Prospects for Parallel Computing 1992

Geoffrey C. Fox
gcf@npac.syr.edu
http://www.npac.syr.edu

Northeast Parallel Architectures Center
111 College Place
Syracuse University
Syracuse, New York 13244-4100

Technology

The best enterprises have both a compelling need pulling them forward and an innovative technological solution pushing them on. In high-performance computing, we have the need for increased computational power in many applications and the inevitable long-term solution is massive parallelism. In the short term, the relation between pull and push may seem unclear as novel algorithms and software are needed to support parallel computing. However, eventually parallelism will be present in all computers—including those in your children's video game, your personal computer or workstation, and the central supercomputer

The technological driving force is VLSI, or very large scale integration—the same technology that has created the personal computer and workstation market over the last decade. In 1980, the Intel 8086 used 50,000 transistors while in 1992 the latest Digital alpha RISC chip contains 1,680,000 transistors—a factor of 30 increase. The dramatic improvement in chip density comes together with an increase in clock speed and improved design so that the alpha delivers over a factor of 1,000 better performance on scientific problems than the 8086–8087 chip pair of the early 1980's.

The increasing density of transistors on a chip follows directly from a decreasing feature size which is now 0.75μ for the alpha. Feature size will continue to decrease, and by the year 2000, chips with 50,000,000 transistors are expected to be available. What can we do with all these transistors?

With around a million transistors on a chip, designers were able to move full mainframe functionality to about 2 cm^2 of a chip. This enabled the personal computing and workstation revolutions. The next factors of 10

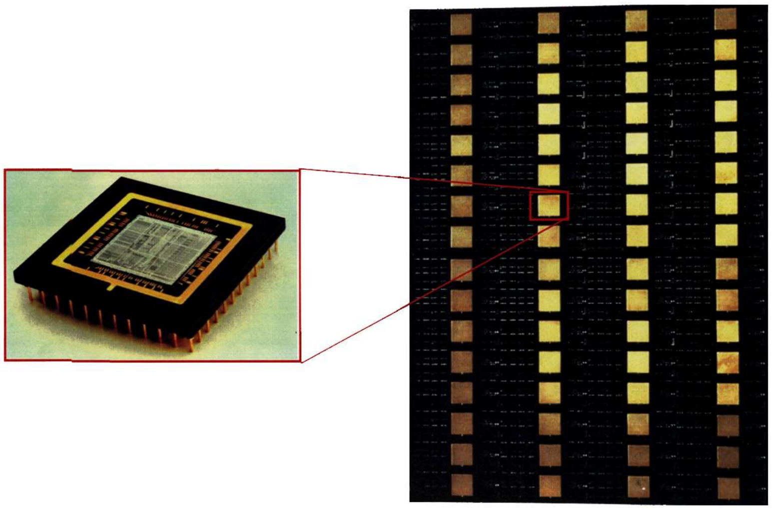

Figure 1: The nCUBE-2 Node and Its Integration into a Board. Up to 128 of these boards can be combined into a single supercomputer.

increase in transistor density must go into some form of parallelism by replicating several CPU's on a single chip.

By the year 2000, parallelism is thus inevitable in all computers. Today, we see it in the larger machines, as we replicate many chips and many printed circuit boards to build systems as arrays of nodes; each unit of which is some variant of the microprocessor. This is illustrated in Figure 1, which shows a nCUBE parallel supercomputer with 64 identical nodes on each board—each node is a single chip CPU with additional memory chips. To be useful, these nodes must be linked in some way, and this is still a matter of much research and experimentation. Further, we can argue as to the most appropriate node to replicate; is it a "small" nodes as in the nCUBE of Figure 1, or is it more powerful "fat" nodes, such as those offered in CM-5 and Intel Touchstone where each node is a sophisticated multichip printed circuit board. However, these detailed issues should not obscure the basic point; parallelism allows one to build the world's fastest and most cost effective supercomputers.

