

Trends in Computer Technology

Bill Plummer, University of Missouri-Columbia

The computer revolution is one of the certifiable success stories of recent times. As a commercial venture it has succeeded wildly, spawning Fortune 500 companies worth hundreds of billions in aggregate. In technical terms it has had an unparalleled record of price/performance improvements. It is even environmentally clean, although among the truly addicted, it produces a sedentary life style that must be regarded as fattening!

One may use the amazing performance improvements to construct analogies with more common enterprises. For instance, if the automobile industry had kept pace over the last 15 years, the Rolls-Royce would cost \$1000 and would get 1000 miles per gallon. Of course, in fairness one must note that it would also be the size of a tricycle.

This record of improvement is due to progress in three spheres of activity: the means of construction of the basic electronic building blocks (the "chips"); the means of assembling the basic elements (computer design and engineering); and the means of controlling this hardware (programming and computer science). We will deal briefly with each of these realms, to see where the growth curve appears to be headed. However, the most significant point to be addressed is more in the realm of psychology: how will the increasing power of technology line up with the expectations of the user community?

Hardware

The breakthrough technological advance of the last decade has been the integrated circuit of complementary metal-oxide-silicon (CMOS) construction. This technique has led to the miniaturization, the energy saving, and the cost improvements which we now enjoy. The steady improvements in performance-- speed of operation, number of transistors per chip, etc. -- has depended mainly on manufacturing technique. The basic physics and chemistry have remained the same for a considerable period.

These chips are based on a wafer of silicon which has been doped with a metal oxide, resulting in the type of material known as a semiconductor. Depending on the dopant, the semiconductor is one of two types, p or n, according to whether it has positive or negative carriers of charge. Channels can then be etched in this wafer by lithographic techniques employing electron beams, light, or in the near future x-rays. When channels are etched and alternating p, insulating, and n layers are formed, groupings of features can be made to form a field-effect transistor. The transistor is the basic element from which the computer circuitry is constructed, and the etching process determines the electrical nature of the circuit. The channels and semiconducting layers are equivalent to all of the tubes and wires found in larger devices.

The complexity of the electronics is determined by the number of transistors that can be placed on the chip and this is critically influenced by the size of the features that can be etched. Presently available computers at my university have feature sizes of 0.5 microns (a micron, is one-millionth of a meter.) This compares with sizes of 2.5 microns in 1980, 25 microns in 1960, and 100 microns for the human

hair. Since the device can be two or three centimeters in size (if too large then the opportunity for defects becomes excessive) it is now possible to have several million components on a single chip.

As the features get smaller and closer together, several good things happen for the circuit designer. The speed of switching goes up, and the energy required to achieve a given switching speed goes down. This is reflected in clock rates often quoted by vendors, and we can roughly track progress in manufacture as speeds go from 25 MHz to 50 MHz and up. If the energy need did not go down then the chip would fail, either because of excessive heat or because of the electric fields that provide the energy would be dense enough to disrupt the semiconductor. These factors help permit the greater number of elements per chip, which is essential for addressing the "down" side of miniaturization: interconnecting discrete elements becomes both an electrical and a mechanical nightmare. Fewer complex chips means fewer discrete connections to contend with.

There are theoretical limits to how small the feature size can become, and these limits suggest that a reduction to one-third of present dimensions is all that may be expected for CMOS. As a practical matter, one-half of the present size may be more realistic. This will surely occur by the end of the century, at which point the most fundamental driving force for improvement will have leveled off. The exponential phase is rather quickly coming to an end.

There are some caveats to be noted. The above estimates are for CMOS only, and do not necessarily apply to other materials that might be used or to other physical effects that might be used to develop circuitry. After all, CMOS was preceded by other technologies (bipolar, etc.) which were supplanted just as they were running out of steam. A leading contender to outstrip silicon-based technology has long been considered to be gallium arsenide. While this material is in use in communications gear and in at least one commercially available computer, it appears that it will increase performance only by a further factor of about two. More speculative schemes based on quantum effects, light, or superconductors seem to be some decades away, but pessimistic predictions in this field have usually returned to haunt the prognosticator.

