Article

International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
2015, Vol. 29(3) 320–330
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342015572031
hpc.sagepub.com
SAGE

A tale of two laws

Michael T Heath

Abstract

As Amdahl's law and Moore's law reach their 50th anniversaries, we review the roles they have played in shaping both perception and reality in high-performance computing. Along the way, we also attempt to clarify some misconceptions that have surrounded both of these highly influential but not always fully appreciated "laws."

Keywords

Amdahl's law, Moore's law, parallel computing, scalability, speedup, efficiency, Dennard scaling

1. Amdahl, Moore, and you

Amdahl's law and Moore's law have had a profound impact on the evolution of high-performance computing (HPC). The two laws are of similar vintage, both having been formulated in the mid-1960s at the dawn of the modern era of HPC, but they differ markedly in many other respects. Amdahl's law expresses a mathematical relationship whose conclusion is logically unassailable, but the applicability of its hypotheses has been the subject of considerable debate. Moore's law, on the other hand, expresses an empirical observation about trends in technological capability and manufacturing efficiency, but its extraordinary predictive accuracy has depended to a large extent on self-fulfilling prophecy. Amdahl's law provoked widespread skepticism concerning the ultimate potential of parallel computing, whereas Moore's law engendered great optimism for the future of computing in general and eventually enabled the present ubiquity of computers in daily life. Accordingly, heroic efforts have been expended both to circumvent Amdahl's law and to prolong Moore's law. As they reach their 50th anniversaries, we review the roles played by these two highly influential "laws" in shaping both perception and reality in HPC. We examine how and why each law came to be formulated, how practitioners have grappled with their implications in the intervening years, and why both remain relevant today.

2. Parallel scalability

Serious interest in parallel computing dates from the mid-1960s, with ILLIAC IV (designed 1965–66, delivered 1971) often cited as the first "massively" parallel computer, designed to have 256 processing elements (although only 64 were actually fabricated) (Hord, 1982). By this time transistors had replaced the vacuum tubes and relays in the logic circuits of the first generation of electronic digital computers, but integrated circuits (invented in 1958) were still in their infancy, so computers were made of individually wired components that filled large cabinets or whole rooms. But improvements in the speed of such "macro" computers were beginning to level off, due to physical constraints such as the speed-of-light limit (about one foot per nanosecond) on signal propagation, thus the growing interest in applying many processors in concert to solve a single problem.

The primary motivation for parallel computing is to increase computational *speed* (work per unit time) by spreading the work over more processors. The goal might be:

- to solve a given problem (same work) in less time;
- to solve a larger problem (more work) in the same time; or
- to gain sufficient capacity to solve ever larger problems regardless of time.

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

Corresponding author:
Michael T Heath, Department of Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana IL 61801, USA.
Email: heath@illinois.edu

Qualitatively, *scalability* refers to the relative effectiveness with which additional processors can be used: as the number of processors increases, does performance increase *commensurately*? As we will see, however, this concept is not easy to quantify, and often difficult to achieve. To help quantify scalability, the following quantities are conventionally defined for a given problem:

- processors, p = number of processors used;
- execution time, $T(p)$ = elapsed wall-clock time using p processors;
- cost, $C(p) = pT(p)$, measured in processor-seconds, processor-hours, etc.;
- speedup, $S(p) = T(1)/T(p)$;
- efficiency, $E(p) = C(1)/C(p) = T(1)/(pT(p)) = S(p)/p$.

The goal then becomes to solve the problem p times as fast, $S(p) = p$, by using p processors, which is equivalent to achieving 100% efficiency, $E(p) = 1$. In practice, such ideal performance is unattainable (barring effects of cache or chance), and one settles for some lower level of efficiency $E(p) < 1$. Arbitrarily low efficiency is obviously unacceptable, however, so a minimal criterion for scalability is that $E(p)$ be bounded away from zero as $p \rightarrow \infty$. As we will see, however, even this seemingly innocuous criterion can be difficult or impossible to meet in many practical situations.

2.1 Empirical speedup considered harmful

Note that speedup and efficiency depend on the serial time $T(1)$, but what does $T(1)$ really mean? Plausible answers include:

- execution time for the parallel code using one processor;
- execution time for the corresponding serial code using one processor (i.e. the same basic algorithm but without any parallel overhead); or
- execution time for the "best" serial algorithm;

but none of these is truly satisfactory. Running the parallel code on a single processor generally incurs significant overhead for communication and synchronization that is unnecessary when $p = 1$. Even if this parallel overhead is removed or bypassed in a serial version of the code, the underlying parallel algorithm may not be optimal when $p = 1$. The criterion for "best" serial algorithm may not be clear-cut (least computing time, least memory usage, etc.), and the optimal choice may depend on particular problem features, such as problem size. Moreover, using a fundamentally different algorithm for the special case $p = 1$ introduces a disruptive discontinuity that resets the particular level of efficiency but usually does not alter the overall trend in relative performance as p varies.

Further complicating matters, the problem of interest may not fit in the memory of a single processor, making direct measurement of $T(1)$ impossible. One alternative approach is to use the smallest p for which the problem fits in memory as a benchmark for computing *relative* speedup, but this particular choice seems arbitrary, and in any case cache behavior may differ greatly as p varies, yielding erratic computed speedups that are not very meaningful.

But who cares about serial execution time anyway? Today's users are *not* interested in what happens when they go from 1 processor to 100,000 processors, they are interested in what happens when they go from 100,000 processors to 200,000 processors. Thus, they care about the behavior of their parallel code, *not* that of some mythical serial code. We conclude that because of their dependence on often ill-defined or irrelevant serial execution time, *speedup* and *efficiency* as conventionally defined are often *not* useful measures of parallel performance for real programs.

