SCCS-736

Basic Issues and Current Status of Parallel Computing—1995

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 1.7×10^6 transistors—a factor of 30 increase. In 1995, the 167 Mhz Ultrasparc contained 5.2×10^6 transistors divided roughly 2:1 btween CPU and cache. The dramatic improvement in chip density comes together with an increase in clock speed and improved design so that today's leading chips deliver over a factor of 5,000 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 was 0.75μ for the alpha in 1992, and 0.5μ for the 1995 Ultrasparc. 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 most mainframe functionality to about 2 cm^2 of a chip. This enabled the personal computing and workstation revolutions. The next factors of 10 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 IBM SP-2, CRAY T3D, Thinking Machines CM-5, and Intel Paragon where each node is a sophisticated multichip printed circuit board. Another major debate is the choice of communication system to link the nodes together. This can vary from the closely coupled hypercube network in the nCUBE to the use of existing or specialized local area networks to link nodes that are conventional computers. 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 illustrates this as a function of time showing, already today, an approximate 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.

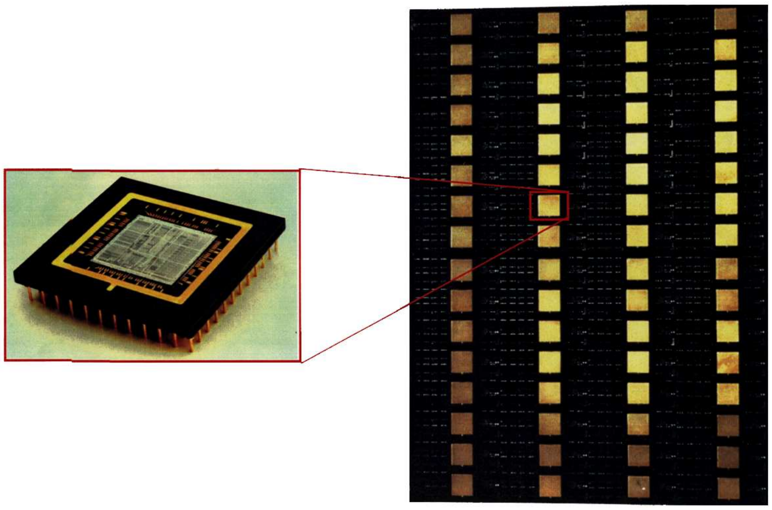

Figure 1: The nCUBE-2 Node and Its Integration into a Board. Up to 128 of these boards can be combined into a single supercomputer.

