USES AND ABUSES OF AMDAHL'S LAW*

S. Krishnaprasad
Mathematical, Computing, and Information Sciences
Jacksonville State University
Jacksonville, AL 36265
Email: skp@jsucc.jsu.edu

ABSTRACT

Amdahl's law has been widely used by designers and researchers to get a rough estimate of performance improvement when alternate designs and implementations are attempted. It gives a simple relationship between the nature of performance improvement and the problem characteristics. The negative way the original law was stated [Amd67] contributed to a good deal of pessimism about the nature of parallel processing. But, after observing remarkable speedups in some large-scale applications, researchers in parallel processing started wrongfully suspecting the validity and usefulness of Amdahl's law. In this paper we present the many uses of Amdahl's law as well as some of its abuses.

1. INTRODUCTION

Amdahl's law, as originally formulated [Amd67], is a simple and direct argument showing that the inherently serial portion of a computation imposes a limit on the potential speedup of parallel processing. It is a remarkably simple and elegant law about the nature of performance improvement. It illuminates many topics in computer science and engineering.

Amdahl was pessimistic about the success of parallel processing. The following is a quote from Gene Amdahl [Amd67]:

> *"For over a decade prophets have voiced the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers in such a manner as to permit co-operative solution...The nature of this overhead (in parallelism) appears to be sequential so that it is unlikely to be amenable to parallel processing techniques. Overhead alone would then place an upper limit on throughput of five to seven times the sequential processing rate, even if the housekeeping were done in a separate processor...At any point in time it is difficult to foresee how the previous bottlenecks in a sequential computer will be effectively overcome."*

But, researchers in massively parallel computation have observed impressive linear speedups for some large applications [Gus96]. This lead to wrongfully suspecting the usefulness of Amdahl's law. Also, using incorrect performance measures, it is very easy to break the law and even achieve super-linear speedups.

In this paper we present the usefulness of Amdahl's law as applied to many topics in computer science and engineering. Some of the topics include cache memory design, instruction set design, processor design, vector processing, and multiprocessing. Some of the misconceptions and abuses of the law are also presented. In Section 2 we discuss Amdahl's law as applied to multiprocessing, highlighting some of the abuses and suspicions. Section 3 shows the use of Amdahl's law in memory hierarchy design. In Section 4 we present the law's applicability in instruction set design, processor design, and vector processing. Section 5 includes some concluding remarks.

2. AMDAHL'S LAW AND MULTIPROCESSING

The original formulation of Amdahl's law [Amd67] states the impact of inherently sequential portion of a task on the speedup during multiprocessing. Suppose f represents the fraction of the task that is inherently sequential then using N processors the speedup is given by

$$\mathbf{S = \frac{1}{(f + (1-f) / N)}}$$

When **f=0, S = N**, resulting in an ideal linear speedup.

When **f=0.2, S<5**, independent of **N**.

When **f=0.5, S<2**, independent of **N**.

For large **N, S =(1/f)**, independent of **N**.

This relationship generates pessimism regarding the viability of massively parallel processing especially if we overestimate the value of the fraction f. But, researchers in parallel computation community started suspecting the usefulness and validity of Amdahl's law after observing impressive linear speedups in some large applications. Gustafson (1988, Sandia National Lab) reported near-linear speedups on 1024-processor hypercube for three practical applications: beam stress analysis, surface wave simulation, and unstable fluid flow. This lead to suspecting the nature of Amdahl's original formulation. For example, Gustafson [Gus96]

argues that Amdahl's law is inappropriate for current approaches to massively parallel processing and suggests an alternate *scaled speedup* measure. E. Barsis (Sandia National Lab) proposed a scaled speedup formula, which is often referred to as Gustafson's law. This is stated as follows: if the fraction of time spent by the sequential part on a parallel system is **g**, then with **N** processors the scaled speedup is **S = g + (1-g)*N**, a simple linear relationship. A note by Stephen J. Williard [Will99] elaborates on this law.

Yuan Shi (1996), in an illuminating article [Shi96], shows that the Gustafson's law and Amdahl's law are not two separate laws and in fact proved the equivalence of the two laws. Gustafson had mistakenly used the value of g as the value for f in Amdahl's law and incorrectly suspected the Amdahl's law. The two fractions, f and g, are shown [Shi96] to be related as

$$\mathbf{S = \frac{1}{1 + (1-g)*N/g}}$$

For example, Gustafson used **g=0.004** and calculated the scaled speedup as 1020 with **N=1024**, but using Amdahl's law got a speedup of 201 using the value of **g** for **f**. If he had used the correct value for **f** corresponding to **g**=.004, which is 0.0000039, then he would have gotten the same speedup of 1020 using Amdahl's law! Thus there is nothing pessimistic about Amdahl's law. In practice, for several applications, the fraction of the serial part happens to be very, very small thus leading to near linear speedups.

It is also very easy to use wrong performance measures and arrive at super-linear speedups during multiprocessing. This often leads to suspecting the Amdahl's law. For example, consider the sorting problem based on element comparisons. Suppose we use selection sort on a list of **N** elements. Worst case number of comparisons is $\mathbf{N^2}$ for this sequential algorithm.

Suppose we use **K**-fold parallel processing by dividing the list into **K** sublists, each of size **N/K**. The parallel implementation of sorting by performing selection sort on each sublists (in parallel) and merging the **K** sublists needs the following number of comparisons:

$$\mathbf{N + (N/K)^2 + N*(K-1)}$$

This gives a speedup of

$$\mathbf{S = \frac{N^2}{N + (N/K)^2 + N*(K-1)} = \frac{1}{1/N + 1/K^2 + (K-!)/N}}$$

For large **N**, $\mathbf{S = K^2}$ which seems to "break" the law yielding super-linear speedup!