2.2 A better alternative

Rather than using a *derived* measure such as speedup to assess scalability, a better alternative is simply to plot raw execution data, specifically a log–log plot of $T(p)$ versus p, for a systematic sequence of runs, varying the number of processors (Van de Velde, 1994). Two common instances of this approach are:

- *Strong scaling*: for a fixed problem, a straight line with slope -1 indicates good scalability, whereas any upward curvature away from that line indicates limited scalability.
- *Weak scaling*: for a sequence of problems with a fixed amount of work per processor, a horizontal straight line indicates good scalability, whereas any upward trend of that line indicates limited scalability.

An example of strong scaling is shown in Figure 1, where the bullets represent execution times for a fixed problem using various numbers of processors (data generated synthetically). The solid line is a least-squares fit of a straight line with slope -1 to the given data points (plotting such a line is not strictly necessary, but it can be a helpful visual aid). Note that the smallest number of processors for which the problem fits in memory happens to be 16, but no notion of relative speedup is required. The tailing off in scalability is evident for $p > 10^4$.

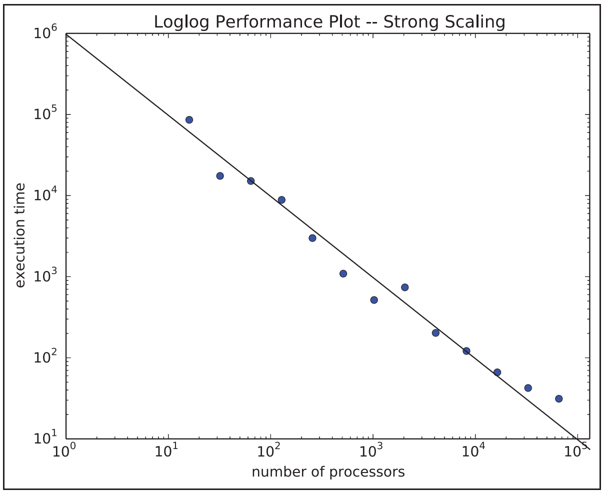

Figure 1. Log–log performance plot.

3. Amdahl's law

The foregoing observations were aimed at empirical performance analysis for real parallel programs. Theoretical performance models can often provide valuable insight into the scalability of the underlying parallel algorithms independent of any specific implementation. By assuming away memory limitations, cache effects, etc., the conventional definitions of concepts such as speedup and efficiency can be applied meaningfully and unambiguously in this theoretical context. One of the earliest and easily the most famous such theoretical performance model forms the basis for Amdahl's law. Amdahl's original paper (Amdahl, 1967) provided only a verbal description of the relationship that later became known as Amdahl's law, but it is equivalent to the following mathematical statement.

Amdahl's law. If a fraction s of the work for a given problem is serial, with $0 < s \leq 1$, while the remaining portion, $1 - s$, is p-fold parallel, then

$$T(p) = sT(1) + (1-s)\frac{T(1)}{p}$$
$$S(p) = \frac{T(1)}{T(p)} = \frac{1}{s + (1-s)/p}$$
$$E(p) = \frac{T(1)}{pT(p)} = \frac{1}{sp + (1-s)}$$

and hence $S(p) \to 1/s$ and $E(p) \to 0$ as $p \to \infty$. If the serial fraction s exceeds 1%, for example, then the speedup can never exceed 100 no matter how many processors are used, and the efficiency becomes arbitrarily low as increasingly many processors are used.

Amdahl's pessimistic conclusion of limited speedup and poor efficiency should come as no surprise, at least qualitatively, as no fixed, finite amount of work can be shared profitably by arbitrarily many processors, with the obvious negative ultimate consequences for speedup and efficiency. Nevertheless, Amdahl's quantifying of these effects, albeit in the context of a very simple computational model, served as an eye opener that justified a continued emphasis on further improving serial computing speeds (aided largely by Moore's law, as we will see), as well as extensive efforts to circumvent Amdahl's unwelcome strictures.

Since Amdahl's simple derivation is valid mathematically, to avoid his sobering conclusion one must evade his hypotheses. One could argue, for example, that the assumed dichotomy between serial and p-fold parallel portions is too simplistic. But in fact it is not an unreasonable proxy for the inevitable lack of perfect parallelism in most computational problems due to precedence constraints, communication overhead, etc. To illustrate, consider these typical examples of parallel performance models for some common numerical computations, where n measures problem size in basic work units:

- summation of n numbers, $T(p) \approx n/p + \log p$;
- solution of a triangular system of order n, $T(p) \approx n^2/p + n$;
- LU factorization of a matrix of order n, $T(p) \approx n^3/p + n$;
- solution of a tridiagonal system of order n, $T(p) \approx (n \log n)/p + \log p$.

These examples were chosen to represent a number of realistic situations that arise often in practice. Summation is ubiquitous, of course, but a similar complexity model, having a perfectly parallel phase followed by a reduction whose length grows at least logarithmically with p, applies to many other computations, such as computing inner products, convergence tests for iterative methods, and Monte Carlo simulations in which independent trials are computed simultaneously in parallel and then all of the results must be averaged to produce the final result. Solution of triangular linear systems and LU factorization with an $n \times n$ matrix typify many problems for which the bulk of the computation is highly parallel, but due to precedence constraints there is a serial thread (here of length n) running through the computation, the length of which grows with problem size. Solution of tridiagonal systems incurs additional work in order to introduce parallelism into an otherwise inherently sequential computation.