Architecture

The way that the transistors are put together into computing components--memory, arithmetic units, etc.-- and the way that these larger functional units are assembled into complete systems have a major effect on the performance of computers. At the component level, one should think about not only the speed of the unit, which can be measured by the clock rate, but about what work is accomplished in each clock cycle. As might be imagined, there is a huge variation in this regard among various devices currently in use. Historically, mainframe computers have a more powerful array of instructions than the PC, and, for a given clock rate have required fewer cycles and hence less time to accomplish some meaningful unit of work such as multiplying two numbers or sorting a data file.

The most significant recent development in computer design at the chip level is in the so-called "reduced instruction set computer", or RISC technology. Traditional computers, even microprocessors, often have a set of several hundred basic instructions which they can execute. Somewhat counter-intuitively, the RISC philosophy focuses on providing a much smaller number of the most used instructions. This focus permits these instructions to be extremely well-designed from the point of view of efficiency. But the real "secret" is that each of these key instructions can execute in the same number of cycles (the ultimate objective being one cycle). The pay-off derives from the fact that any complex task being performed by the computer will be limited by the slowest of the instructions that must be invoked. So a few weak instructions-- an almost inevitable consequence of a complex instruction set-- reduces the

overall effectiveness of the calculation unless the programmer makes a very detailed study of the lowest level behavior of the system. This time consuming task is obviated if there simply are not any such bottlenecks to be avoided.

RISC architectures have, for several years, been the basis of all of the powerful workstations that are used for computationally demanding tasks such as scientific calculation, engineering design, or special effects in movies such as *Terminator-2*. In 1994 such chips have reached the mass market with the introduction the Apples "PowerMac" line of computers based on the "PowerPC" chip developed jointly by Apple, IBM, and Motorola. The personal computer will thus continue to improve in performance considerably more than the basic CMOS technology would support as they catch up in architectural features.

The other major trend is to found at the system-level of design, and this is the thrust toward computer systems that contain multiple central processing units. The idea is no more complicated than the notion that "many hands make light work". The traditional computer has one processor that sequentially executes instructions that eventually accomplish some unit of useful work. If the tasks can be spread out over several processors all working concurrently, then the job will be done faster even if the individual processors are no faster. Machines have been built with several hundred processors, and designs exist that are scalable to the range of 65,000 processors. When improvements in chips appear to be limited to a ten-fold increase, the attractiveness of "massively parallel processor" (MPP) architectures is obvious.

However, there are numerous difficulties with this superficial picture, and there is another handy aphorism to explain them: "too many cooks spoil the broth!" Many tasks are intrinsically sequential in nature, such as writing a sentence. It would not be effective to have a team of ten writers and assign each one to write every tenth word. Nor would most paragraphs benefit by assigning each writer every tenth sentence. Cooperative writing is practiced, of course, by dividing the task into larger blocks, each of which has some degree of logical completeness and, consequently, some degree of independence from the other blocks.

While the quality of the content may be improved, very few would claim that the writing task is simplified in a multi-author document or committee report. The optimum case is when the document can be structured in advance, different sections delegated to different people, and finally the whole edited and integrated. The preplanning and post writing tasks represent a degree of overhead that is intrinsic to the "parallelizing" of the writing task. And the situation may be very far from optimum: writers of the different sections may require intensive communication throughout the process to make sure that consistent terminology is used, definitions are introduced appropriately, and that the inevitable ambiguities in the outline are resolved. It is the communications overhead that is the critical issue for the success of most parallel applications, and this is significantly impacted by the initial strategy for partitioning the problem among the processors. A final pitfall that might be noted is that the job is not complete until the slowest of the writers finishes (a problem well understood by the editors of conference proceedings.!) so that the potential gains are lost if the units of work can not be completed in equal times.

While the detailed hardware design in parallel computers can partly mitigate some of these problems, there is little hope from this quarter for general purpose machines. The design that works best for one class of problems is frequently abysmal for other cases because of the inherent characteristics of the problems. It appears that the parallel environment will place greater responsibility on programmers, and the best hope appears to depend on developments in the area of software construction.