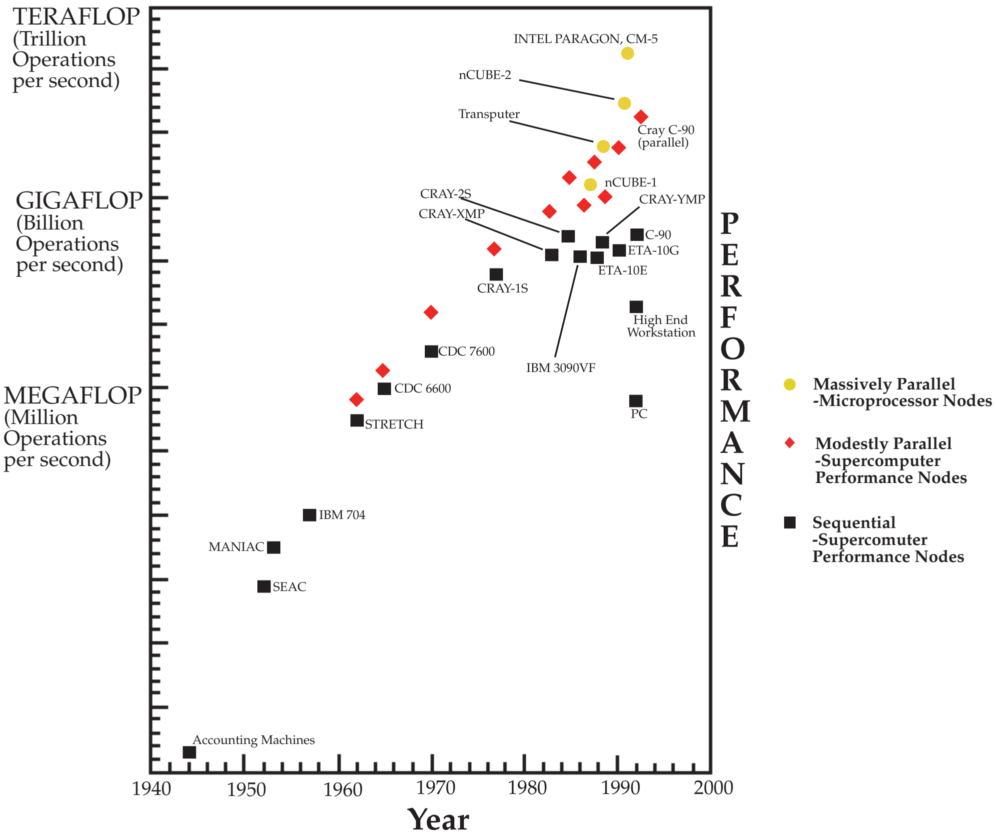

Figure 2: Performance of Parallel and Sequential Supercomputers

Figure 2 illustrates this as a function of time showing, already today, a factor of 10 advantage for parallel versus conventional supercomputers.

Parallelism may only be critical today for supercomputer vendors and users. By the year 2000, all supercomputers will have to address the hardware, algorithmic, and software issues implied by parallelism. The reward will be amazing performance and the opening up of new fields; the price will be a major rethinking and reimplementation of software, algorithms, and applications.

Grand Challenges

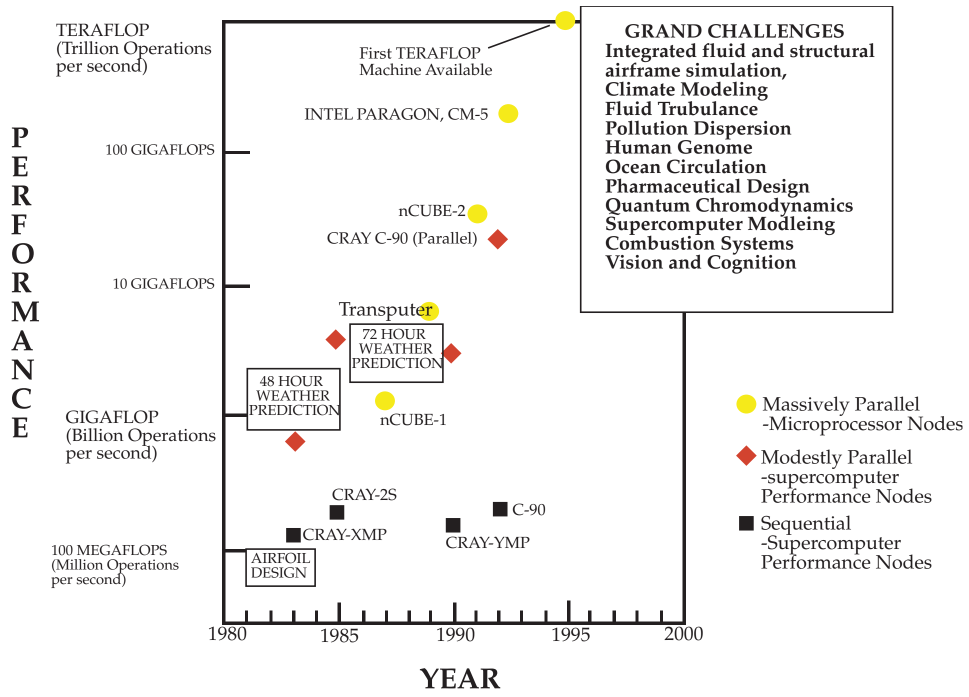

Figure 3: Grand Challenge Applications. Some major applications that will be enabled by parallel supercomputers.

