ldots \tag{13}$$

So, for any $p = 1, 2, 3, \ldots$

$$\xi(n,1) \cdot T(n,1) = \xi(n,p) \cdot T(n,p) \tag{14}$$

From equation (14) we get:

$$\xi(n,1) = \xi(n,p) \cdot \frac{T(n,p)}{T(n,1)} \tag{15}$$

Replacing $\xi(n,1)$ in equation (8) by (15) we will get:

$$T(n,p) = \xi(n,p) \cdot T(n,p) + \frac{T(n,1)}{p} - \frac{\xi(n,p) \cdot T(n,p)}{p} \tag{16}$$

[2] The precise speedup model was presented e.g., in [4, 15]. The most important consequence of taking into account the overhead component in the formula of the total time T(n, p) is that the expressions on the speedup for all p > 1 will define the upper bound.

[3] For the consistency of the presentation, similar to other publications (e.g. [15]), time normalization used by Gustafson has been abandoned.

Now, multiplying both sides by p and moving all components with $T(n,p)$ to the left hand side we obtain:

$$p \cdot T(n,p) - p \cdot \xi(n,p) \cdot T(n,p) + \xi(n,p) \cdot T(n,p) = T(n,1) \quad (17)$$

To get the value of speedup $S(n,p)$ resulting from the parallelization we divide both sides of the equation (17) by $T(n,p)$. So, eventually, the speedup is equal to:

$$S(n,p) = \frac{T(n,1)}{T(n,p)} = p - (p-1) \cdot \xi(n,p) \quad (18)$$

In this way we received exactly Gustafson-Barsis's Law (12).

By performing the presented operations in the backward order the reverse derivation may be done.

4. Conclusions

In this paper it is shown that Gustafson-Barsis's Law can be directly derived from Amdahl's Law and vice-versa. It was obtained owing to the unification of terms and noticing that in both models, for any number of processors, one thing is constant – time of the execution of the sequential part. After that only simple mathematical operations are necessary to pass from one law to the other. This derivation should clarify the relationship between the two laws.