The fallacy here is that we are comparing two different types of algorithms. The parallel implementation version is essentially a mergesort while the sequential version is a selection sort. The total number steps performed in the parallel version and the sequential version are not the same. Indeed, if they were the same, we would get the correct sub-linear speedup. The same divide-and-conquer method if run on a single processor would take $\mathbf{N + K*(N/K)^2 + N*(K-1)}$ steps. This gives a correct speedup of

$$S = \frac{N + K*(N/K)^2 + N*(K-1)}{N + (N/K)^2 + N*(K-1)} = K \qquad \textbf{For large } N$$

3. USE IN MEMORY HIERARCHY DESIGN

One of the goals in computer design is to provide a large memory and a fast memory. Though this appears to be a difficult task, incorporating a hierarchy of memory systems has solved this problem. The principle of locality (both temporal and spatial) has been successfully exploited in the memory hierarchy of modern computer systems ([Hen96], [Pat97]). With the growing size of software applications run on current machines, there is a corresponding demand for larger main memory (hundreds of MBs) and cache memory (hundreds of KBs.) In fact, to meet the fast clock speeds (gigahertz) of modern processors multilevel cache memory is used to enhance the performance of the traditional single-level cache. Level I cache is part of the processor chip module and Level II cache is typically on an off-chip module.

Amdahl's law can be applied to get a rough estimate of the performance of the memory systems in the hierarchy. For example, the speedup of main memory access due to a single-level cache memory is given by the formula

$$S = \frac{"}{H + (1 - h) * "}$$

h: cache hit ratio

": T_m/T_c

T_m: main memory access time

T_c : cache memory access time

If **h=0.5**, then **S < 2**, independent of how fast the cache is. Thus the hit ratio limits the performance of the access. This may sound pessimistic but, in practice, software applications depict a remarkably high degree of spatial and temporal localities leading to very high cache hit rates. This in turn results in good speedup and performance. David O'Neal of National Center for Supercomputing Applications [One99] presents some interesting performance results for multilevel cache memory systems.

4. USE IN INSTRUCTION SET AND PROCESSOR DESIGN

To predict the performance enhancement due to an improved feature it is convenient to restate the essence of Amdahl's law as follows [Pat97]:

$$\text{Speedup} = \frac{\text{Execution time before a feature is improved}}{\text{Execution time after the improved feature}}$$

Here, the fraction of task that does not use the feature limits the performance or speedup. This should guide the designers in the design process. For example, before an attempt is made to improve the speed of multiplication operation, one should know roughly the fraction of time a task performs multiply operations. Suppose a task takes 100 seconds to run on a processor. Say 40% of this time is consumed by multiply operations (which we will try to improve). Since 60% of the task is unaffected by the improvement, the speedup is given by

S = 100 / (60+(40/K)) where multiply operation is made **K**-times faster.

Thus, **S < 1.67** independent of **K**. This provides useful feedback to the designer. A guiding rule is that frequently used instructions should be improved. But the designer should be aware of the performance limits due to other slower instructions in the program. Whether to improve the performance of integer arithmetic operations or floating point arithmetic operations involves similar investigation.

Similar in spirit, the slowest stage in the pipeline limits the overall pipeline performance. The lowest-performance component on the input/output path limits the I/O system throughput. Performance of suitable applications on vector processors is often dramatic even though there may be some inherently scalar operations. This again is due to the fact that for large problem sizes the value of the fraction f in Amdahl's law is very, very small.

5. CONCLUSIONS

In this paper we have presented the usefulness of Amdahl's law as applied to many topics in computer science and engineering. When it is used correctly and in proper context, it does give performance improvement estimates that are often useful to the designer. The original formulation of the law was apparently pessimistic with regard to parallel processing. But impressive multiprocessing speed-up results lead to the formulation of an apparently new law called Gustafson's law which was more optimistic. Many misunderstandings existed in the parallel processing community regarding the nature of performance improvement and the applicability of these two laws. Yuan Shi demonstrated the equivalence of the two laws thus clearing up many of the misunderstandings.

REFERENCES

[Amd67] Amdahl, G.M., "Validity of the single-processor approach to achieving large scale computing capabilities," Proceedings of AFIPS Conference, 1967, pp. 483-485.

[Gus96] Gustafson, J.L., "Reevaluating Amdahl's Law," Ames lab web link http://www.scl.ameslab.gov/Publications/AmdahlsLaw/Amdahls.html, June 1996.

[Hen96] Hennessy, J.L. and Patterson, D.A, Computer Architecture: A Quantitative Approach, 2nd Ed., Morgan Kaufmann, 1996.

[One99] O'Neal, David, "On Microprocessors, Memory Hierarchies and Amdahl's Law," Carnegie Mellon University, Pittsburgh Supercomputing Center, Pittsburgh, PA. See web link http://archive.ncsa.uiuc.edu/EP/CSM/presentations, November 1999.

[Pat97] Patterson, D.A. and Hennessy, J.L., Computer Organization & Design: The Hardware/Software Interface, 2nd Ed., Morgan Kaufmann, 1998.

[Shi96] Shi, Y., "Reevaluating Amdahl's Law and Gustafson's Law," Computer and Information Sciences department, Temple University, Web link http://joda.cis.temple.edu/~shi/docs/amdahl/amdahl.html, October 1996.

[Will99] Williard, S.J., "The Gustafson-Baris law," see the web link http://home.wlu.edu/~whaleyt/classes/parallel/topics/Gustafson.html.