Parallel Computers Take Lead in Complex Problem Solving

Michael R. Leuze *
Mathematical Sciences Section
Oak Ridge National Laboratory
P. O. Box 2008, Building 6012
Oak Ridge, TN 37831–6367
(615) 574–3146
leuze@msr.epm.ornl.gov

Robert C. Ward *
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P. O. Box 2008, Building 6025
Oak Ridge, TN 37831–6359
(615) 574–6185
ward@msr.epm.ornl.gov

Abstract

Parallel computing concepts are not new; many early computers used parallelism in some form. However, it is only within the last 20 years that true multiprocessor systems have been constructed, and only within the last 6 to 8 years that these systems have become cost effective. Today, the most powerful computers in existence are parallel. Despite many remaining problems, numerous applications programs at the leading edge of scientific research are exploiting the power of these machines.

*Supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy.

1 Introduction.

What is parallel processing? Hockney and Jesshope discuss four primary ways of introducing parallelism into computing systems in their book **Parallel Computers** [10]. The first, *pipelining*, is based on the same concept as assembly lines in an industrial plant. A pipeline consists of a number of hardware stages, each designed to perform a specific subtask. This approach allows the computing system to have several computations active at once, at various stages of completion. The second, *functional parallelism*, involves duplication of logic or arithmetic units, which are then capable of simultaneous operation. The third, *array parallelism*, implies a collection of identical processing elements under common control. Since each processing element receives instructions from a controller but operates on its own private data, these machines are commonly called SIMD, or *Single Instruction Multiple Data*, computers. The fourth approach, *multiprocessing*, denotes a collection of tightly-coupled processors, each controlled independently. Each processor executes its own process using its own data. Consequently, these machines are called MIMD, or *Multiple Instruction Multiple Data*, computers. MIMD machines are further categorized as shared-memory computers, if all processors can access a large global memory, or distributed-memory computers, if each processor can directly access only its own local memory. In computer architectures currently being designed and implemented, it is not unusual to have two, three, or even four forms of parallelism present in the same system. For a more complete taxonomy of parallel computers, as well as speculation on future machines, see [2].

Why parallel processing? What is the driving force behind the expenditure of resources and effort to build and program more and more powerful tightly-coupled multiprocessor systems? The motivation for the development of more powerful systems comes from the need to address bigger, more realistic applications problems than can be solved using the totality of resources available on today's largest and fastest machines. There are, in fact, hundreds of such applications in many disciplines: physics, chemistry, economics, medical imaging, materials science, structural analysis, and computational fluid dynamics [4,7]. As Kuhn and Padua have pointed out, "Whenever a computer designer has reached for a level of performance beyond that provided by his contemporary technology, parallel processing has been his apprentice" [12, p. 1]. We can expect an increase of perhaps one or two orders of magnitude in computer system performance due to technological advances, but as fundamental physical limitations are approached, future performance increases will come from the introduction and exploitation of parallelism.

2 The Past.

Parallel processing is not a new idea. From references, such as that of Menabrea [11, p. 46], it is clear that Charles Babbage, the designer of a mechanical computing device, was familiar with the idea of parallel computation in the early 19$^{\text{th}}$ century. A number of early digital computers incorporated parallel features, such as add or multiply units that could function autonomously. During the 1960s and 1970s, there was a gradual introduction of parallel features into the overall serial design of scalar computers, and the first true multiprocessor systems began to emerge.

The development of the Illiac–IV, a distributed-memory SIMD computer, began in 1966 at the University of Illinois under a contract from the U. S. Department of Defense. Unfortunately, the Illiac–IV was too ambitious for the technology of its time. In spite of cost overruns and performance far below initial projections, the Illiac–IV pioneered new technologies and new architectural ideas and significantly influenced the development of commercial successors, including the Connection Machine [9] from Thinking Machines Corporation and the MasPar MP–1 [3].