The President has instituted this year, the five-year federal High Performance Computing and Communications Initiative. This will spur the development of the technology described above and is focused on the solution of grand challenges shown in Figure 3. These are fundamental problems in science and engineering, with broad economic and scientific impact, whose solution could be advanced by applying high performance computing techniques and resources.

The activities of several federal agencies have been coordinated in this initiative. DARPA is developing the basic technology, which are applied to the grand challenges by DOE, NASA, NSF, NIH, EPA, and NOAA. Selected activities include the mapping of the human genome in DOE, climate modeling in DOE and NOAA, coupled structural and airflow simulations of advanced powered lift, and a high-speed civil transport by NASA.

Well-Known Parallel Computers

We can learn quite a bit about the use and design of parallel computers by studying parallelism in nature and society. In fact, one can view society or culture as a set of rules and conventions to allow people to work together, i.e., in parallel, effectively, and harmoniously.

A simple illustration is the way we tackle a large project—the construction of the space shuttle. It would be attractive to solve this sequentially by hiring a single superman to complete this project. This is prohibited by current physical phenomenology, and so instead one puts together a team, maybe in this case involving 100,000 "ordinary" people. These people work in parallel to complete the shuttle. A parallel computer is quite similar, we might use 10^5 digital computers working together to simulate airflow over a new shuttle design. Key in NASA's shuttle project is the management structure. This becomes, for the analogy, the issue of computer hardware and software architecture; a key research area in computer science.

We can view the brain as a parallel computer with some 10^{12} neurons working together to solve information processing and decision-making problems. The neurons are analogous to the node shown in Figure 1(a); nature links neurons by axons and dendrites, not wires and printed circuit board traces used by nCUBE. However, the basic design—interconnected elements communicating by message passing—is the same and further both nature's and digital parallel computer use the same mechanism of data parallelism to solve problems concurrently.

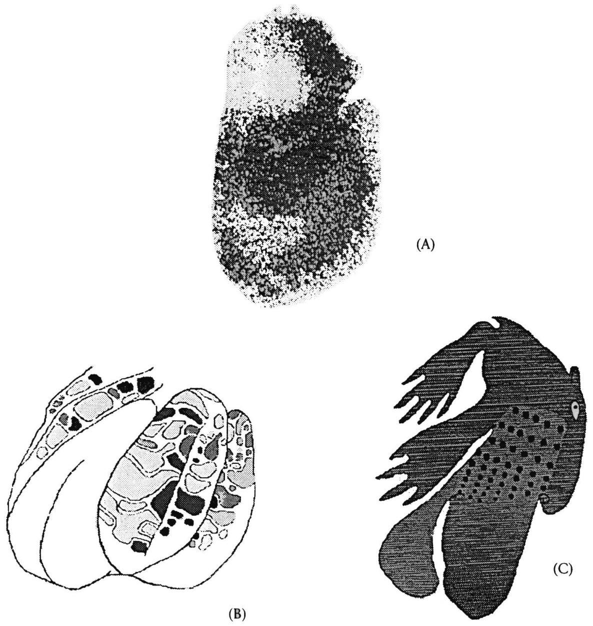

Figure 4: Three Parallel Computing Strategies Found in the Brain (of a Rat). Each figure depicts brain activity corresponding to various functions: (A) continuous map of a tactile inputs in somatosensory cortex, (B) patchy map of tactile inputs to cerebellar cortex, and (C) scattered mapping of olfactory cortex as represented by the unstructured pattern of 2DG update in a single section of this cortex [Nelson:90b].

Data Parallelism

Parallel computing is general purpose because there is a single unifying mechanism on which it is based—this is called domain decomposition or data parallelism. Nature solves complex problems by dividing them up and assigning particular neurons, or group of neurons, to different parts of the problem. This is illustrated in Figure 4, which shows that different areas of the brain are responsible for disentangling tactile information from different parts of the body. Again, vision is a major task for the brain and there is direct spatial mapping of received pixels of light at the retina to neurons in the brain.