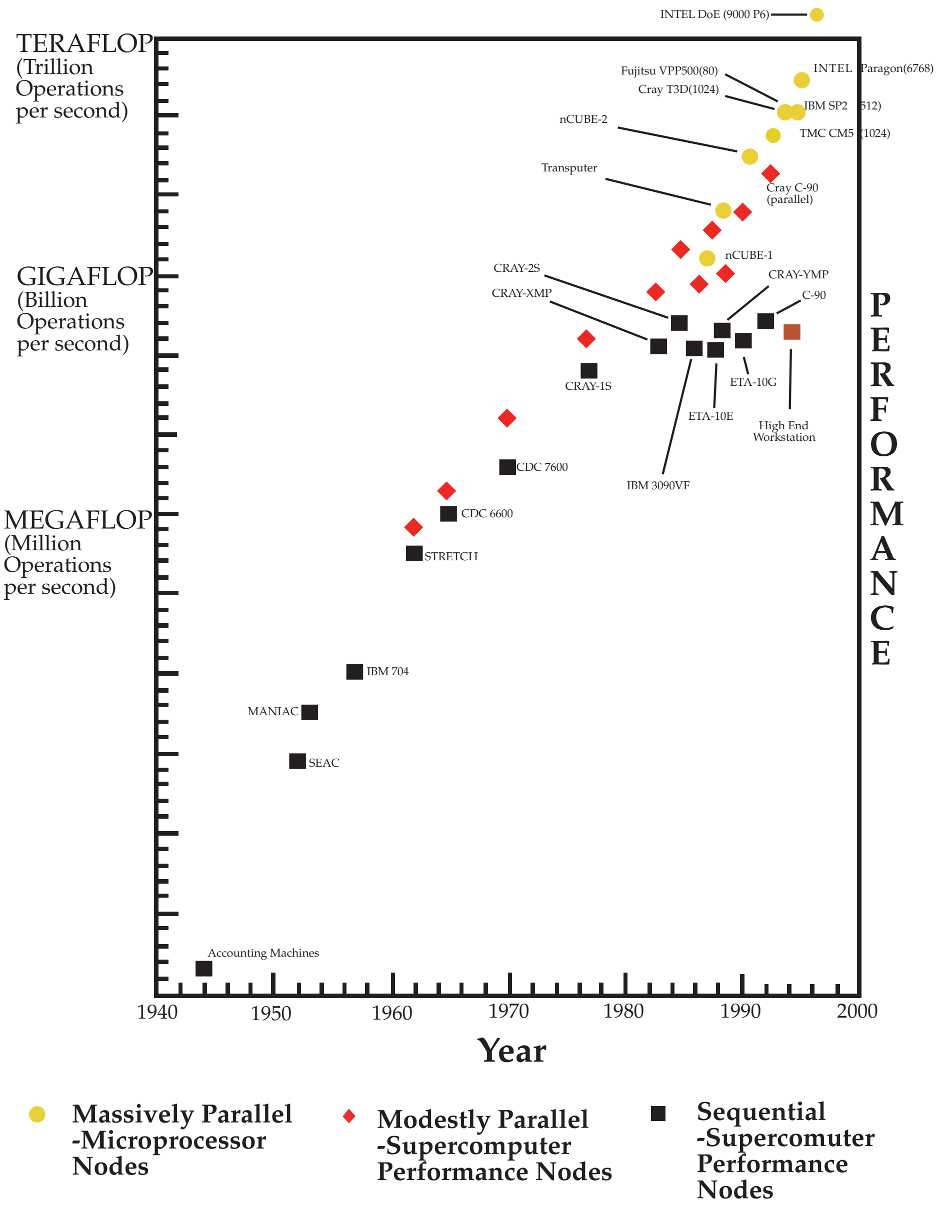

Figure 2: Performance of Parallel and Sequential Supercomputers

Grand Challenges

The President instituted, in 1992, the five-year federal High Performance Computing and Communications Initiative. This has spurred the development of the technology described above and was initially 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. ARPA is developing the basic technology, which are applied to the grand challenges by DOE, NASA, NSF, NIH, EPA, and NOAA. Many of these agencies are also playing a critical role in technology development, while DoD has initiated a major computer modernization program to integrate HPCC technology into their infrastructure. 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.

The successes with grand challenges are well documented in the Federal 1996 "blue book", which is available on the Web. However, much attention has shifted recently to a set of companion problems—the so called National Challenges—which emphasize large scale information processing and distributed systems. These areas include digital libraries, health care, education, manufacturing, and crisis management, and we expect comparable major impact from the use of HPCC technologies, even though raw number-crunching performance will not be the critical issue.

