Frontiers of Supercomputing

by B. L. Buzbee, N. Metropolis, and D. H. Sharp

For many years scientific computing has been a major factor in maintaining U.S. leadership in many areas of science and technology and in the application of science to defense needs. Electronic computers were developed, in fact, during and after World War 11 to meet the need for numerical simulation in the design of nuclear weapons, aircraft, and conventional ordnance. Today. the availability of supercomputers ten thousand times faster than the first electronic devices is having a profound impact on all branches of science and engineering—from astrophysics to elementary particle physics. from fusion energy research to automobile design. The reason is clear: supercomputers extend enormously the range of problems that are effectively solvable.

Although the last forty years have seen a dramatic increase in computer performance, the number of users and the range of applications have been increasing at an even faster rate, to the point that demands for greater performance now far outstrip the improvements in hardware. Moreover, the broadened community of users is also demanding improvements in software that will permit a more comfortable interface between user and machine.

Scientific supercomputing is now at a critical juncture. While the demand for higher speed grows daily. computers as we have known them for the past twenty to thirty years are about as fast as we can make them. The growing demand can only be met by a radical change in computer architecture. a change from a single serial processor whose logical design goes back to Turing and von Neumann to an aggregation of up to a thousand parallel processors that can perform many independent operations concurrently. This radical change in hardware will necessitate improvements in programming languages and software. If made, they could significantly reduce the time needed to translate difficult problems into machine-computable form. What now takes a team of scientists two years to do may be reduced to two months of effort. When this happens. practice in many fields of science and technology will be revolutionized.

These radical changes would also have a large and rapid impact on the nation's economy and security. The skill and effectiveness with which supercomputers can be used to design new and more economical civilian aircraft will determine whether there is em.. ploymnent in Seattle or in a foreign city. Computer-aided design of automobiles is already playing an important role in Detroit's effort to recapture its position in the automobile market. The speed and accuracy with which information can be processed will bear importantly on the effectiveness of our national intelligence activities.

Japan and other foreign countries have already realized the significance of making this quantum jump in supercomputer development, Japan is, in fact, actively engaged in a highly integrated effort to realize it. At this juncture the United States is in danger of losing its long-held leadership in supercomputing.

Various efforts are being made in this country to develop the hardware and software necessary for the change to massively parallel computer architecture. But these efforts are only loosely coordinated.

How can the United States maintain its worldwide supremacy in supercomputing? What policies and strategies are needed to make the revolutionary step to massively parallel supercomputers? How can individual efforts in computer architecture. software. and languages be best coordinated for this purpose?

To discuss these questions. the Laboratory and the National Security Agency spore sored a conference in Los Alamos on August 15-19, 1983. Entitled "Frontiers of Supercomputing." the conference brought together leading representatives from industry, government, and universities who shared the most recent technical developments as well as their individual perspectives on the tactics needed for rapid progress.

Before presenting highlights of the conference we will review in more depth the importance of supercomputing and the trends in computer performance that form the background for the conference discussions.

The Importance of Supercomputers

The term "supercomputer" refers to the most powerful scientific computer available at a given time. The power of a computer is measured by its speed, storage capacity (memory). and precision. Today's commercially available supercomputers, the Cray I from Cray Research and the CYBER 205 from Control Data Corporation, have a peak speed of over one hundred million operations per second, a memory of about four million 64-bit "words," and a precision of at least 64 bits.

There are presently about seventy-five of these supercomputers in use throughout the world. They are being used at national laboratories to solve complex scientific problems in weapon design, energy research, meteorology, oceanography, and geophysics. In industry they are being used for design and simulation of very large-scale integrated circuits, design of aircraft and automobiles, and exploration for oil and minerals. To demonstrate why we need even greater speed, we will examine some of these uses in more detail.

The first step in scientific research and engineering design is to model the phenomena being studied. In most cases the phenomena are complex and are modeled by equations that are nonlinear and singular and. hence. refractory to analysis. Nevertheless. one must somehow discern what the model predicts. test whether the predictions are correct, and then (invariably) improve the model, Each of these steps is difficult, time-consuming, and expensive. Supercomputers play an important role in each step. helping to make practical what would other-

wise be impractical. The demand for improved performance of supercomputers stems from our constant desire to bring new problems over the threshold of practicality.

As an example of the need for supercomputers in modeling complex phenomena, consider magnetic fusion research. Magnetic fusion requires heating and compressing a magnetically confined plasma to the extremes of temperature and density at which thermonuclear fusion will occur. During this process unstable motions of the plasma may occur that make it impossible to attain the required final conditions. In addition, state variables in the plasma may change by many orders of magnitude. Analysis of this process is possible only by means of large-scale numerical computations, and scientists working on the problem report the need for computers one hundred times faster and with larger memories than those now available.

Another example concerns computing the equation of state of a classical one-component plasma. A one-component plasma is an idealized system of one ionic species immersed in a uniform sea of electrons such that the whole system is electrically neutral. If the equation of state of this "simple" material could be computed, it would provide a framework from which to study the equations of state of more complicated substances. Before the advent of the Cray-1, the equation of state of a one-component plasma could not be computed throughout the parameter range of interest.

Using the Cray-1, scientists developed a better, more complex approximation to the interparticle potential. Improved software and very efficient algorithms made it possible for the Cray-1 to execute the new calculation at about 90 million operations per second. Even so, it took some seven hours; but, for the first time, the equation of state for a one-component plasma was calculated throughout the interesting range.

There are numerous ways in which supercomputers play a crucial role in testing models. Models of the circulation of the oceans or the atmosphere or the motion of tectonic plates cannot be tested in the laboratory, but can be simulated on a large computer. Many phenomena are difficult to investigate experimentally in a way that will not modify the behavior being studied. An example is the flow of reactants and products within an internal combustion engine. Supercomputers are currently being used to study this problem.