The most common multiprocessors of the 1970s and 1980s have been shared-memory systems in which a relatively small number of processors (from 2 to 30) are linked to a global memory. Vendors manufacturing computers of this type include Alliant, Convex, Cray, Encore, and Sequent. The popularity of these systems stems primarily from ease of use. Transforming programs developed for serial computers is simpler if communication between processors takes place in a shared memory, rather than through the passing of messages, as is required with a distributed-memory machine. However, the power of shared-memory computers is limited, since the switching mechanism required makes it difficult to scale them to large numbers of processors.

Researchers at the California Institute of Technology pioneered the development of distributed-memory MIMD computers with binary n–cube, or hypercube, interconnection networks. These machines, the Cosmic Cube [18] and the Mark II and III hypercubes [16], were the predecessors of the commercial products of Intel [17] and nCUBE [14].

3 Recent Surge: 1985–Present.

In the mid 1980s, improvements in VLSI fabrication techniques resulted in the availability of powerful, inexpensive microprocessors (see Figure 1), and it became clear that massive parallelism was a cost-effective path to powerful computing systems. A number of vendors introduced scalable distributed-memory machines, including the BBN Butterfly, the Intel iPSC, and the nCUBE.

Over the next several years, significant increases in performance have resulted from improvements in hardware, software, and multiprocessor design. RISC (Reduced Instruction Set Computer) technology [15] has produced microprocessors as powerful as a Cray–1. The use of a message-passing technique known as "worm-hole routing" [5] greatly increased interprocessor communication efficiency. The introduction of parallel I/O, the simultaneous access of multiple storage devices such as disks, has given systems the ability to address computational problems that generate large quantities of data. Intel's decision to switch from a hypercube interconnection topology to a two-dimensional mesh allowed simpler, faster message routers to be used in their Touchstone Delta Prototype [13], currently the world's most powerful computer. Compact parallel systems, important in mobile robot design, space ex-