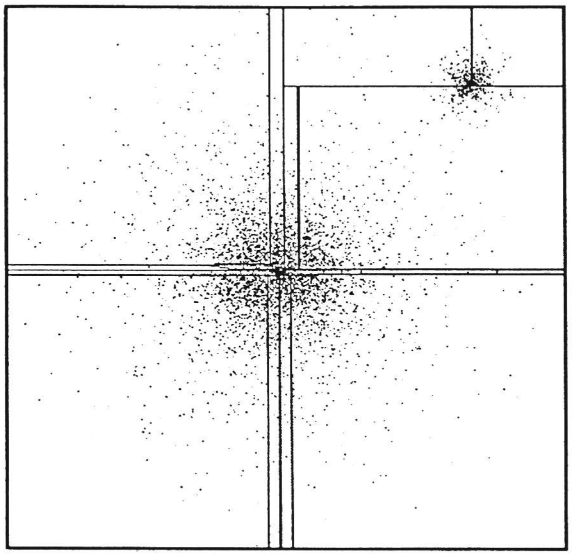

Figure 5: A Two-dimensional Projection of a Model Universe in which Two Galaxies are on a Collision Course. This is a simplified version with 18,000 "stars" of a large simulation reported in [Salmon:89b]. The irregular decomposition onto a 16-node machine is illustrated above.

Parallel simulation of interacting particles, shown in Figure 5, is handled by data parallelism with individual particles being assigned to a particular node in the parallel machine. The astrophysical simulation of Figure 5 is very inhomogeneous and corresponding the spatial regions assigned to a node are irregular and indeed time dependent. This complexity was challenging for the implementation, but the resultant program achieved excellent performance with a speedup of over 800 on a 1024-node nCUBE. Similar success can be anticipated for parallel implementations of molecular dynamics codes, such as CHARMM.

Several chemistry computations involve generation and manipulation of large full matrices that represent the interaction Hamiltonian. Energy level calculations involve eigenvalue determination while scattering can use matrix multiplication and linear equation solution. The same concept of data parallelism is used with, as seen in Figure 6, a simple regular decomposition of the matrix onto the processors. Parallelism is present both for generation of the matrix elements that proceed independently in each node, and the eigenvalue and other matrix operations. A general matrix library LA-

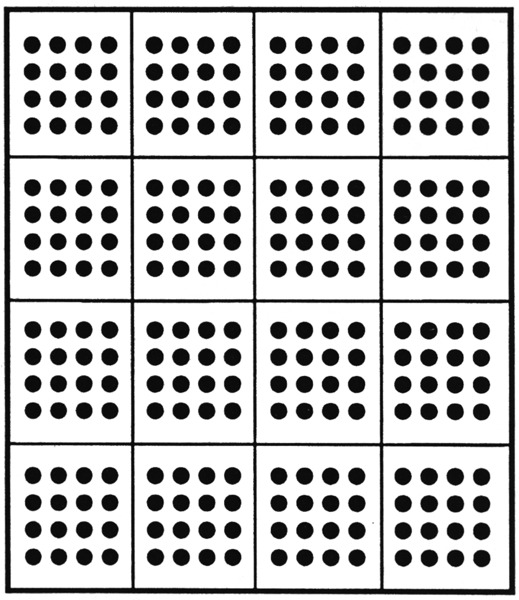

Figure 6: 16×16 Matrix Decomposed onto a 4×4 Parallel Computer Array

PACK will soon be available for a broad class of high-performance vector and parallel computers.

Problems consist of algorithms applied to a large data domain. Data parallelism achieves parallelism by splitting up domain and applying the computational algorithm concurrently to each point.

Current Parallel Machines

The field of parallel computing changes rapidly with, as in the workstation market, vendors leapfrogging each other with new models. Further, any given model is essentially obsolete after some three years, with new machines having very different design and software support. Here, we will discuss some of the machines that are interesting in 1992. There are three broad classes of machines. The first is the so-called SIMD, or synchronous machine, where we have a coupled array of computers with distributed memory and processing units, i.e., each processor unit is associated with its own memory. On SIMD machines, each node executes the same instruction stream. The latest example of this is the Maspar MP-1, remarketed with additional software as the DECmpp by Digital The MP-1 has up to 16K four-bit processors and one Gigabyte (10^9 bytes) of memory and approximately one GigaFLOPS (10^9 floating point operations per second) peak performance. The Connection Machine CM-1, CM-2, and CM-200 are also SIMD distributed memory machines.