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

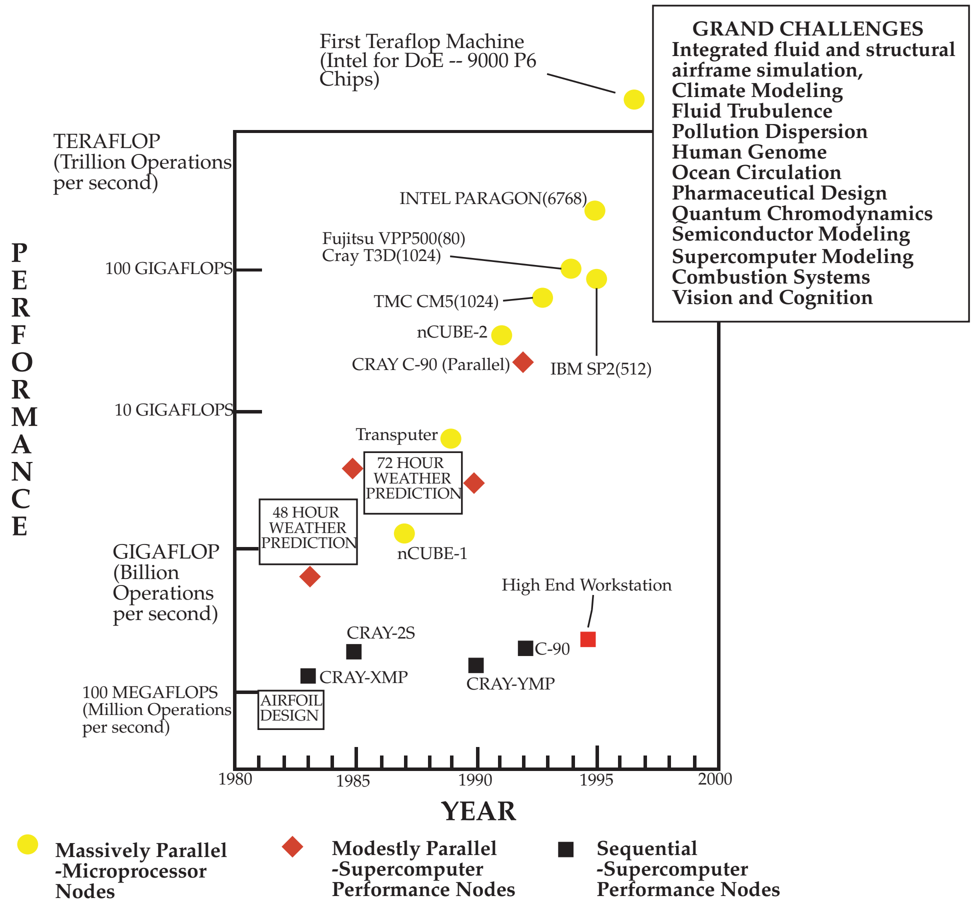

Figure 3: Grand Challenge Applications. Some major applications that will be enabled by parallel supercomputers.

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.

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.

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.

Many large scale computations, such as those from the fields of chemistry and electromagnetism, 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

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].

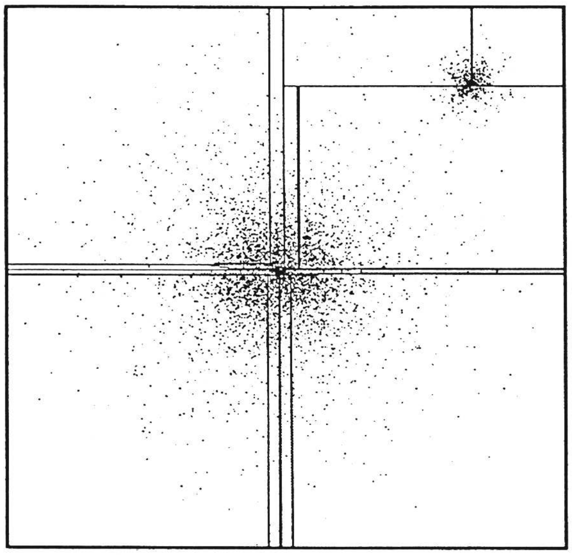

Figure 5: A Two-dimensional Projection of a Model Universe in which Two Galaxies are on a Collision Course. This is a simplified version with 18,000 "stars" of a large simulation reported in [Salmon:89b]. The irregular decomposition onto a 16-node machine is illustrated above.