Log–log performance plots for these examples (normalized by the total amount of work in each case) are shown in Figure 2, along with performance plots given by Amdahl's law for various serial fractions s. The behavior and scalability of the real examples is visually indistinguishable from those for Amdahl's law with an appropriately chosen serial fraction s. In each case, the serial fraction is an effective proxy for the terms in the

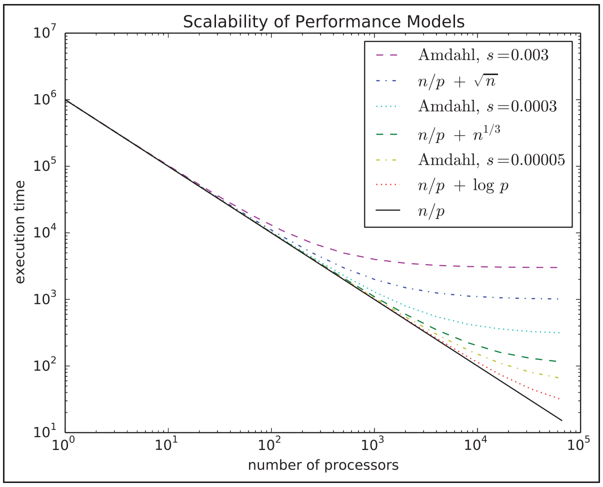

Figure 2. Scalability of various performance models.

performance model that do not decrease as p increases. In fact, for any parallel computation one can infer an effective serial fraction empirically from measured speedup, as in the Karp–Flatt metric (Karp and Flatt, 1990), although empirical speedup has its own shortcomings, as we have already seen. Thus, one cannot deny the applicability of Amdahl's law simply by arguing that the performance model underlying it is too unrealistic, and therefore one must grapple with the (effective) serial fraction for a given problem of interest.

3.1 Consequences of Amdahl's law

Fortunately, for most problems the effective serial fraction *decreases* as problem size increases, and hence Amdahl's limits on speedup and efficiency become correspondingly less onerous. Thus, the most common response to the limited speedups and declining efficiency implied by Amdahl's law is to solve increasingly larger problems as the number of processors increases (Gustafson, 1988; Singh et al., 1993). However, the *rate* of growth in problem size, relative to the growing number of processors, is constrained: if the problem size grows too slowly, then efficiency may decline unacceptably, but if the problem size grows too rapidly, then execution time may grow unacceptably. Thus, solving ever larger problems is not a panacea, but a two-edged sword that necessitates walking a fine line between declining efficiency and growing execution time, which may not be possible in practice, as we will see next.

Amdahl's key observation was that any inherently serial work imposes a lower bound on execution time that is independent of the number of processors. Increasing the problem size can reduce the *proportion* of serial work, but it often increases the total *amount* of effectively serial work, and thus the lower bound on execution time *increases* with problem size. Increasing problem size also increases the parallel portion of the execution time unless the problem size grows no faster than the number of processors. Thus, total execution time *increases* unless the problem size grows more *slowly* than the number of processors, yielding declining efficiency.

One can infer from Amdahl's law how rapidly the serial fraction must decrease with increasing problem size in order to achieve a desired efficiency. For example, to maintain a given constant efficiency E, $0 < E \leq 1$, Amdahl's law requires that

$$s = \frac{1-E}{E} \cdot \frac{1}{p-1}$$

Applying this result to a performance model of interest then determines the corresponding minimum growth rate in problem size. For example, if $T(p) \approx n/p + \log p$, then the problem size n must grow proportionally to $p \log p$, causing execution time to grow like $\log p$. More generally, this observation leads to one of the most useful measures of scalability, the *isoefficiency function*, which is defined to be the minimum rate at which the problem size (as measured by the total amount of computational work) must grow in order to maintain constant efficiency as p grows (Grama et al., 1993). The growth rate of the isoefficiency function with respect to the number of processors then characterizes the *degree* of scalability:

- $\Theta(p)$, highly scalable;
- $\Theta(p \log p)$, reasonably scalable;
- $\Theta(p\sqrt{p})$, moderately scalable;
- $\Theta(p^2)$ or higher, not scalable.

For a given problem, its isoefficiency function divided by p gives the growth rate of the execution time as p increases. For example, if the isoefficiency function is $\Theta(p^2)$, then doubling the number of processors also doubles the execution time, which is clearly untenable. For any slower growth rate, however, constant efficiency cannot be maintained, and efficiency necessarily declines.

We have just seen that except for the usually unattainable ideal isoefficiency function $\Theta(p)$, execution time necessarily increases, and possibly quite rapidly, unless one is willing to accept ever lower efficiency. Incurring ever longer turnaround times in order to meet efficiency goals may or may not be acceptable, depending on the purpose of the computation. Turnaround time may be limited by, among other factors:

- hard real-time constraints, as in control systems or weather prediction;
- practical working schedules requiring say hourly or daily turnaround;
- mean time between failures of computing equipment.

Even when limiting turnaround time is not crucial, there may still be little or no value in solving ever larger problems. For example, over-resolving a problem beyond the accuracy needed or beyond that warranted by the available data solely to achieve higher parallel efficiency is of dubious value.

For example, if you are trying to intercept an incoming missile, then it is not helpful to assume 10 or 100 incoming missiles instead, because you will not be around to brag about the improved efficiency with which you can track them. For weather prediction, efficiency could be improved through increasing problem size by:

- refining the spatial resolution of the computational mesh;
- enlarging the geographic region covered; or
- adding more detailed physics;

but none of these will *decrease* turnaround time and are instead likely to *increase* it, eventually threatening the ability to make timely predictions. Additional processors can potentially be used more helpfully to perform sensitivity analysis or uncertainty quantification, thereby improving the quality and reliability of results, but again this yields no reduction in execution time.

Solving ever larger problems may be warranted if the additional scientific payoff is sufficiently great, but in that event one must accept ever-increasing execution times. On the other hand, solving ever larger problems just to meet efficiency goals is of questionable value. This point can be succinctly summarized by a variation on Richard Hamming's famous dictum, "*the purpose of computing is insight, not numbers*," namely, "*the purpose of computing is insight, not keeping processors busy*."

Alternatively, instead of maintaining constant efficiency, one could insist on fixed execution time (Worley, 1990; Gustafson, 1992). But a similar analysis shows that in order to maintain constant execution time, the problem size (as measured by total computational work) must grow more *slowly* than the number of processors, and hence efficiency inevitably declines.