Software

The importance of computers derives from the universality of their application, and this derives from their programmability. The subway moves people within narrow constraints. The automobile is much more programmable with respect to routes and times. Still, it cannot be changed by a mouse-click into an airplane or an ocean liner; there is no universal transportation device. Consideration of algorithms, data structures, and programming matters moves us from materials science and computer engineering into the realm of computer science.

It is perhaps not generally known that advances in software have accounted for performance improvements of the same order of magnitude as those produced by hardware. The impact of the choice of algorithm, or the strategy for organizing the work, can be illustrated by some simple examples.

Consider a system of equations, which has the matrix form

$$Ax=b.$$

A straightforward solution would be to find the inverse of A , denoted A^{-1} , in terms of which the solution is

$$x=A^{-1}b.$$

For the same amount of work by the computer as it takes to find A^{-1} it is also possible to factor A into two simpler matrices, denoted L and U , which have the following structure: L is all zeros above the diagonal, and U is all zeros below. Using this factorization, $A=LU$, the original problem is equivalent to the two equations:

$$Ly=b \text{ and}$$

$$Ux=y$$

where y has been introduced as an intermediate construct. This hardly looks like progress; one equation has become two, and the new quantity y has been introduced.

However, assuming that there are n equations, the effort to compute x from A^{-1} is proportional to n^2 . Because of the structure of L and U the effort in the second formulation is proportional to n for each of x and y . If $n=1000$, the effort by factorization is of the order of 2000, while using the inverse makes it of order 1,000,000! This saving is similar to decades of chip development, and is not yet reachable with available MPP machines. It follows from the observation that getting $A=LU$ is better than getting A^{-1} because the ensuing calculation is linear with n rather than quadratic.

In several other important applications it has been found possible to reduce the number of operations required for the solution from n^2 to $n \log n$. While not terribly significant for small n , this difference is quite noticeable if $n=1000$.

One of the major successes of theoretical computer science has been to extract the essence of broad classes of problems and obtain a lower bound on the number of calculations required *regardless of the particular algorithm employed*. Thus if the "best" that is allowed theoretically is $n \log n$, and an algorithm is in hand that achieves this, then there is no need to look further. Conversely, if the best known method is of order n^2 then further study is justified.

This area of investigation, called computational complexity theory, has spurred major improvements in technique, but it has also produced some unpleasant shocks. Some problems (fortunately rather abstruse) are demonstrably insoluble; other problems (unfortunately rather simple-seeming) are solvable but grow exponentially in complexity as a function of some key parameter n . Identification of

intractable problems is an important aspect of understanding the limits of computing capability that will exist even in the face of the stunning successes of hardware and software evolution.

Limits of Functionality

We have noted an exponential growth in computer capability, which is beginning to taper off. Limits are discernable unless MPP comes through or entirely new materials are developed. We have also noted that larger problems may quickly overwhelm capacity, and that only in some cases can this be overcome by clever programming. When "better" is equivalent to "bigger" there may be real limits to the improvement of functionality.

There is another non-linear agency at work when questions of benefits of computing are addressed, namely the psychology of the human mind. One need only look at the State Lottery to see that the perceived value of a ticket (the potential for riches) greatly exceeds its economic value (based on cost versus the probability of pay-off). And one need only visit a discount store to see that many will choose a cheaper alternative, even if there is a disproportionate sacrifice in performance (e.g. durability).

Word processing and spread sheets are often given credit for fueling the microcomputer revolution. Without such highly useful applications, the growth of computer power would be pointless, but these applications did not have their paradigm-shifting impact until the computer technology was adequate to the task. Considerable competence was clearly there by 1980, enough so to change society forever.