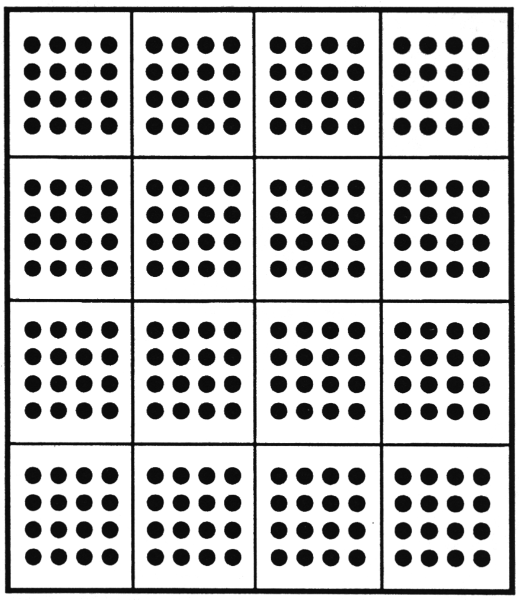

Figure 6: 16×16 Matrix Decomposed onto a 4×4 Parallel Computer Array

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, SCALA-PACK is 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 1995. There are three broad classes of machines. The first is the so-called SIMD, Single Instruction Multiple Data 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 MP-2 has up to 16K 32-bit processors and one Gigabyte (10^9 bytes) of memory and approximately six GigaFLOPS (10^9 floating point operations per second) peak performance. The Connection Machine CM-1, CM-2, and CM-200 from Thinking Machines, and the AMT DAP are also SIMD distributed memory machines.

The MIMD distributed memory architecture is the second major importance architecture, and recent large MPPs (Massively Parallel Processors) have all been of this design where both memory and processing capability are physically distributed. An influential machine of this class is the CM-5 from Thinking Machines, shown in Figure 7, which was a radical departure from their previous SIMD architectures, and this symbolized a growing realization that MIMD architectures were the design of choice for general applications that required the key MIMD characteristic that each node can execute its own instruction stream. The largest CM-5 configuration has 1,024 nodes, 32 Gigabytes of memory, and on some applications, can realize 80 GigaFLOPS. However, the more recent 512-node IBM SP-2 installed at Cornell will outperform the larger number of nodes on the Los Alamos CM-5 system. This illustrates the importance of using the best available sequential node—in IBM's case a powerful RS6000 RISC chip. The CM-5 was handicapped by its custom VLSI on the node. Even if the hardware design was optimal (problematical for the custom CM-5 vector node), we are on a very short technology cycle, and we cannot build the necessary software (in this case compilers) to support idiosyncratic hardware. Any new custom architecture must recognize that its competition—current designs implemented with decreasing feature size—automatically double their performance every 18 months without architecture innovation. All current machines, except the nCUBE, have firmly centered their MIMD parallel systems on pervasive PC or workstation technology—IBM (RS6000), CRAY (Digital alpha), Meiko (Sun), SGI (MIPS) and Convex (HP). Intel just announced that they will deliver to DoE, a TeraFLOPS computer built around their new P6 processor, which has over 10 million transistors packaged in a two-chip (processor and CPU) module. This follows a successful set of machines built around the i860 chip set, which includes a major 1,840-node system at Sandia with the Intel "Delta Touchstone" shown in Figure 8 being particularly influential as the first large-scale production MPP supercomputer. Interestingly, DoE is targeting the Intel TeraFLOPS system at simulations of existing nuclear weapons whose continued performance and maintenance is unclear in

Figure 7: The CM-5 Produced by Thinking Machines

a world where experimental testing is forbidden.

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. 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, is much better than that of conventional vector supercomputers, such as those from Cray, IBM, and Japanese vendors. These current Cray and Japanese vector supercomputers are also parallel with up to 16 processors in the very successful CRAY C-90. Their architecture is MIMD shared memory with a group of processors accessing a single global memory.