In summary, the constraints and tradeoffs implied by Amdahl's law remain relevant today, almost 50 years later. Heroic efforts by hardware designers and users have made large-scale parallel computers the workhorses of scientific computing. Their success did not come by somehow overturning Amdahl's law, however, but by learning to deal with its consequences creatively and effectively. Indeed, Amdahl's fundamental message of ruthlessly limiting parallel overhead becomes even more compelling as processor counts exceed one million.

3.2 Categorizing speedup

Before leaving the topic of parallel speedup, we note an unfortunate inconsistency in the terminology commonly used to describe it. According to conventional terminology in parallel computing:

- *linear* speedup means $S(p) = p$;
- *superlinear* speedup means $S(p) > p$;
- *sublinear* speedup means $S(p) < p$.

However, according to standard mathematical terminology:

- *linear* speedup means $S(p) = \alpha p$ for some constant $\alpha > 0$;
- *superlinear* speedup means $S(p) = \alpha p^{\beta}$ for some constants $\alpha > 0$ and $\beta > 1$;
- *sublinear* speedup means $S(p) = \alpha p^{\beta}$ for some constants $\alpha > 0$ and $\beta < 1$.

This marked discrepancy in terminology leads to potentially confusing anomalies, such as

- mathematically linear speedups with $\alpha = 2$ or $\alpha = 0.5$, for example, would conventionally be deemed superlinear or sublinear, respectively;
- mathematically superlinear speedup with $\alpha \ll 1$ and $\beta > 1$ would conventionally be deemed sublinear, at least initially;
- mathematically sublinear speedup with $\alpha \gg 1$ and $\beta < 1$ would conventionally be deemed superlinear, at least initially.

Similar anomalies are illustrated in Figures 3–6. Figure 3 shows three standard speedup curves, one that is mathematically linear ($\beta = 1$), another that is mathematically sublinear ($\beta = 0.5$) but falls on the superlinear half of the plot using conventional terminology, and a third that is mathematically superlinear ($\beta = 2.0$) but falls on the sublinear half of the plot using conventional terminology. The same performance data are shown in a log–log performance plot in Figure 4, which makes it clear (from their respective slopes) which curve is really linear, sublinear, or superlinear. Figure 5 shows three standard speedup curves, each of which is mathematically linear, but which would be conventionally categorized as superlinear, linear, or sublinear, respectively. The same performance data are shown in a log–log performance plot in Figure 6, which makes it clear that all three are actually linear (with slope -1), and they differ only in their (constant) relative efficiencies.

This distinction has implications that are not merely semantic. Linear speedup in the conventional sense is not a realistic goal, as it requires achieving 100% efficiency. Linear speedup in the mathematical sense is a more reasonable goal, however, as it merely implies constant efficiency E for some $0 < E \leq 1$. As we have seen, this goal is often achievable, but it may require

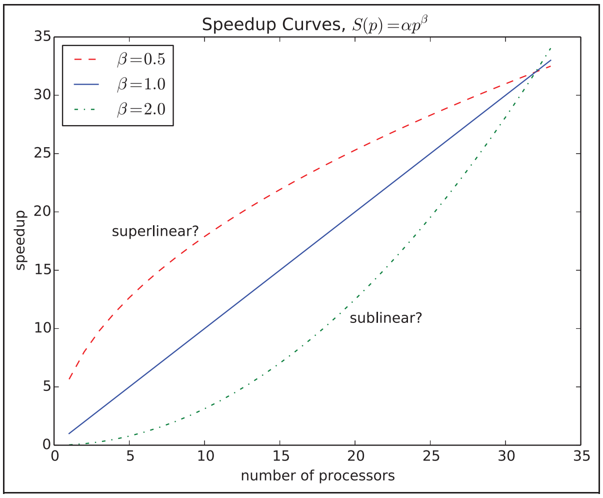

Figure 3. Speedup curves for various performance models.

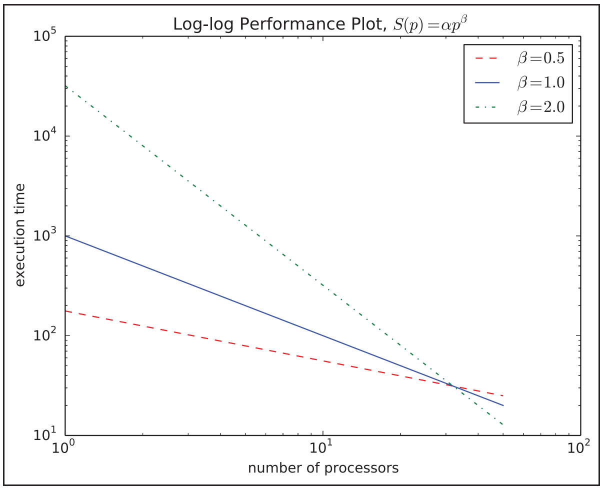

Figure 4. Log–log performance plot for performance models shown in Figure 3.

solving ever larger problems with increasing turnaround times.

4. Moore's law

One measure of the impact of Moore's law on modern society is the frequency with which it is cited, especially in the popular press. Unfortunately, it is almost as frequently misstated, or at least misleadingly stated. Here are a couple of typical examples:

> "*Computer processing power doubles about every 18 months.*"
> –http://www.technologyreview.com

> "*Processor speeds, or overall processing power for computers will double every two years.*"
> –http://www.mooreslaw.org

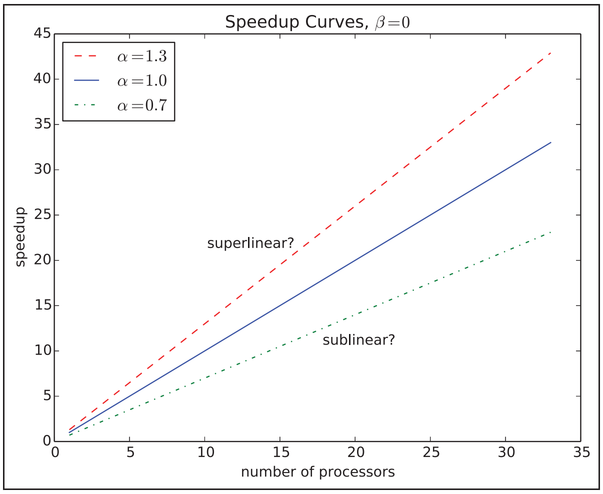

Figure 5. Speedup curves for various performance models.