Thinking Machines surprised the community by changing the architecture of their latest CM-5, shown in Figure 7, to be the so-called MIMD distributed memory architecture. Again, we have a coupled collection of nodes—each with memory and processor—but now each node can execute its own instruction stream. The largest CM-5 delivered has 1,024 nodes, 32 Gigabytes of memory, and can, on some applications, realize 80 GigaFLOPS. The CM-5 installations are not fully implemented, and the largest parallel computer in operation today is the Intel "Delta Touchstone" System at Caltech, shown in Figure 8. This is also a MIMD distributed memory machines with 528 nodes, but a very different node interconnection scheme. Intel's Touchstone family continues to evolve, and the latest "Paragon" model, which will be available later this year, should have similar performance to the CM-5. The nCUBE, shown in Figure 1, also has the MIMD distributed memory design.

All the parallel machines discussed above are "scalable" and available in configurations that vary from small \$100,000 systems to a full size supercomputer at approximately \$30,000,000; the number of nodes and performance scales approximately linearly with the price. The DECmpp (Maspar) has deliberately aimed at the low end of the market and the largest 16K system costs factor of 25–50 less than the 1024-node CM-5 discussed above. In fact, as all the machines use similar VLSI technology, albeit with designs that are optimized in different ways, they very crudely have similar price performance. This, as shown in Figure 2, much better than that of conventional vector supercomputers, such as those from Cray, IBM, and Japanese vendors.

Current Cray and IBM supercomputers are also parallel with up to 16 processors in the latest CRAY C-90. their architecture is MIMD shared memory with a group of processors accessing a single global memory. This design is also seen on machines like the Sequent and high-end Silicon Graphics workstations, but all these machines have a modest number ($\leq$ 32) processors as they are hard to scale to many processors, i.e., to "massive parallelism."

An ingenious compromise is seen in the recent Kendall Square machine KSR-1, and the experimental DASH computer at Stanford. These have a

Figure 7: The CM-5 Produced by Thinking Machines

Figure 8: The "Delta Touchstone" Parallel Supercomputer Installed at Caltech and Produced by Intel

so-called virtual shared memory. The machine is built with a distributed memory, but special hardware (caches) and software make this distributed memory "act" as though it is globally available to all processors.

Currently, the dominant parallel computer vendors are American with only modest competition from Europe, with systems built around the transputer chip from Inmos. Japanese manufacturers have so far made little contribution to this field, but as the technology matures, we can expect them to provide formidable competition.

The Best Architecture?

Each of the machines and architectures described above have both strong and weak point, as they have been optimized in different ways.

The shared and virtual shared memory architectures have been designed for easier software, and in particular, for easier porting of existing Fortran codes. It is, however, difficult to scale them to large systems, and retain good cost performance.

The data parallel methodology described earlier fits well with distributed memory machines, but substantial reworking of software is needed so that compilers can exploit the inherent parallelism of the problem. However, distributed memory machines are clearly scalable to very large systems, and with the appropriate software, the vast majority of large-scale problems will run on them. The trade-off between SIMD and MIMD is also reasonably well understood in terms of a problem classification introduced by Fox. Regular applications, such as the matrix operations seen in Figure 6 are suitable for SIMD machines; MIMD computers can perform well on both these and the irregular problems typified by the particle dynamics simulation in Figure 5. We estimate that roughly half of existing large supercomputer simulations could use SIMD efficiently with the other half needing the extra flexibility of the MIMD architecture.

The hardware and software are evolving so as to integrate the various architectures. In the future, the user will be presented with a uniform interface to the different parallel machines. One will be able to make choices, as for conventional machines, based on the parallel computer's performance on one's application mix. One will not be faced, as in the past, with radically different software environments on each design. Future architectural developments will improve performance by moving critical functionality from software to hardware; this will offer the user increased performance from an unchanged software model.

Software

The adoption of parallel machines as a mainstream computing tool is held up by the lack of application codes that can run on them. Most successful uses of parallel machines have come from academic and research applications with less than 10,000 lines of code and where the software and parallel algorithm have been developed from scratch. The task of reimplementing large codes, such as the important 100,000 line CHARMM chemistry application for parallel machines is highly nontrivial.