References

1. Amdahl G.M., *Validity of the single-processor approach to achieving large scale computing capabilities*, Proceedings of Spring Joint Computer Conference AFIPS '67, Vol. 30, 1967, 483–485, DOI: 10.1145/1465482.1465560.
2. Barlas G., *Multicore and GPU Programming: An Integrated Approach*, Morgan Kaufmann, Waltham, MA, 2015.
3. Basu S.K., *Parallel and Distributed Computing: Architectures and Algorithms*, PHI Learning, Delhi, 2016.
4. Bertsekas D.P., Tsitsiklis J.N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall Inc., Englewood Cliffs, 1989.
5. Chattopadhyay A., *Handbook of computer architecture*, Springer Nature Singapore Pte Ltd., 2025.
6. Foster I., *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison Wesley, Boston, 1995.
7. Gustafson J.L., *Reevaluating Amdahl's law*, "Communications of the ACM", Vol. 31, No. 5, 1988, 532–533, DOI: 10.1145/42411.42415.
8. Gustafson J.L., Montry G.R., Benner R.E., *Development of parallel methods for a 1024-processor hypercube*, "SIAM Journal on Scientific and Statistical Computing", Vol. 9, No. 4, 1988, 609–638, DOI: 10.1137/0909041.
9. Hill M.D., Marty M.R., *Amdahl's law in the multicore era*, "Computer", Vol. 41, No. 7, 2008, 33–38, DOI: 10.1109/MC.2008.209.
10. Huang T., Zhu Y., Qiu M., Yin X., Wang X., *Extending Amdahl's law and Gustafson's law by evaluating interconnections on multi-core processors*, "The Journal of Supercomputing", Vol. 66, 2013, 305–319, DOI: 10.1007/s11227-013-0908-9.
11. Hwang K., *Cloud Computing for Machine Learning and Cognitive Applications*, MIT Press, Cambridge MA, 2017.
12. Li Z., Duan F., Che H., *A unified scaling model in the era of big data analytics*, Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications, HP3C '19, 67–77, DOI: 10.1145/3318265.3318268.
13. Orii S., *Metrics for evaluation of parallel efficiency toward highly parallel processing*, "Parallel Computing", Vol. 36, No. 1, 2010, 16–25, DOI: 10.1016/j.parco.2009.11.003.
14. Paul J.M., Meyer B.H., *Amdahl's law revisited for single chip systems*, "International Journal on Parallel Programming", Vol. 35, No. 2, 2007, 101–123, DOI: 10.1007/s10766-006-0028-8.
15. Quinn M., *Parallel Programming in C with MPI and OpenMP*, McGraw-Hill Higher Education, New York, 2003.
16. Rafiev A., Al-Hayanni M.A.N., Xia F., Shafik R., Romanovsky A., Yakovlev A., *Speedup and power scaling models for heterogeneous many-core systems*, "IEEE Transactions on Multi-Scale Computing Systems", Vol. 4, No. 3, 2018, 436–449, DOI: 10.1109/TMSCS.2018.2791531.
17. Rauber T., Rünger G., *Parallel Programming: for Multicore and Cluster Systems*, Springer Nature Switzerland AG, 2023, DOI: 10.1007/978-3-031-28924-8.
18. Reilly E.D., *Milestones in Computer Science and Information Technology*, Greenwood Press, Westport, 2003.
19. Robertazzi T.G., Shi L., *Amdahl's and Other Laws*, [In:] *Networking and Computation,* Springer Nature Switzerland, Cham, Switzerland, 2nd edition, 2020, 139–149, DOI: 10.1007/978-3-030-36704-6_6.
20. Schryen G., *Speedup and efficiency of computational parallelization: A unifying approach and asymptotic analysis*, "Journal of Parallel and Distributed Computing", Vol. 187, 2024, DOI: 10.1016/j.jpdc.2023.104835.
21. Shi Y., *Reevaluating Amdahl's Law and Gustafson's Law*, [www.cis.temple.edu/~shi/docs/amdahl/amdahl.html], October 1996, Unpublished manuscript.
22. Sun X.-H., Chen Y., *Reevaluating Amdahl's law in the multicore era*, "Journal of Parallel and Distributed Computing", Vol. 70, No. 2, 2010, 183–188, DOI: 10.1016/j.jpdc.2009.05.002.
23. Sun X.-H., Ni L., *Scalable problems and memory-bounded speedup*, "Journal of Parallel and Distributed Computing", Vol. 19, No. 1, 1993, 27–37, DOI: 10.1006/jpdc.1993.1087.
24. Theys M.D., Ali S., Siegel H.J., Chandy M., Hwang K., Kennedy K., Sha L., Shin K.G., Snir M., Snyder L., Sterling T., *What are the top ten most influential parallel and distributed processing concepts of the past millenium?* "Journal of Parallel and Distributed Computing", Vol. 61, No. 12, 2001, 1827–1841, DOI: 10.1006/jpdc.2001.1767.

Prawa Amdahla i Gustafsona-Barsisa – ponowne spojrzenie

Streszczenie: Prawa Amdahla i Gustafsona-Barsisa, należące do najbardziej wpływowych koncepcji w dziedzinie przetwarzania równoległego i rozproszonego, opisują ograniczenia przyspieszenia uzyskiwanego dzięki zrównolegleniu aplikacji, gdy podany jest sekwencyjny ułamek czasu jej wykonania na maszynach sekwencyjnych i równoległych. W literaturze te dwa prawa są wyprowadzane oddzielnie lub ich powiązania są przedstawiane w dość skomplikowany sposób. W niniejszym artykule wykazano, że za pomocą prostych przekształceń arytmetycznych w kilku wierszach prawo Gustafsona-Barsisa można uzyskać z prawa Amdahla i odwrotnie.

Słowa kluczowe: obliczenia równoległe, obliczenia rozproszone, przyspieszenie, miary wydajności

Andrzej Karbowski, DSc, PhD
andrzej.karbowski@pw.edu.pl
ORCID: 0000-0002-8162-1575

M.Sc. 1983, Ph.D. 1990 from WUT, D.Sc. 2012 from WUT. With WUT since 1983. Research visitor: Politecnico di Milano and Universita di Genova, 1992, Edinburgh Parallel Computing Centre, 2000. Interests: Large scale systems, distributed computations, optimal control and management in risk conditions, decision support systems, neural networks, environmental systems management, control and decision problems in computer networks.