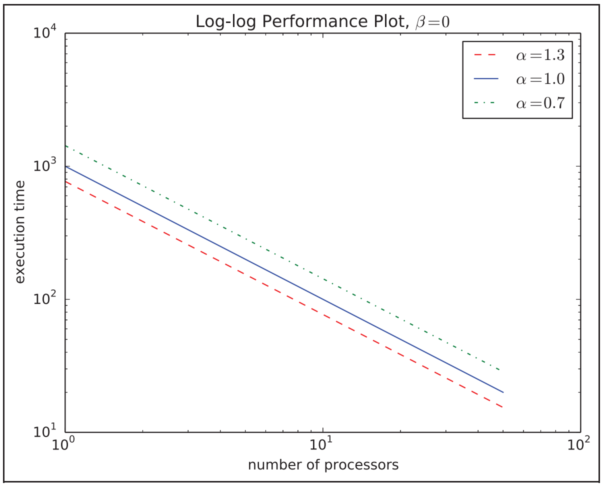

Figure 6. Log–log performance plot for performance models shown in Figure 5.

Here is what Moore actually said (Moore, 1965):

> "*The complexity for minimum component costs has increased at a rate of roughly a factor of two per year* (*see graph*)."

The graph referred to is reproduced here in Figure 7. Although Moore's original statement was actually an observation about the past, he went on to say later in the same paragraph, "there is no reason to believe it [the rate of increase] will not remain nearly constant for at least ten years," or until 1975, as indicated by the extrapolated dashed line on his graph. As it turned out, Moore's projection actually held for more than 50 years (see Figure 8, which begins with the first complete microprocessors in 1971), although this required subsequently adjusting the doubling period to between 1.5 and 2 years.

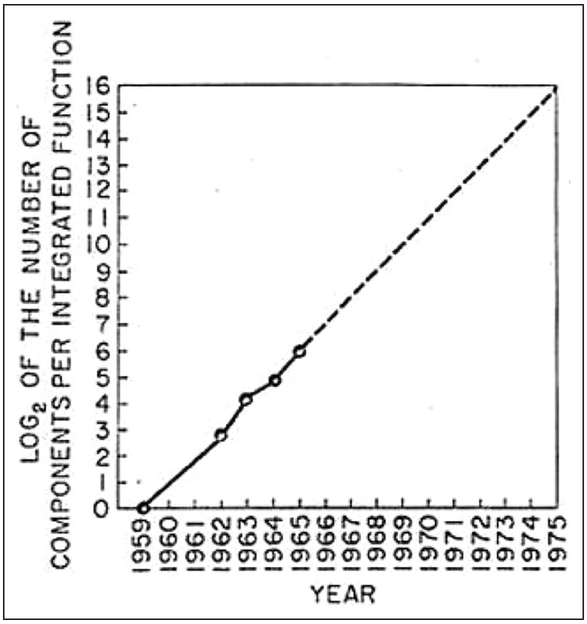

Figure 7. Moore's original 1965 plot showing his extrapolation to 1975 (Moore, 1965).

Note, however, that Moore's statement did not address "processing power" or "processor speeds," but rather *complexity*, by which he meant the "number of components per integrated function," as spelled out explicitly in his graph. In addition to transistors, Moore included resistors, diodes, and capacitors in the component count (transistor count alone was later adopted as the standard measure of complexity). Moore's focus on circuit density was also reflected in the title of the article, "*Cramming more components onto integrated circuits.*"

Note also that Moore did not predict the maximum complexity that would be technologically feasible, but rather the level of complexity that would be most cost effective to manufacture. Thus, it was not only a prediction about future technological capabilities, but also about economic costs and manufacturing efficiencies, as well as indirectly about the size of prospective markets for the resulting products. Apparently, Moore correctly foresaw the fantastically rapid growth in demand for integrated circuits and devices made from them, leading to the billions we see today in a wide array of consumer products. This aspect of Moore's prediction contrasts sharply with the statement made in 1943 (only 22 years earlier) and often attributed to T J Watson,

> "*I think there is a world market for maybe five computers.*"

Moore's law is not a law of nature, however: it remained valid because of economic incentives for chip designers and manufacturers to meet its aggressive goals. In that sense, the remarkable accuracy and longevity of Moore's multifaceted projection have resulted in part from self-fulfilling prophecy.

4.1 Source and implications of Moore's law

Another fascinating aspect of Moore's prediction was its brashness: based on only five data points, and only 6 years after the initial introduction of the planar transistor, he extrapolated a decade into the future (and, as it turned out, well beyond). As the graph in Figure 7 shows, the component count at the time Moore made his prediction had reached all of 64; nevertheless, he boldly projected a further thousand-fold increase by 1975.

Moore's original projection was based simply on plotting the few data points available and observing their approximately linear behavior on a semi-log scale. Subsequently, Moore resolved his projection as a composite of three main trends in the design and manufacturing of integrated circuits (Moore, 1975; Mack, 2011):

- increasing chip area (20% per year);
- decreasing feature size (25% per year in components per unit area);
- improving circuit and device designs (33% per year).

Taken together, these factors produced an improvement of 1.20 × 1.25 × 1.33 = 2× per year. Some of these trends subsequently slowed down, but were partially offset by acceleration of others, leading to an overall doubling rate of 18 months to 2 years instead of the original 1-year doubling rate. Chip area, for example, stopped growing in the late 1990s. The primary driver in recent years has been the continued shrinking of feature size (see Figure 9), while manufacturing cost per unit area has remained roughly constant, so that cost per component decreases.