But despite this impact these applications have continued to evolve. Memory and disk requirements have increased a hundred-fold, and the processor requirements have increased almost as much. Subjectively, has the benefit increased by a factor of 100? A (very!) unscientific sample suggests the perceived increase in value is more nearly 2 than 10^2 , suggesting a more nearly exponential than linear increase in value. Computer improvement has permitted this increase in complexity to occur with no increase in cost, and indeed the addition of features is probably significantly driven by a spirit of "how much can we get on the new box." Regardless of the numerical accuracy of this appraisal, the conclusion is inescapable that utility does not rise at the same exciting pace that computer capacity does.

There are some evidences of logistic curves in the psychological dimension. For example, a reduction from 10 seconds to 1 second to recalculate a spread sheet is probably significant to most users. Improving from 1 second to 1/10 th of a second is discernable, but not very important. An improvement from 1/10th to 1/100th is not even discernable.

Some interesting insights into the computational cost of benefits are provided by efforts at computer programs that play chess. Apart from the interest provided by complexity of chess as an intellectual activity, there is also a well-established numerical scale associated with chess-playing ability: the U.S. Chess Federation Rating System. This system is based on comparative results in sanctioned tournaments. The central idea is that a player gains points by winning or by drawing a game against a higher ranked opponent, and loses points by losing or drawing against a lower ranked player. By obtaining a high enough rating, a player also receives such designations as expert, master, or grandmaster. One or two computer programs have reached the grandmaster level, and freeware available on the internet can turn a desktop machine into an above-average tournament player.

The chess programs start by examining the available moves and choosing one based on certain rules provided by the program's author. For example, giving checkmate would always be played if possible.

At this point, the program is said to have examined one half-move or one "ply". This level does not produce good chess. It is much better if, for each of its possible moves, it examines the possible responses by its opponent. This two-ply search permits some anticipation of possibilities, but is still not good. The program gets stronger as it looks at more and more plies, avoiding moves that lead to the opponent's advantage, and seeking moves that lead to its own advantage at a subsequent point. This ability to look ahead is characteristic of stronger human players, although not carried out in such a mechanistic way. The number of plies that can be examined is limited by the time each player is allotted for making his moves.

In the average chess position there are something roughly 36 legal moves available. Therefore to exhaustively examine one ply requires 36 move evaluations, to evaluate two plies requires 36 evaluations for each of the first 36 possibilities (i.e. 36^2), and to evaluate n plies requires 36^n evaluations. Put slightly differently, each additional ply increases the computing requirement by a factor of 36. Clever programming has permitted this naive exhaustive approach to be improved upon, and leading programs may prune the number of positions to be evaluated down to about 6 per ply. This means that the factor of 36 provides one move by each side (two plies), and the cost of n plies is reduced to 6^n evaluations.

It has been found that the U.S.C.F. rating of computer chess systems goes up almost linearly with the number of plies examined. Thus we have a linear increase in benefit demanding an exponential increase in computational effort.

The leading system (Deep Thought) has achieved grandmaster level by looking ahead 10 plies. By the foregoing estimate this would require examining some 60 million positions per move, or about 300,000 positions per second to comply with tournament time limits. To look ahead two additional full moves (4 plies) requires the speed to increase by more than 1000 times. Since the chess algorithms can exploit parallelism to some extent, this will likely be achieved and would be close to world championship level play.

The places that achievable computer power will influence most us are probably in advanced interfaces and in advanced information management techniques. Today we have achieved capabilities to store large amounts of data, to use sound and graphics and imaging techniques to display and analyze it, and to use the internet obtain additional material from remote sources. Such uses of the computer have created a class of *digital media* that combine and transcend print, film, audio, and video techniques. They are characterized by mixing together the traditional media, and by an interactive capability to navigate through the contents. These methods are becoming available in educational settings, and on home computers thanks to multimedia productions on CD-ROMs.

Experiments at the University of Missouri and elsewhere have revealed substantial capabilities within the typical desktop or laptop machines of today. The limitations which have been encountered can be quantified, and both device improvements and parallel methods have been identified to move us forward into "virtual reality" mediated learning settings. These opportunities fall comfortably within our technological reach, even as the curve is bending over. As has been true throughout the computer age, the greater challenge is the harnessing of these developments in the service of curricular innovation and quality of life.