Testing a nuclear weapon at the Nevada test site costs several million dollars. Running a modern wind tunnel to test airfoil designs costs 150 million dollars a year. Supercomputers can explore a much larger range of designs than can actually be tested, and expensive test facilities can be reserved for the most promising designs.

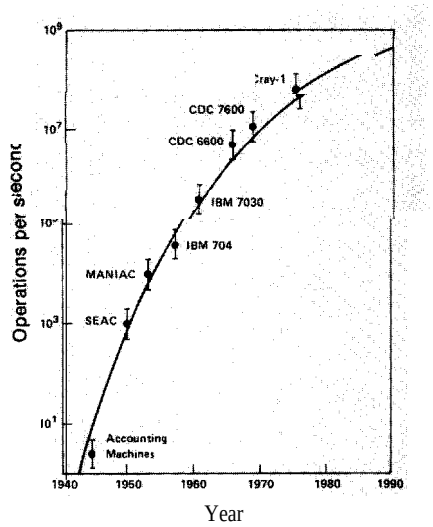
Supercomputers are needed to interpret test results. For example, in nuclear weapons tests many of the crucial physical parameters are not accessible to direct observation, and the measured signals are only indirectly related to the underlying physical processes. D. Henderson of Los Alamos characterized the situation well: "The crucial linkage between these signals, the physical processes, and the device design parameters is available only through simulation."

Finally, supercomputers permit scientists and engineers to circumvent some real-world constraints. In some cases environmental considerations impose constraints. Clearly, the safety of nuclear reactors is not well suited to experimental study. But with powerful computers and reliable models, scientists can simulate reactor accidents, either minor or catastrophic, without endangering the environment.

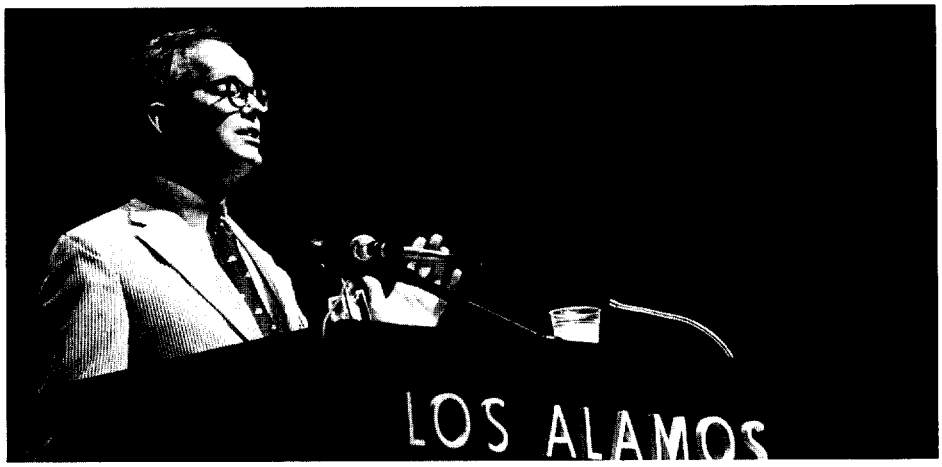
Time can also impose severe constraints on the scientist. Consider, for example, chemical reactions that take place within microseconds. One of the advantages of large-scale numerical simulation is that the scientist using it can "slow the clock" and with graphic display, observe the associated phenomena in slow motion. At the other end of the spectrum are processes that take years or decades to complete. Here numerical simulation can provide an accelerated picture that may help, for example, in assessing the long-term effects of increasing the percentage of carbon dioxide in our atmosphere.

Trends in Performance and Architecture

Most scientists engaged in solving the complex problems outlined above feel that an increase of speed of *at least* two orders of magnitude is required to make significant progress. The accompanying figure shows that the increase in speed, while very rapid at the start, now appears to be leveling off.



The rapid growth has been due primarily to advances in microelectronics. First came the switch from cumbersome and capricious vacuum tubes to small and reliable semiconductor transistors. Then in 1958 Jack Kilby invented a method for fabricating many transistors on a single silicon chip a fraction of an inch on a side, the so-called integrated circuit. In the early 1960s computer switching circuits were made of chips each containing about a dozen transistors. This number increased to several thousand (me-



Admiral B. R. Inman in his keynote address stressed that progress in the information industry will be a critical factor in meeting our security and economic needs.

dium-scale integration) in the early 1970s and to several hundred thousand (very large-scale integration, or VLSI) in the early 1980s, Furthermore. since 1960 the cost of transistor circuits has decreased by a factor of about 10,000.

The increased circuit density and decreased cost has had two major impacts on computer power. First. it became possible to build very large. very fast memories at a tolerable cost, Large memories are essential for complex problems and for problems involving a large data base. Second, increased circuit density reduced the time needed for each cycle of logical operations in the computer.

Until recently a major limiting factor on computer cycle time has been the gate. or switch, delays. For vacuum tubes these delays are 10^{-5} second, for single transistors 10^{-7} second. and for integrated circuits 10^{-9} second. With gate delays reduced to less than a nanosecond. cycle times are now limited by the time required for signals to propagate from one part of the machine to another. The cycle times of today's supercomputers. which contain VLSI components, are between 10 and 20 nanoseconds and are roughly proportional to the linear dimensions of the computer. that is, to the length of the longest wire in the machine.

The figure summarizes the history of computer operation, and the data have been extrapolated into the future by approximation with a modified Gompertz curve. The asymptote to that curve, which represents an upper limit on the speed of a single-processor machine, is about three billion operations per second. Is this an accurate forecast in view of forthcoming developments in integrated circuit technology? The technology with the greatest potential for high speed is Josephson-junction technology>. However, it is estimated that a supercomputer built with that technology would have a speed of *at most* one billion operations per second. which is greater than the speed of the Cray-1 or the CYBER 205 by only a factor of ten.