Moore's original prediction did not address clock speed or power consumption, but later in the same article (Moore, 1965) he observed,

> "*Shrinking dimensions on an integrated structure makes it possible to operate the structure at higher speed for the same power per unit area.*"

But this is not the only option. Due to the greater efficiency of smaller circuits one can:

- maintain the same power but increase clock speed;
- maintain the same power and clock speed but increase functionality; or
- maintain the same clock speed but use less power.

For the first several generations of Moore's law, chip designers chose the first of these options, with clock speeds doubling about every 2 years from roughly 1975 to 2005, then leveling off at about 3 GHz due primarily to limits on power (heat) dissipation (see Figure 10).

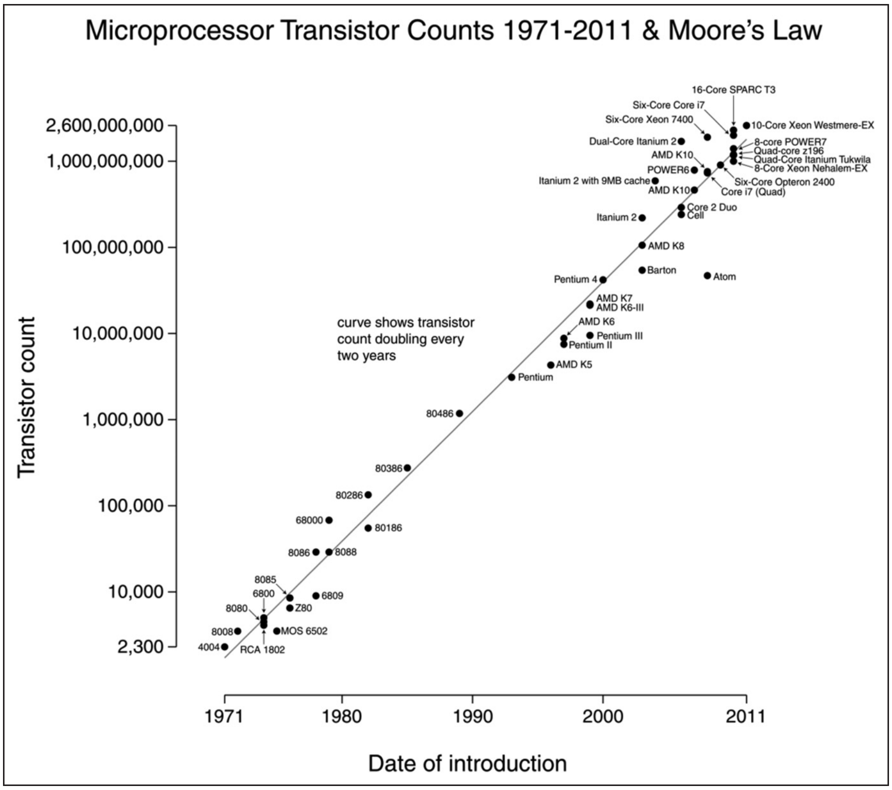

Figure 8. Plot of microprocessor transistor counts 1971–2011 showing continuation of Moore's law (http://en.wikipedia.org/wiki/Moore's_law).

Meanwhile, Moore's original projection (concerning complexity, not speed) has continued to hold more or less to the present, although it too may be beginning to wane. With clock speeds having leveled off, the second option has become the more recent trend, using those additional transistors to increase functionality (e.g. multicore chips). With the advent of millions of mobile devices, however, the third option (lower power) may become the dominant trend in order to prolong battery life.

In 1974, near the end of the period of Moore's original projection, these power issues were analyzed in greater detail by Dennard et al. (1974), who observed that both voltage and current scale proportionally with feature size, so power scales proportionally with area, leading to a doubling of performance per watt approximately every 2 years. Their analysis was specifically for MOSFETs (metal-oxide-semiconductor field-effect transistors), but their conclusions were widely taken to be generally applicable to integrated circuits more broadly and were adopted by the semiconductor industry as a roadmap for future development. However, as feature size continued to shrink in accordance with Moore's law, this so-called Dennard scaling began to break down around 2005 due to increasing current leakage and thermal noise, leading to the current stagnation in clock speeds (Figure 10).

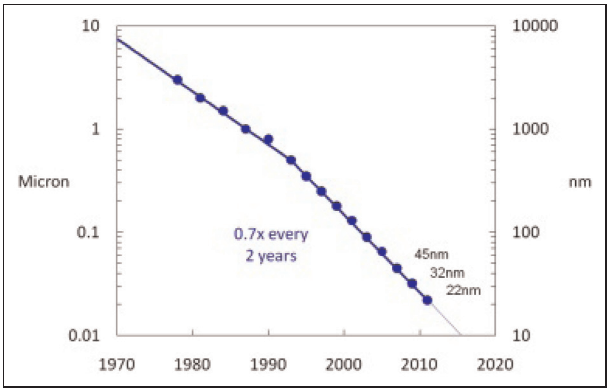

Figure 9. Plot showing decreasing feature size of integrated circuits in accordance with Moore's law.