This shared memory MIMD design is the third major class of parallel architecture. In the vector supercomputer, one builds the fastest possible

Figure 8: The "Delta Touchstone" Parallel Supercomputer Installed at Caltech and Produced by Intel. This system uses a mesh architecture linking 512 nodes, and was a prototype for the Paragon.

processor node. This minimizes the number of nodes in the final system (for given cost), but given that parallelism is inevitable and that most problems are not restricted in number of nodes that can be used effectively, an attractive choice is to use the most cost-effective nodes. These are precisely nodes used in the most successful distributed memory MIMD machines. This class of shared memory machine includes the Silicon Graphics (SGI) Power Challenge Systems, whose major sales at the low end of the market have led to a rapid growth in the importance of SGI as a key player in the high-performance computing arena. Correspondingly, the shared memory architecture is receiving increasing attention, which can be examined in a little more detail. Shared memory systems were always considered limited, as the technology such as the bus needed to implement it, would not scale, and indeed, such (bus based) systems are limited to at most 16–32 nodes. However, nodes have increased in power so much that a "modest" 32-node shared memory system costing around $2 million is a major supercomputer for most users. Shared-memory systems have always had the advantage that it is easier to implement attractive software environments. The net result is that many expect shared memory to continue to grow in importance and become a dominant feature of mainstream MPP systems. Burton Smith's new Tera shared-memory supercomputer implements this with a special pipelining algorithm so that all processors can access all memory locations in a uniform time. However, most expect a clustered or virtual shared memory architecture with NUMA—nonuniform memory access time. Here machines, such as the Convex Exemplar, new SGI systems, the Stanford experimental DASH, and the now defunct Kendall Square KSR-1,2 are built with a distributed memory, but special hardware and software makes the 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.

Clusters of Workstations—The Informal Supercomputer

Above, we discussed carefully designed systems with special high-speed networks linking nodes usually derived from commercial PC or workstation technology. However, many have explored using workstations linked with conventional networks, such as Ethernet, fddi, or ATM as an informal parallel system. Such COWs or NOWs (clusters or networks of workstations) are clearly MIMD-NUMA distributed memory parallel machines. They should be able to run any software or parallel algorithm designed for machines such as the IBM SP2 or Intel Paragon. However, the latter machines have higher bandwidth and lower latency communication. Thus, they can support many more applications with high efficiency. However COWs are attractive because experience has shown that they do perform many parallel tasks effectively. They offer many political and fiscal advantages. COWs can be formed from existing "idle" workstations and PCs overnight and on weekends. In principle, no cost is involved other than that already incurred for desktop computers to support the organization.

These ideas can be generalized to *metacomputing* where any arbitrary set of heterogeneous computers are linked together to address a single problem.

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 has, however, proved difficult to scale them to large systems, and retain good cost performance. However, many believe that new hardware developments may change this. This MPP trend to shared memory should be contrasted with the opposite tendency seen with metacomputing, which is probably the most rapidly growing area and clearly distributed memory.

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 estimated in 1990 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 increasing interest in adaptive irregular algorithms which require MIMD systems, is decreasing the relevance of SIMD machines.

The hardware and software are evolving so as to integrate the various architectures. In the future, the user will hopefully be presented with a uniform interface to the different parallel machines. Although it is not clear that MPPs and heterogenous metacomputers can effectively be supported with the same software model. 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, such as the illusion of shared memory, 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 to over 1,000,000 line industrial applications 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. SIMD machines from AMT, Maspar, and Thinking Machines first popularized this approach with languages, such as CM Fortran. These ideas were extended to MIMD systems, and the latest research data parallel languages can handle complex irregular applications. An industry standard HPF or High-Performance Fortran has been adopted, and the first commercial compilers are now becoming available. 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 PVM and the new industray standard MPI (Message Passing Interface). Some metacomputing and, in particular, use of novel World Wide Web technology is driving much interesting work in this area.

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 “host-node” mechanism until recently. The IBM SP-2, from the start, adopted a different strategy with full UNIX available democratically on each node. This allows the machine to either be viewed as a highly coupled parallel or as a distributed system. These are still important software issues remaining in the integration of these two faces of parallelism. However, this appears to be the way of the future, and naturally links COWS, metacomputing, and MPPs.