Thus, supercomputers appear to be close to the performance maximum based on our experience with single-processor machines. If we are to achieve an increase m speed of two orders of magnitude or more. we must look to machines with multiple processors arranged in parallel architectures. that is, to machines that perform many operations concurrently.

Three types of parallel architecture hold promise of providing the needed hundredfold increase in performance:

lockstep vector processors.
tightly coupled parallel processors. and
massively parallel devices.

The first type may be the least promising. It has been shown that achieving maximum performance from a vector processor requires vectorizing at least 90 percent of the operations involved, but a decade of experience with vector processors has revealed that achieving this level of vectorization is difficult,

The second type of architecture employs tightly coupled systems of a few high-performance processors. In principle, collaboration of these processors on a common task can produce the desired hundredfold increase in speed. But important architectural issues. such as the communication geometry between processors and memories, remain unresolved. Analysis of the increase in speed possible from parallel processing shows that the research challenge is to find algorithms, languages, and architecture that. when used as a system, allow a large percentage of work to be processed in parallel with only a minimum number of additional instructions. (See "The Efficiency of Parallel Processing.") However, formulating algorithms for this second type is somewhat easier than for lockstep vetor processors.

Recent work on tightly coupled parallel processors has concentrated on systems with two to four vector processors sharing a large memory. Such machines have been used successfully for parallel processing of scientific computation. Logically, the next steps are systems with eight, sixteen, even sixty-four processors: however, scientists may not be able to find sufficient concurrent tasks to achieve high parallelization with sixty-four processors. The problem lies in the granularity of' the task, that is, the size of the pieces into which the problem can be broken. To achieve high performance on a given processor, granularity should be large. However, to provide a sufficient number of concurrent tasks to keep a large number of processors busy, granularity will have to decrease, and high performance may be lost.

The situation is even more challenging when we consider a massively parallel system with thousands of processors communicating with thousands of memories. In general. scientists cannot find and manage parallelism for such large numbers of processors. Rather. the software must find it, map it onto the architecture, and manage it. Therein lies a formidable research issue. In fact. the issues of concurrency are so great that some scientists are suggesting that we will have to forego the familiar ordering of computation intrinsic in sequential processing. This has revolutionary implications for algorithms.

In summary, the architecture of supercomputers is likely to undergo fundamental changes within the next few years, and these changes may affect many aspects of large-scale computation. To make this transition successfully. a substantial amount of basic research and development will be needed.

Robert S. Cooper
DARPA

Sidney Fernbach
Control Data Corporation

Jacob T. Schwartz
Courant Institute

Conference Highlights

National and industry Perspectives. in his opening comments. Laboratory Director Donald Kerr characterized the present state of affairs as presenting challenges on two frontiers—the intellectual frontier concerned with scientific and technological progress and a leadership frontier concerned with national policies and strategics aimed at maintaining U.S. leadership in supercomputing.

U.S. Senator Jeff Bingaman (New Mexico) raised several questions that were a focus of attention during the conference. How should our overall national effort in supercomputing be coordinated? What is the proper role of government! Are recent initiatives such as the formation of SRC and MCC and the DARPA project sufficient? (SRC, the Semiconductor Research Corporation, and MCC, the Microelectronics and Computer Technology Corporation. are nonprofit research cooperatives drawn from private industry. DARPA. the Pentagon's Defense Advanced Research Projects Agency. has recently inaugurated a Strategic Computing and Survivability project.) Senator Bingaman also stressed that Congress needs to be made more keenly aware of the importance of supercomputing.

In his keynote address Admiral B. R. Inman (U.S. Navy, retired, and now president of MCC) outlined challenges and opportunities facing the United States in coming decades. In the military sphere the USSR will continue to pose a formidable challenge, especially in view of their increasingly effective and mobile conventional forces. (This point was also stressed in the remarks of DARPA director R. S. Cooper and Undersecretary of Defense R. DeLauer.) Robust economic properity, however, offers the opportunity for progress toward world stability. Admiral Inman stressed that a critical factor in meeting our security and economic needs will be progress in the information industry information processing, supercomputing, automation, and robotics. He cited numerous actions that arc required to put the United States in a better position to seize and exploit opportunities as they arise. These include greater investments at universities for graduate training in science and mathematics, new organizations for pooling scarce talent and resources, revised governmental procedures (such as three year authorization bills) to facilitate commitments to longer range research projects, new antitrust legislation, in part to enable the pooling of talent, and a consensus on national security policy.

Both Senator Bingaman and Admiral Inman stressed that the key to rapid progress in the supercomputing field lies in effective collaboration among the academic, industrial, and governmental sectors. Several speakers addressed the question of how the different components of' this triad could best support and reinforce each other's activities.

The viewpoint of the supercomputer manufacturers was presented by William Norris of' Control Data Corporation and by John Rollwagen of Cray) Research. Both speakers called attention to the thinness of the current supercomputer market and pointed out that a larger market is required to sustain an accelerated research and development program. Mr. Norris called for pool ing of research and revisions in antitrust laws. He took the occasion to announce the formation of a new firm, ETA Systems. Inc.. whose goal is to develop, for delivery by the end of 1986, a computer capable of executing ten billion floating point operations per second.

Mr. Rollwagen called attention to the fact that the market for supercomputers might increase substantially if software were available that made them easier to use. Norris and Rollwagen also suggested that a government program to place supercomputers in universities would not only help the market but would produce other benefits as well.