A more comprehensive study was undertaken by Koomey et al. (2010), who examined trends in power consumption since the beginning of electronic digital computers with the ENIAC in 1946. Remarkably, the data they collected for a wide variety of computers (mainframes, supercomputers, desktops, laptops, etc.) showed that energy efficiency, as measured by the number of computations per unit of energy (kWh) dissipated, doubled every 1.57 years from 1946 to 2009 (see Figure 11). Since a significant portion of their data was for computers that predated the advent of integrated

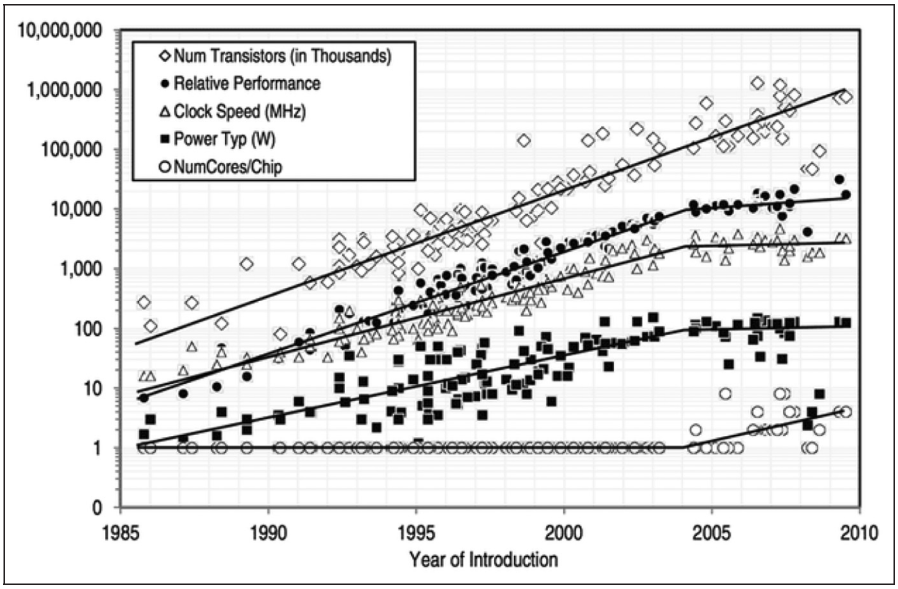

Figure 10. Plot showing flattening of clock speed, power, and relative performance around 2005, with corresponding increase in number of cores per chip (Fuller and Millett, 2011).

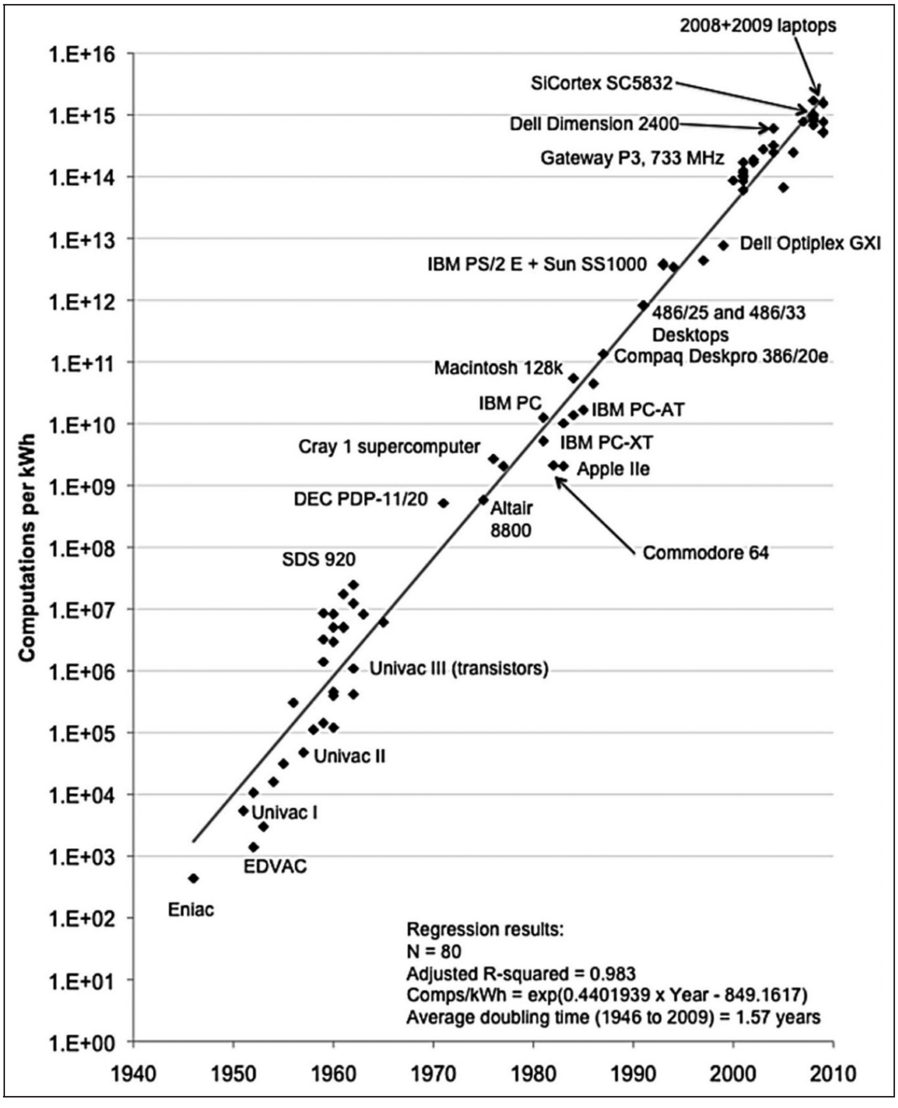

Figure 11. Plot showing computations per kWh 1946–2009 (Koomey et al., 2010).

circuits, not all of the credit can be attributed to Moore's law. Currently, GPUs (graphics processing units) are continuing this doubling of performance per watt with each new generation. And for mobile devices this means that for a fixed computing load the amount of battery required is approximately halved every year and a half. Alas, this fortunate trend cannot continue indefinitely, as conventional computation cannot be made arbitrarily energy efficient according to the Landauer limit (Landauer, 1961).