The need to rework the software is clearly the major inhibitor to the rapid adoption of parallel machines. However, there has been significant progress in developing "portable scalable languages and software environments." These allow us to reimplement or develop new applications with the assurance that the resultant software will run well on all the current and projected parallel machines for which the problem is suitable.

There are no compelling new "parallel languages," but rather the successful approaches have extended existing languages. We will briefly discuss Fortran here, but similar remarks can also be made for C, C++, Ada, Lisp, etc. There are two classes of extensions to Fortran, which we discuss in turn.

Data Parallel Fortran

Here, parallelism is represented by a sequence of array operations where each element is calculated independently by user defined or system library functions—these allow one to add or multiply arrays and vectors, find maxima and combine such elemental operations. The user aids the compiler in implementing parallel array operations with commands that lay out the arrays over the nodes of the parallel machine. The best known example of such a data parallel language is CMFortran, developed by Thinking Machines. Currently, the "High Performance Fortran Forum" expects to produce an informal industry standard by the end of 1992 for a data parallel Fortran suitable for both SIMD and MIMD machines. The initial version of this standard is based on several research ompilers, including FortranD, which is an extension of Fortran77 and Fortran90, developed by a collaboration between Rice and Syracuse universities. High Performance Fortran offers a uniform software environment, which will allow the user to develop applications independently of the different hardware architectures discussed above—this is what we mean by a scalable portable software system.

Message Passing Fortran

We expect that data parallel versions of Fortran will eventually be able to efficiently support a large fraction of large science and engineering simulations. However, more general and less demanding on the compiler, are extensions of Fortran which allow the user to explicitly generate the messages needed on MIMD machines. The resultant "Fortran plus message passing model" (Fortran + MP), is suitable for all problems for which Fortran is a reasonable language on a conventional computer. Fortran + MP is usually more time consuming for the user to develop than data parallel Fortran, and is only suitable for MIMD machines. However, in this broad class (MIMD) of distributed or shared memory machines, Fortran + MP is portable and scalable using such message passing systems as Express, Zipcode, Linda, or PICL. An industry standards forum is just starting for message passing.

Distributed Computing and Operating Systems

A "real" software environment for parallel machines must offer many other services besides the parallel language to meet expectations of users of conventional (super)computers. Operating services for SIMD machines are provided by the UNIX host and MIMD machines have also used this mechanism up to now. Intel's new Paragon i860-based machine will feature Mach on each processor. Mach is a sophisticated version of UNIX optimized for distributed computing. Mach currently needs to be augmented by fast dedicated message passing to properly support parallel applications. This highlights the distinctions between "parallel" and "distributed" computing. A distributed network of, say, workstations is indeed a MIMD distributed memory parallel processor. However, this hardware and its current support software, such as Mach, has lower communication bandwidth and, in particular, higher start-up costs (communication latency) than dedicated parallel machines with their high-performance interprocessor connection network.

Operating System Services

Modern parallel computers offer parallel disk systems, which will allow many applications to match the high compute performance of Figure 2 with scaling disk I/O (input/output) performance. The software and methodology for accessing these parallel disks is still rudimentary and more experience is needed to develop this. Particularly interesting is the multicomputers with initial availability on the nCUBE and Meiko systems.

Parallel debuggers are available, and these are clear extensions from the sequential environment for the data parallel applications that dominate science and engineering simulations. Monitoring and evaluation of the performance of the computer is particularly important for parallel machines as it can signal poor decomposition and other inhibitors to good use of the machine. Another new software tool will automatically decompose and distribute problems over the nodes of a parallel machine. This has been demonstrated, but not yet implemented as a robust portable tool.

Applications

Most experience on parallel machines has been with academic and research problems. However, the field can only realize its full potential and be commercially successful if it is accepted in the real world of industry and government applications. Some of these are seen in the grand challenges, which are the focus of the federal high-performance computing and communications initiative.

However, more generally, parallel computing offers U.S. industry the opportunity of a global competitive advantage. This is a technology where the U.S. has a clear lead over Europe and Japan, and this technology leadership can be turned into a potent "weapon" in the global economic "war" we expect in the 1990s.

We have surveyed some of these industrial applications of parallel computers as part of a new project, ACTION, funded at Syracuse University by New York State.