This point was strongly reinforced from the university standpoint by Nobel laureate Kenneth Wilson. He stressed that the potential market for supercornputers was much greater than was generally supposed and that the key to developing this market was adequate software. Researchers at universities can make a major contribution in this area by, for example, expanding the use of modular programming for large, complex problems. In addition. he emphasized that one of the best ways to create a market for supercomputcrs is to train many people in their use. This can be done if' the universities have supercomputers.

Wilson also pointed out that at present universities are largely isolated from the supercomputer community. For example. while supercomputer manufacturers are planning evolutionary steps in machine design, workers at universities are concentrating on the revolutionary massively parallel architectures and appropriate software and algorithms to make such systems usable. Would not each sector gain from closer interaction! How should this be accomplished?

Several speakers were concerned about the fact that highly reliable VLSI components. which form an essential element of

Richard DeLauer
ndersecretary of Defense

Kenneth G. Wilson
Cornell University

John Rollwagen
Cray Research

James F. Decker
DOE

modern supercomputers, are often difficult to procure from domestic vendors. The result is that more and more of these components are being purchased from Japanese manufacturers. In responding to the suggestion that the U.S. government provide more support to the domestic semiconductor industry, DeLauer pointed out that his agency was already furnishing some help. Moreover, there are numerous domestic industries call ing for aid, and available funds are limited. Hence the semiconductor industry as well as the supercomputer industry must find effective ways to leverage the support that the DOD can give.

There was a consensus at the conference on the need for clearly defined national supercomputer goals and on a strategic plan to reach them. The Federal Coordinating Committee for Science and Engineering Technology has asked the DOE to prepare such a plan. The current status of this plan was reviewed by James Decker of the DOE. The plan provides for the government to accelerate its use of supercomputers, to acquire and use experimental systems, and to encourage long-range research and development with tax incentives and increased support.

Throughout the conference the Japanese initiatives in supercomputing were much on people's minds. J. Worlton reviewed Japan's activities in the supercomputer field and assessed the strengths and weaknesses of their development strategies. The effective cooperation that they have achieved between government. industry, and academia was viewed as a substantial asset. While no one suggested that Japanese organizational methods could provide a detailed role model for the United States, a concerted effort to achieve comparable cooperation among the various sectors of the U.S. supercomputing community is certainly in order.

Scientific Developments in Architecture. Software. Algorithms, and Applications. The scientific talks presented at he conference were organized into session covering advanced architectures, supercomputing at Los Alamos, software, and algorithms and applacations.

The trend toward parallel computing architectures was abundantly evident in presentations by computer manufacturers and academic researchers. By the end of the decade. commercially supplied systems with eight or more processors will be available. B. Smith of Denelcor presented a new industrial entry. the first attempt, *ab initio,* at modest parallelism.

Most academic projects make extensive use of VLSI, exploiting 32-bit microprocessors and 256K memory chips. Many of them can be classified as "dancehall" machines. Dancehall systems have processors aligned along one side. memories on the other, and a communication geometry to bind them together, High performance on these systems necessitates algorithms that keep all of the processors "dancing" all of the time.

James Browne of the University of Texas, Austin, observed that there are three general models of massively parallel systems based on whether decisions concerning the communication. synchronization. and granularity of parallel operations are fixed at design time. compile time, or execution time, These decisions are made at design time in fixed-net architectures. Systolic arrays con taining hardware components designed to carry out specific algorithms are one example of a fixed-net architecture. Such systems provide fast execution of the algorithms for which they are designed, but other algorithms may have to be adapted to fit the architecture.

In bind-at compile time architectures the topology of the interconnection of processors and memory can be reconfigured to suit a particular algorithm but remains in that configuration until explicitly reconfigured. The TRAC machine at the University of Texas at Austin is an example of bind at-compile time architecture. Reconfigurable architectures must have more complex control systems than those that bind at design time, and the reconfiguration process does take time. The trade-off is that reconfigurable architectures can be used for many different algorithms.

Bindat-execution-tirne architectures do not explicitly specify topology but, through use of control structures and shared memory. allow any topology. These systems are the most flexible since no algorithm is constrained by an inappropriate architecture. The price is the overhead required to synchronize and communicate information, Dataflow machines are an example of bind-a-execution-time systems.

Today, effective use of a multiprocessor system (even a vectorizing system) requires intimate knowledge of the hardware, the compiler's mapping of code onto the hardware. and the problem at hand. On systems with one hundred or more processors, software must free the user from these details because their complexity will defy human management. Many issues involving algorithms, models, architectures, and languages for massively parallel systems have yet to be

resolved.

John Armstrong of IBM, in his discussion of VLSI technology, indicated that as components become more integrated, the need for basic research and development in materials science and packaging technology increases. Further. he believes that the current level of research in these areas is inadequate,

Several speakers noted that supercomputer systems require high-speed peripherals (in particular, disks) and that. in general. little work is being done to advance their performance. Resolution of this problem will require collaboration between manufacturers and government laboratories.

The current status of large-scale computation in nuclear weapons design (D. Henderson, Los Alamos). fluid dynamics (J. Glimm, Courant Institute), fusion reactor design (D. Nelson. DOE), aircraft and spacecraft design (W. Ballhaus, NASA-Ames Research Center), and oil reservoir simulation (G. Byrne. Exxon Research) was reviewed. An observation of general applicability was made by Glimm. Problems in all these areas are three-dimensional. time-dependent. nonlinear, and singular. Simple algorithms converge slowly when applied to such problems. Hence, most interesting problems are undercomputed. To achieve an increase in resolution by an order of magnitude in a three-dimensional, time-dependent problem requires an increase in computing power by a factor of 10,000, far beyond projected hardware improvements. Thus, to bridge the gap between needs and projected hardware capabilities, one must look to the development of better algorithms and more powerful software for their implementation,

Several new applications of supercomputing were also discussed, These included design of special-purpose, very large-scale integrated circuits (D. Rose, Bell Laboratories), robotics (J. Schwartz, Courant Institute). CAD/CAM (J. Heiney, Ford Motor Co.) and finally. an unexpected topic, the study of cooperative phenomena in human behavior (F. Harlow. Los Alamos). Also presented was an exhibition of animated computer graphics, an application whose full realization demands far more of computers than today's supercomputers can provide. Quite apart from the dazzle, animated graphics represents a key mode of interaction between the computer and the brain. which if properly exploited could progress far beyond the already impressive development.