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. A major initiative—the Scalable I/O Initiative—is being led by Messina from Caltech in this area. Particularly interesting is the introduction of a commercial relational database, ORACLE, on distributed memory multicomputers with the IBM SP-2 implementation particularly attractive as IBM has already such a strong presence in the (conventional computer) commercial world.

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. Other software tools can automatically decompose and distribute problems over the nodes of a parallel machine.

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 described earlier.

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 explored industrial applications of HPCC, and this is discussed in another article in this book. This work—funded by a New York State technology transfer activity, InfoMall—emphasizes that we expect that the dominant industrial use of MPPs will be in information related areas with large-scale MPP numerical simulations playing a secondary role.

References

[Andrews:91a] Andrews, G. R. *Concurrent Programming: Principles and Practice.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1991.

[Almasi:94a] Almasi, G. S., and Gottlieb, A. *Highly Parallel Computing.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994. second edition.

[Angus:90a] Angus, I. G., Fox, G. C., Kim, J. S., and Walker, D. W. *Solving Problems on Concurrent Processors: Software for Concurrent Processors*, volume 2. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.

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

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

[Chandy:92b] Chandy, K. M., and Taylor, S. *An Introduction to Parallel Programming.* Jones and Bartlett, 1992.

[CSEP:95a] "Computational science educational project." Web address http://csep1.phy.ornl.gov/csep.html.

[Dongarra:94a] Dongarra, J., van de Geign, R., and Walker, D. "Scalability issues affecting the design of a dense linear algebra library," *J. Parallel and Distributed Computing*, 22(3):523–537, 1994.

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

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

[Foster:95a] Foster, I. *Designing and Building Parallel Programs.* Addison-Wesley, 1995. http://www.mcs.acl.gov/dbpp/.

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

[Fox:94a] Fox, G. C., Messina, P. C., and Williams, R. D., editors. *Parallel Computing Works!* Morgan Kaufmann Publishers, San Francisco, CA, 1994. http://www.infomall.org/npac/pcw/.

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

[Gropp:95a] Gropp, W., Lusk, E., and Skjellum, A. *Using MPI: Portable Parallel Programming with the Message Passing Interface.* MIT Press, 1995.

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

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

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

[HPCC:96a] National Science and Technology Council, "High performance computing and communications," 1996. 1996 Federal Blue Book. A report by the Committee on Information and Communications. Web address http://www.hpcc.gov/blue96/.

[HPF:93a] High Performance Fortran Forum. "High performance Fortran language specification." Technical Report CRPC-TR92225, Center for Research on Parallel Computation, Rice University, Houston, Texas, 1993.

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

[Koelbel:94a] Koelbel, C., Loveman, D., Schreiber, R., Steele, G., and Zosel, M. *The High Performance Fortran Handbook.* MIT Press, 1994.

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

[Messina:91d] Messina, P., and Murli, A., editors. *Practical Parallel Computing: Status and Prospects.* John Wiley and Sons, Ltd., Sussex, England, 1991. Caltech Report CCSF-13-91.

[McBryan:94a] McBryan, O. "An overview of message passing environments," *Parallel Computing*, 20(4):417–444, 1994.

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

[Salmon:89b] Salmon, J., Quinn, P., and Warren, M. "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. Caltech Report C3P-780b.

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

[Stone:91a] Stone, H. S., and Cocke, J. "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.

[SuperC:92a] *Proceedings of Supercomputing '92*, Los Alamitos, California, November 1992. IEEE Computer Society Press. Held in Minneapolis, Minnesota.

[SuperC:93a] *Proceedings of Supercomputing '93*, Los Alamitos, California, November 1993. IEEE Computer Society Press. Held in Portland, Oregon.

[SuperC:94a] *Proceedings of Supercomputing '94*, Los Alamitos, California, November 1994. IEEE Computer Society Press. Held in Washington, D.C.

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

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