Interesting possibilities include:

- use in the oil industry for both seismic analysis of new oil fields, and the reservoir simulation of existing fields;
- environmental modeling of past and potential pollution in air and ground;

- fluid flow simulations of aircraft, and general vehicles, engines, air-conditioners, and other turbomachinery; integration of structural analysis with the computational fluid dynamics of airflow; car crash simulation;

- design of new drugs for the pharmaceutical industry by modeling new compounds;

- simulation of electromagnetic and network properties of electronic systems—from new components to full printed circuit boards;

- identification of new materials with interesting properties, such as superconductivity;

- simulation of electrical and gas distribution systems to optimize production and response to failures;

- production of animated films and educational and entertainment uses, such as simulation of virtual worlds in theme parts and other virtual reality applications;

- support of geographic information systems, including real-time analysis of data from satellite sensors in NASA's "Mission to Planet Earth"; and

- a relatively unexplored area, known as "command and control" in the military area and "decision support" or "Information Processing" in the civilian applications. These combine large databases with extensive computation. In the military, the database could be sensor information, and the processing a multi-track Kalman filter. Commercially, the database could be the nation's medicaid records, and the processing would aim at cost containment by identifying anomalies and inconsistencies. My survey of New York State industry would suggest that this combination of databases, and significant computation is the largest opportunity for parallel computing.

References

[Almasi:89a] George S. Almasi and Allan Gottlieb. *Highly Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1989.

[Andrews:92a] Gregory R. Andrews. *Concurrent Programming: Principles and Practice*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1991.

[Arbib:90a] Michael Arbib and J. Alan Robinson, editors. *Natural and Artificial Parallel Computation*. The MIT Press, Cambridge, MA, 1990.

[Brawer:89a] Steven Brawer. *Introduction to Parallel Programming*. Academic Press, Inc. Ltd., London, 1989.

[Doyle:91a] John Doyle. Serial, parallel, and neural computers. *Futures*, 23(6):577–593, 1991. (July/August).

[Duncan:90a] Ralph Duncan. A survey of parallel computer architectures. *Computer*, 23(2):5–16, 1990.

[Fox:88a] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker. *Solving Problems on Concurrent Processors*, volume 1. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.

[Golub:89a] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1989. 2nd Edition.

[Hayes:89a] J. P. Hayes and T. Mudge. Hypercube supercomputers. *Proceedings of the IEEE*, 77(12):1829–1841, 1989.

[Hennessy:91a] John J. Hennessy and Norman P. Jouppi. Computer technology and architectures: An evolving interaction. *IEEE Computer*, pages 18–29, 1991.

[Hillis:85a] W. D. Hillis. *The Connection Machine*. MIT Press, Cambridge, MA, 1985.

[Hockney:81b] R. W. Hockney and C. R. Jesshope. *Parallel Computers*. Adam Hilger, Ltd., Bristol, Great Britain, 1981.

[Lazou:87a] C. Lazou. *Supercomputers and Their Use*. Oxford University Press, Oxford, Great Britain, 1987.

[Messina:91d] Paul Messina and Almerico Murli, editors. *Practical Parallel Computing: Status and Prospects*. John Wiley and Sons, Ltd., Sussex, England, 1991.

[Nelson:90b] M. E. Nelson, W. Furmanski, and J. M. Bower. Brain maps and parallel computers. *Trends Neurosci.*, 10:403–408, 1990.

[Salmon:89b] John Salmon, Peter Quinn, and Mike Warren. Using parallel computers for very large N-body simulations: Shell formation using 180K particles. In A. Toomre and R. Wielen, editors, *Proceedings of the Heidelberg Conference on the Dynamics and Interactions of Galaxies*. Springer-Verlag, April 1989.

[Skerrett:92a] P. J. Skerrett. Future computers: The Tera Flop race. *Popular Science*, page 55, 1992.

[Stone:91a] Harold S. Stone and John Cocke. Computer architecture in the 1990s. *IEEE Computer*, pages 30–38, 1991.

[SuperC:91a] *Proceedings of Supercomputing '91*, Los Alamitos, California, 1991. IEEE Computer Society Press.

[Trew:91a] Arthur Trew and Greg Wilson. *Past, Present, Parallel: A Survey of Available Parallel Computing Systems*. Springer-Verlag, Berlin, 1991.

[Zima:91a] Hans Zima and Barbara Chapman. *Supercompilers for Parallel and Vector Computers*. ACM Press, New York, 1991.