Left to right: Admiral B. R. Inman, Senator Jeff Bingaman, and Laboratory Director Donald Kerr at the opening of the conference.

K. H. Speierman of the National Security Agency delivering the conference summary.

Conference Summary

A broad spectrum of interests and points of view, was expressed during the week. The participants concurred on many general points, and these were well summarized by K. Speierman of the National Security Agency'. Among them were the following critical issues,

Systems Approach. All aspects of massively parallel systems—architecture, algorithms, and software-must be developed in concert.

VLSI Supply. Highly reliable VLSI components are often difficult to procure from domestic vendors. Further, associated research and development in materials science and packaging technology are inadequate.

High-Speed Peripherals. High-speed peripherals are essential to supercomputing systems, and very little research and development is being done in the United States in this area.

Future Market. Many of the participants foresee a much larger market for supercomputers due to growing industrial use.

Nearly forty years ago the art of computing made a discontinuous leap from electromechanical plodding to crisp, electronic swiftness. Many aspects of life have been both broadened and rendered sophisticated by the advent of electronic computation. The quantum jump envisioned today is from the ultimate in serial operation to a coordinated parallel mode. ■

The proceedings of the conference are to be published by the University of California Press as a volume in the Los Alamos Series in Basic and Applied Sciences.

The Efficiency of Parallel Processing

by B. L. Buzbee

Parallel processing, or the application of several processors to a single task, is an old idea with a relatively large literature. The advent of very large-scale integrated technology has made testing the idea feasible, and the fact that single-processor systems are approaching their maximum performance level has made it necessary. We shall show, however, that successful use of parallel processing imposes stringent performance requirements on algorithms, software, and architecture.

The so-called asynchronous systems that use a few tightly coupled high-speed processors are a natural evolution from high-speed single-processor systems, Indeed, systems with two to four processors will soon be available (for example, the Cray X-MP, the Cray-2, and the Control Data System 2XX). Systems with eight to sixteen processors are likely by the early 1990s. What are the prospects of using the parallelism in such systems to achieve high speed in the execution of a single application? Early attempts with vector processing have shown that plunging forward without a precise understanding of the factors involved can lead to disastrous results. Such understanding will be even more critical for systems now contemplated that may use up to a thousand processors.

The key issue in the parallel processing of a single application is the speedup achieved, especially its dependence on the number of processors used. We define speedup *(S)* as the factor by which the execution time for the application changes: that is,

$$S = \frac{\text{execution time for one processor}}{\text{execution time for } p \text{ processors}}$$

To estimate the speedup of a tightly coupled system on a single application, we use a model of parallel computation introduced by Ware. We define a as the fraction of work in the application that can be processed in parallel. Then we make a simplifying assumption of a two-state machine; that is, at any instant either all p processors are operating or only one processor is operating. If we normalize the execution time for one processor to unity, then

$$S\ (p,a) = \frac{1}{(1\text{-}a) + a/p}$$

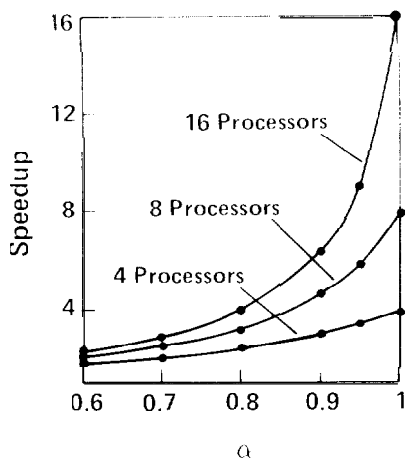
Note that the first term in the denominator is the execution time devoted to that part of the application that *cannot* be processed in parallel, and the second term is the time for that part that can be processed in parallel. How does speedup vary with *a?* In particular. what is this relationship for $a = 1$. the ideal limit of complete parallelization? Differentiating S, we find that

$$\frac{\partial S(p,\alpha)}{\partial \alpha}\bigg|_{\alpha=1} = p^2 - p \ .$$

The accompanying figure shows the Ware model of speedup as a function of a for a 4-processor, an 8-processor, and a 16-processor system. The quadratic dependence of the derivative on p results in low speedup for a less than 0.9. Consequently, to achieve *significant* speedup, we must have highly parallel algorithms. It is by no means evident that algorithms in current use on single-processor machines contain the requisite parallelism. and research will be required to find suitable replacements for those that do not. Further, the highly parallel algorithms available must be implemented with care. For example, it is not sufficient to look at just those portions of the application amenable to parallelism because a is determined by the entire application. For a close to 1. changes in those few portions less amenable to parallelism will cause small changes in a, but the quadratic behavior of the derivative will translate those small changes in a into large changes in speedup.