As we have seen, smaller transistors are better (cheaper, faster, lower power, more reliable), which enabled enormous gains in performance and cost effectiveness to accompany the inexorable shrinking of feature size implied by Moore's law. Unfortunately, however, a threshold is eventually encountered below which smaller becomes worse rather than better, due to power dissipation, current leakage, thermal noise, and ultimately quantum effects, thereby negating all of the advantages listed above except cheaper, and even that becomes questionable as more exotic materials and manufacturing processes are introduced in an attempt to prolong Moore's law, which increases manufacturing costs in the bargain, exacerbating the already exponential growth in the cost of fabrication facilities. Having now reached this threshold, the future fate of Moore's law has become less important, as the payoff from further shrinking of feature size becomes increasingly marginal instead of the pure win of the preceding era. One can debate whether or when Moore's law has or will end, but in any case the 40-year free lunch appears to be over, regardless of whether another generation or two of shrinkage may prove to be possible at reasonable cost.

The relationships formulated by Moore and by Dennard were destined to break down eventually, due to both physical and economic constraints. When interest in large-scale parallel computing first arose in the mid-1960s, reasons cited were often physical limits on single processor speeds, such as the speed-of-light limit on signal propagation, thereby forcing consideration of parallel computing. Nevertheless, thanks to the trends noted by Moore and Dennard, single-processor speeds (and especially their cost effectiveness) continued to advance exponentially for the next 40 years, confounding many early attempts at building commercially viable parallel computers. Ironically, when the day of reckoning eventually came, it was barriers imposed by 19th century physics (thermodynamics) rather than 20th century physics (relativity and quantum mechanics) that first proved insuperable.

The economic and societal impact of the 50-year reign of Moore's law is inestimable, with computers and other electronic devices now permeating every aspect of daily life in a manner unimaginable 50 years ago. To cite a familiar but telling example, the exponential miniaturization, high performance, and low cost that followed from Moore's law have made it possible to buy for a few hundred dollars a cell phone that fits in a pocket and runs for days on battery power, yet has an order of magnitude more processing power and memory capacity than a US$10 million, 5.5 ton, 115 kW Cray-1 supercomputer that was the world's fastest from 1976 until 1982. Whatever may lie ahead in the "post-Moore" era, we should be grateful that Moore's law persisted long enough to provide the extreme degree of interconnectedness, information access, and productivity enhancement that we now enjoy, not to mention the availability of high-end computers for scientific computing at the petascale and beyond.

5. Epilogue

There has been an interesting interplay between Amdahl's law and Moore's law throughout their mutual infancy, adolescence, and maturity. Consider the very first sentence of Amdahl's paper (Amdahl, 1967):

> "*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 cooperative solution.*"

Amdahl's deprecation of parallel processing was explicitly intended to bolster the case for developing faster sequential processing, as indicated by the title of his paper ("*Validity of the single processor approach* …"). Fortunately for Amdahl's cause, Moore's law would soon enable relentless and relatively easy gains in single-processor performance, thereby forestalling successful commercial deployment of parallel processing for decades.

Ironically, Amdahl's first sentence eerily foreshadowed the situation 40 years later when Dennard scaling ground to a halt and parallel processing once again emerged as the answer to stagnating single-processor performance, but this time with much greater commercial success, thanks to the additional processing cores that Moore's law could still deliver cheaply even if they could not run any faster. Amdahl may yet have the last laugh, however, as many of the impediments to effectively exploiting parallelism that he decried (irregular applications, load imbalance, interprocessor coordination, slowed convergence rates, etc.) remain challenging today.

Acknowledgement

A preliminary version of this paper was presented as an invited talk at SC14 in New Orleans, Louisiana, 19 November 2014.

References

Amdahl GM (1967) Validity of the single processor approach to achieving large scale computing capabilities. In: *AFIPS Spring joint computer conference*, pp. 483–485.

Dennard RH, Gaensslen FH, Yu HN, Rideout VL, Bassous E and LeBlanc AR (1974) Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9(5): 256–268.

Fuller SH and Millett LI (eds.) (2011) *The Future of Computing Performance: Game Over or Next Level?* Washington, DC: National Academy of Sciences.

Grama A, Gupta A and Kumar V (1993) Isoefficiency: measuring the scalability of parallel algorithms and architectures. *IEEE Parallel and Distributed Technology* 1: 12–21.

Gustafson JL (1988) Reevaluating Amdahl's law. *Communications of the ACM* 31: 532–533.

Gustafson JL (1992) The consequences of fixed time performance measurement. In: *Proceedings of the twenty-fifth Hawaii international conference on systems science*

Hord RM (1982) *The ILLIAC IV: The First Supercomputer*. New York: Springer.

Karp AH and Flatt HP (1990) Measuring parallel processor performance. *Communications of the ACM* 33: 539–543.

Koomey JG, Berard S, Sanchez M and Wong H (2010) Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing* 33(3): 46–54.

Landauer R (1961) Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development* 5: 183–191.

Mack CA (2011) Fifty years of Moore's law. *IEEE Transactions on Semiconductor Manufacturing* 24(2): 202–207.

Moore GE (1965) Cramming more components onto integrated circuits. *Electronics* 38(8): 114–117.

Moore GE (1975) Progress in digital integrated electronics. In: *IEEE international electronic devices meeting*, pp. 11–13.

Singh JP, Hennessy JL and Gupta A (1993) Scaling parallel programs for multiprocessors: methodology and examples. *IEEE Computer* 26(7): 42–50.

Van de Velde EF (1994) *Concurrent Scientific Computing*. New York: Springer.

Worley PH (1990) The effect of time constraints on scaled speedup. *SIAM Journal on Scientific and Statistical Computing* 11: 838–858.

Author biography

Michael T Heath is Professor and Fulton Watson Copp Chair Emeritus in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He received his Ph.D. in Computer Science from Stanford University in 1978. His research interests are in scientific computing and parallel computing. He is an ACM Fellow, SIAM Fellow, AIAA Associate Fellow, and a member of the European Academy of Sciences. He received the Taylor L Booth Education Award from the IEEE Computer Society in 2009, and is author of the widely adopted textbook *Scientific Computing: An Introductory Survey*, 2nd edition, published by McGraw-Hill in 2002.