Those who have experience with vector processors will note a striking similarity between the Ware curves and plots of vector processor performance versus the fraction of vectorizable computation. This similarity is due to the assumption in the Ware model of a two-state machine since a vector processor can also be viewed in that manner. In one state it is a relatively slow, general-purpose machine. and in the other state it is capable of high performance on vector operations,

Ware's model is inadequate in that it assumes that the instruction stream executed on a parallel system is the same as that executed on a single processor. Seldom is this the case because multiple-processor systems usually require execution of instructions dealing with synchronization of the processes and communication between



Speedup as a function of parallelism (a) and number of processors.

processors. Further, parallel algorithms may inherently require additional instructions. To correct for this inadequacy, we add a term, $\sigma(p)$, to the execution time for parallel implementation that is at best nonnegative and usually monotonically increasing with p. Actually, σ is a function not only of p but of the algorithm. the architecture, and even of α. Let $S(p,\alpha,\sigma)$ denote speedup for this modified model. Then

$$S(p,\alpha,\sigma) = \frac{1}{(1-\alpha) + \alpha/p + \sigma(p)} \ .$$

If the application can be put completely in parallel form, then

$$S(p,\alpha,\sigma)\big|_{\alpha=1} = \frac{p}{1 + p\sigma(p)} \ .$$

In other words, the maximum speedup of a real system is less than the number of processors p, and it may be significantly less. Also note that, whatever the value of a, S will have a maximum for sufficiently large p because α/p becomes insignificant while $\sigma(p)$ continues to increase,

Thus the research challenge in parallel processing involves finding algorithms, programming languages, and parallel architectures that, when used as a system, yield a large amount of work processed in parallel (large *a)* at the expense of a minimum number of additional instructions (small σ). ■

The HEP Parallel Processor

by James W. Moore

Although there is an abundance of concepts for parallel computing, there is a dearth of experimental data delineating their strengths and weaknesses. Consequently, for the past three years personnel in the Laboratory's Computing Division have been conducting experiments on a few parallel computing systems. The data thus far are uniformly positive in supporting the idea that parallel processing may yield substantial increments in computing power. However, the amount of data that we have been able to collect is small because the experiments had to be conducted at sites away from the Laboratory, often in "software-poor" environments.

We recently leased a Heterogeneous Element Processor (HEP) manufactured by Denelcor, Inc. of Denver, Colorado. This machine (first developed for the Army Ballistic Research Laboratories at Aberdeen) is a parallel processor suitable for general-purpose applications. We and others throughout the country will use the HEP to explore and evaluate parallel-processing techniques for applications representative of future supercomputing requirements. Only the beginning steps have been taken, and many difficulties remain to be resolved as we move from experiments that use one or two of this machine's processors to those that use many. But what are the principles of the HEP?

Parallel processing can be used separately or concurrently on two types of information: instructions and data. Much of the early parallel processing concentrated on multiple-data streams. However, computer systems such as the HEP can handle both multiple-instruction streams and multiple-data streams. These are called MIMD machines.

The HEP achieves MIMD with a system of hardware and software that is one of the most innovative architectures since the advent of electronic computing. In addition, it is remarkably easy to use with the FORTRAN language. In its maximum configuration it will be capable of executing up to 160 million instructions per second.

The Architecture

Figure 1 indicates the general architecture of the HEP. The machine consists of a number of process execution modules (PEMs), each with its own data memory bank. connected to the HEP switch. In addition, there are other processors connected to the switch, such as the operating system processor and the disk processor. Each PEM can access its own data memory bank directly, but access to most of the memory is through the switch.

In a MIMD architecture, entire programs or, more likely, pieces of programs, called *processes*, execute in parallel, that is, concurrently. Although each process has its own independent instruction stream operating on its own data stream, processes cooperate by sharing data and solving parts of the same problem in parallel, Thus. throughput can be increased by a factor of N, where N is the average number of operations executed concurrently.

The HEP implements MIMD with up to sixteen PEMs, each PEM capable of executing up to sixty-four processes concurrently. It should be noted, however, that these upper limits may not be the most efficient configuration for a given, or even for most, applications. Any number of PEMs can cooperate on a job, or each PEM may be running several unrelated jobs. All of the instruction streams and their associated data streams are held in main memory while the associated processes are active.

How is parallel processing handled within an individual PEM? This is done by "pipelining" instructions so that several are in different phases of execution at any one moment. A process is selected for execution each machine cycle, a single instruction for that process is started, and the process is made unavailable for further execution until that instruction is complete. Because most instructions require eight cycles to complete, at least eight processes must be executed concurrently in order to use a PEM fully. However, memory access instructions require substantially more than eight cycles. Thus, in practice. about twelve concurrent processes are needed for full utilization, and a single HEP PEM can be considered a "virtual" 8- to 12-processor machine. If a given application is formulated and executed using p processes (where p is an integer from 1 to 12), then execution time for the application will be inversely proportional top.

Now what happens when individual PEMs are linked together? Each PEM has its own program memory to prevent conflicts in accessing instructions, and all PEMs are connected to the large number of other data memory banks through the HEP switch (a high-speed, packet-switched network). One result of these connections is that the number of switch nodes increases more rapidly than the number of PEMs. As one changes from a 1-PEM system toward the maximum 16-PEM configuration, the transmittal time through the HEP switch, called latency,

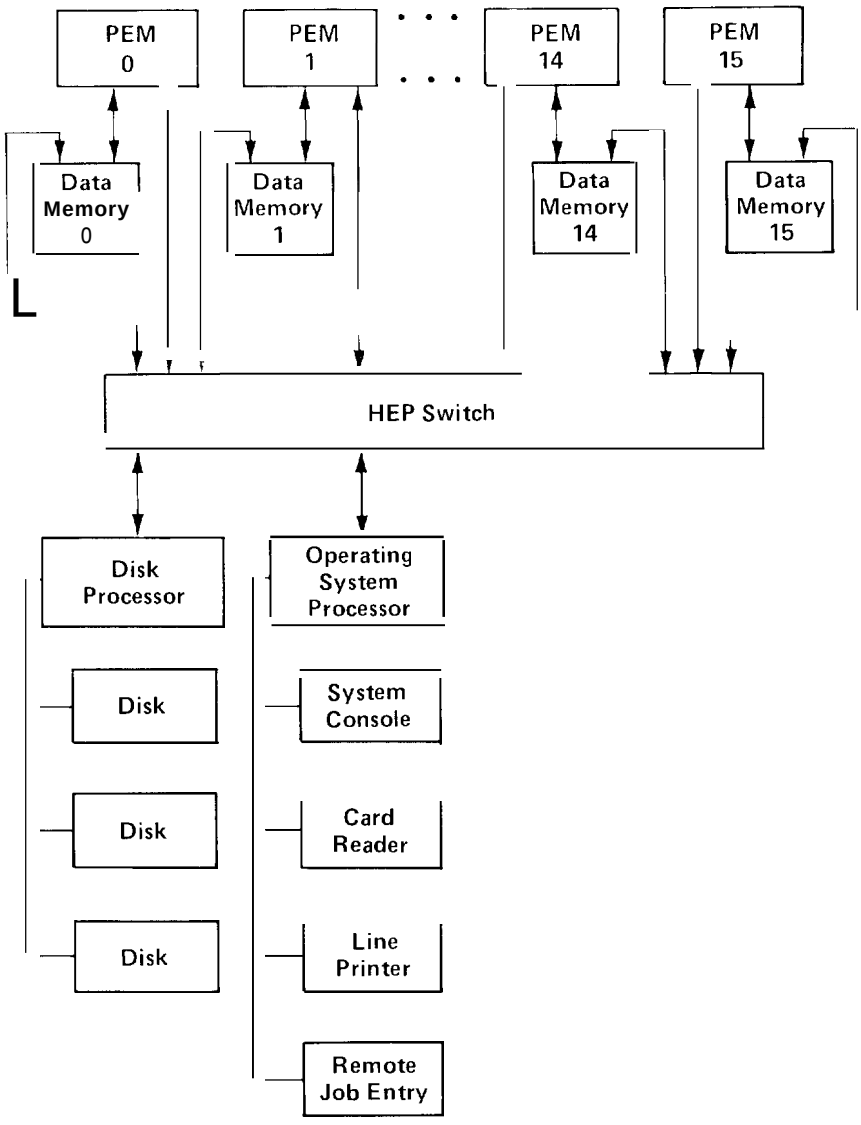
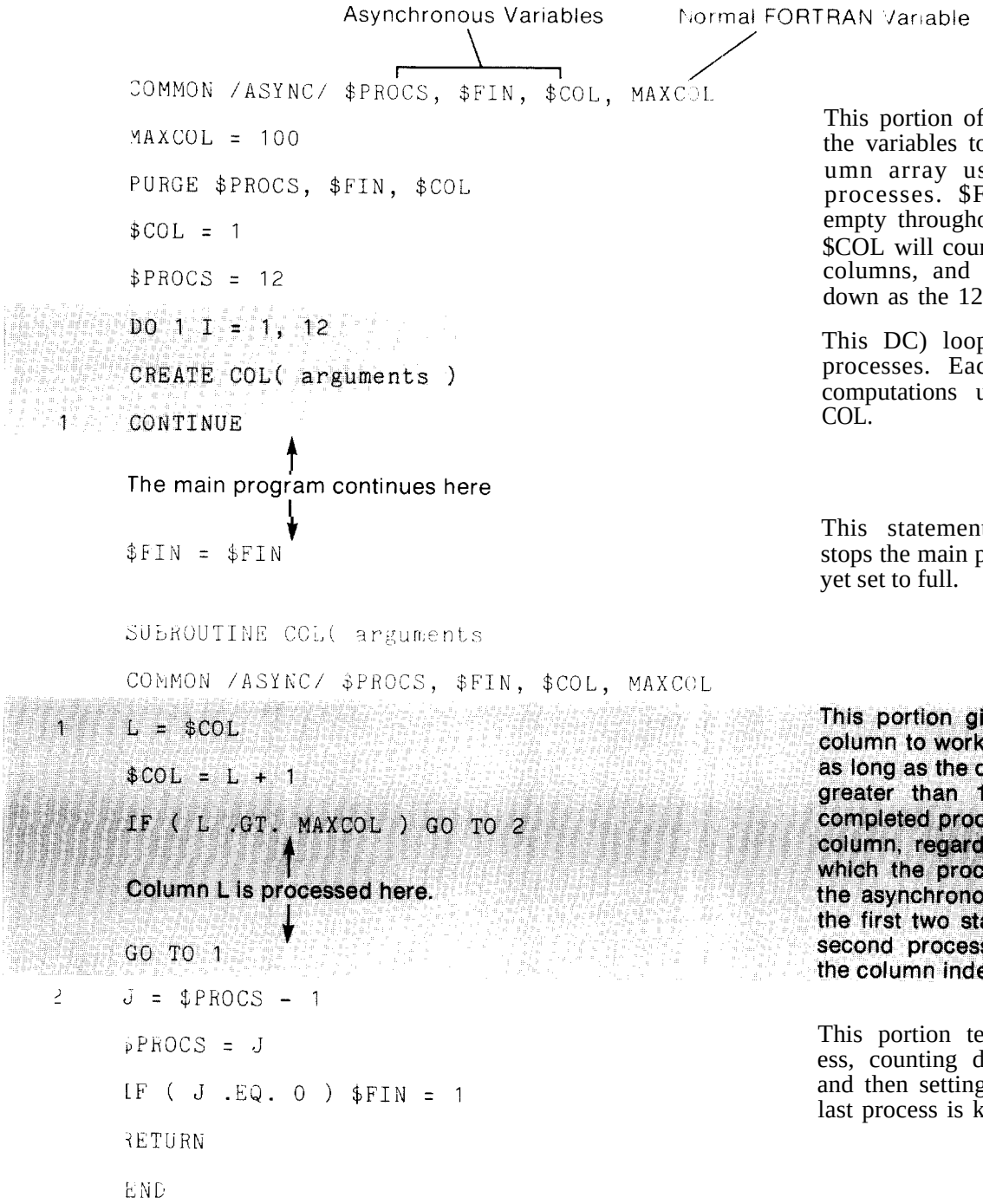


Fig. 1. The HEP parallel processor.



This portion of the program sets up the variables to process an 100-column array using 12 concurrent processes. $FIN remains set at empty throughout the computations, $COL will count up through the 100 columns, and $PROCS will count down as the 12 processes die off.

This DC) loop CREATES the 12 processes. Each process does its computations using SUBROUTINE COL.

This statement, otherwise trivial, stops the main program if $FIN is not yet set to full.

This portion gives each process a column to work on in the array and, as long as the column index L is not greater than 100, will send each completed process back for another column, regardless of the order in which the processes finish. Use of the asynchronous variable $COL in the first two statements prevents a second process from starting until the column index has been reset.

This portion terminates each process, counting down with $PROCS, and then setting $FIN to full as the last process is killed.

Fig. 2. An example of HEP FORTRAN,

quickly becomes substantial. Such latency increases the number of processes that must be running in each PEM to achieve full utilization. Although there is not enough data yet to provide good estimates on how fast latency actually increases with the number of PEMs, experience with a 2-PEM system suggests that a 4-PEM system will require about twenty concurrent processes in each PEM.

Process Synchronization

A critical issue in MIMD machines is the synchronization of processes. The HEP solves this problem in a simple and elegant manner, Each 64-bit data word has an extra bit that is set to full each time a datum is stored and is cleared to empty each time a datum is fetched, In addition, two sets of memory instructions are employed. One set is used normally throughout most of the program. This set ignores the extra bit and will fetch or store a data word regardless of whether the bit is full or empty. Data may then be used as often as required in a process.

The second set of instructions is defined through the use of asynchronous variables, Typically, this set is used only at the start or finish of a process, acting as a barrier against interference from other processes, The set will not fetch from an empty word or store into a full word. Thus, to synchronize several processes an asynchronous variable is defined that can only be accessed, using the second set of instructions, at the appropriate time by each process. A process that needs to fetch an asynchronous variable will not do so if the extra bit is empty and will not proceed until another process stores into the variable, setting it full. Because the full and empty properties of this extra bit are implemented in the HEP hardware at the user level. requiring no operating system intervention. the usual synchronization methods (semaphores, etc.) can be used, and process synchronization is very efficient.

FORTRAN Extensions to Support Parallelism

Only two extensions to standard FORTRAN arc required to exploit the parallelism inherent in the HEP: process creation and asynchronous variables. Standard FORTRAN can. in fact, handle both, but the current HEP FORTRAN has extensions specifically tailored to do so.

These extensions allow the programmer to create processes in parallel as needed and then let them disappear once they are no longer needed. Also the number of PEMs being used will vary with the number of processes that are created at any given moment.

Process creation syntax is almost identical to that for calling subroutines: CALL is replaced by CREATE, However, in a normal program, once a subroutine is CALLed, the main program stops; in HEP FORTRAN. the main program may continue while a CREATEd process is being worked on. If the main program has no other work, it may CALL a subroutine and put itself on equal footing with the other processes until it exits the subroutine. A process is eliminated when it reaches the normal RETURN statement.

We can illustrate these techniques by showing how the HEP is used to process an array in which each column needs to be processed in the same manner but independently of the other columns (Fig. 2). First, one defines a subroutine that can process a single column in a sequential fashion. We could usc this subroutine by creating a process for each Column and then scheduling all the processes in parallel, but there is a limit on the number of processes that each PEM can handle. A better technique would be to CREATE eight to twelve processes per PEM and let the processes *self-schedule.* Each process selects a column from the array, does the computation for that column, then looks for additional columns to work on. Several asynchronous variables are the key to this technique. Each process that is not computing checks the first of' these variables both to see if it can start a computation and, if so, which column is next in line. At the end of that computation and regardless of what stage any other process has reached. the process checks again to see if there are further columns to be dealt with. If not, the process is terminated. A second asynchronous variable counts down as the processes die off. When the last operating process completes its computation. a number is stored in a third, previously empty asynchronous variable, setting its extra bit to full. This altered variable is a signal to the main program that it may use the recently generated data. This method tends to smooth irregularities in process execution time arising from disparities in the amount of processing done on the individual columns and, further, does not require changes if the dimension of the array is changed.

More elegant syntactic constructs can be devised. but the HEP extensions are workable. For well-structured code, conversion to the HEP is quite easy, For more complex programs the main difficulty is verifying that the parallel processes defined are in fact independent. No tools currently exist to help with this verification.

Early Experience with the HEP

Several relatively small FORTRAN codes have been used on the HEP at Los Alamos. One such code is SIMPLE, a 2000-line, two-dimensional, Lagrangian, hydro-diffusion code. By partitioning this code into processes, about 99 per cent of it can be executed in parallel. The speedup on a 1-PEM system is close to linear for up to eleven processes and then flattens out, indicating that the PEM is fully utilized, To achieve this degree of speedup on any MIMD machine requires a very high percentage of parallel code. How difficult it will be to achieve a high percentage of parallelism on full-scale production codes is an open question, but the potential payoff is significant. ■

U.S. GOVERNMENT PRINTING OFFICE 1983 — 776-101/1007

LA-UR-83-2940