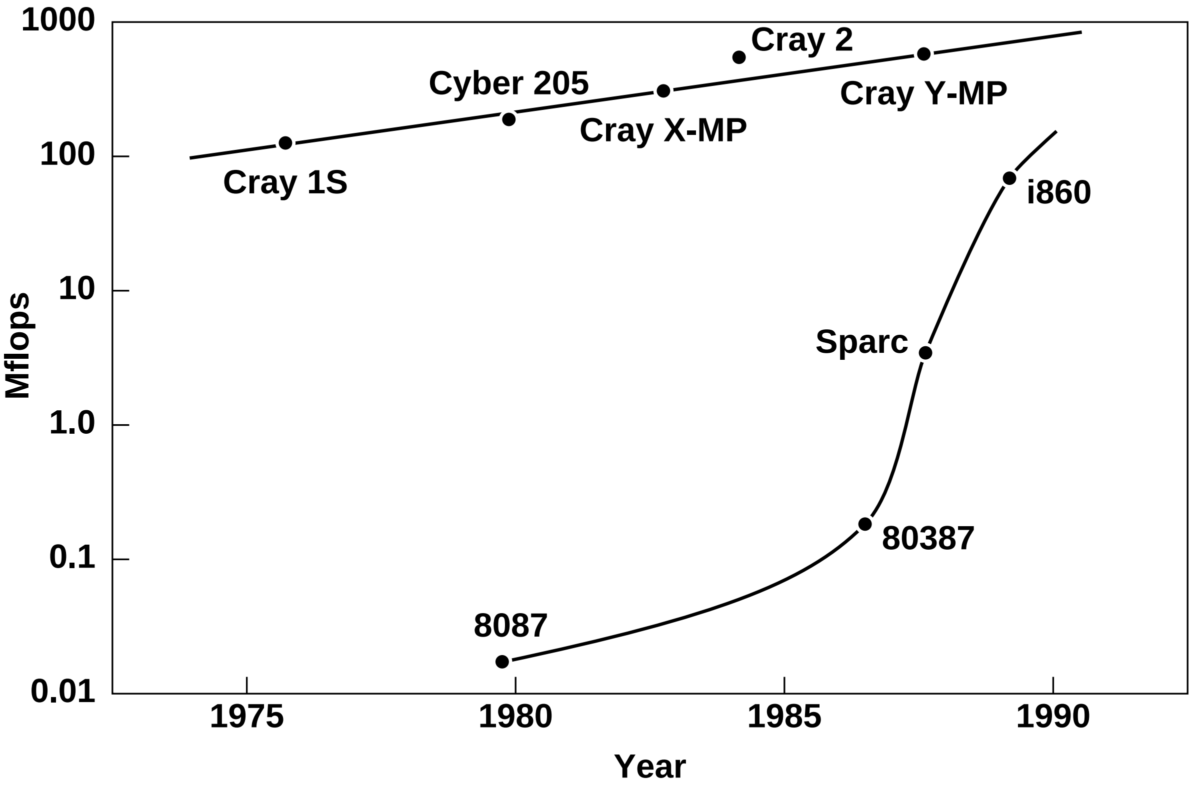

Figure 1: *Comparison of the* peak *64–bit floating point performance (Mflops) of supercomputers and microprocessors versus time.*

ploration, and many other applications, have been built by nCUBE using custom hardware and innovative packaging techniques.

Hardware and software advances have also made distributed-memory machines easier to use. For example, one way to make machines easier to program is to make a distributed memory appear to the user as a single shared memory. Hardware approaches to solving this problem have been adopted by BBN and by Kendall Square Research. Software tools, such as debuggers and performance monitors, are making programming more efficient. With a tool like ParaGraph [8] it is possible to view multiprocessor performance data in a variety of ways, thus increasing understanding of parallel program behavior.

Software systems like PVM [1] exploit the power of many processors by linking together a heterogeneous collection of serial and parallel machines, allowing them to function as a single computational unit. Groups of scientific workstations connected by PVM have achieved price/performance ratios superior to any existing multiprocessor system.

4 Applications.

Scientific discovery is on the verge of a new revolution. Just as the industrial revolution was triggered by the power of steam engines, this new revolution is being triggered by the power of high-performance computing engines. During recent years, the computational resources available to scientists and engineers have increased dramatically. As a result, researchers in all scientific fields are able to envision new approaches to fundamental problems. A new scientific paradigm, linking theory and experimentation, is emerging in the computing laboratory. Powerful new computers provide a means of modeling reality to a level of detail that was previously impossible. Through use of models, scientists and engineers are generating new knowledge and suggesting hypotheses for future investigation. Many experiments that once required complex and costly physical models are now being performed with computer models capable of revealing finer detail in a small fraction of the time.

Unfortunately, many difficulties still exist in developing parallel programs for multiprocessor systems. Porting existing programs, some of which have evolved over decades and consist of hundreds of thousands of lines of code, to parallel computers is difficult or impossible. Some programs designed for older serial machines must be completely restructured to exploit the power of parallel computers. Others are based on algorithms that are inherently sequential in nature and must be discarded. Automatic parallelizing compilers, which will eventually relieve the user from much of the tedium associated with parallel programming, are still very primitive. Most of the work must still be done by hand. Nevertheless, high-performance computers have already proven to be invaluable in a large number of applications areas, including

- *Superconductivity:* using lattice Monte Carlo simulation models in condensed-matter physics;
- *Particle Physics:* simulating particle interaction processes for development of massive detectors for the Superconducting Super Collider; designing experiments to study plasmas of quarks and gluons in the Relativistic Heavy Ion Collider (RHIC) at Brookhaven

Machine	Number of Processors	Performance (Gflops)	Problem Size
Intel Touchstone Delta	512	11.9	25000
Thinking Machines CM–2	2048	9.0	28672
Fujitsu AP1000	512	2.251	25600
nCUBE 2	1024	1.9	21376
Intel iPSC/860	128	1.9	8600

Table 1: *Performance data for solving linear systems on the largest massively parallel computers. Problem size is the order of the largest linear system that can be solved on a particular machine.*

National Laboratory;

- *Human Genome:* applying parallel network models to chromosomal band patterns;
- *Global Climate Change:* modeling the coupled atmosphere, ocean, biosphere system to understand the impact of chemical and energy releases on weather patterns;
- *Heavy-Ion Physics:* modeling the production of lepton pairs, a phenomenon important to beam stability and focusing, during high-energy heavy-ion nuclear collisions; and
- *Fluid Dynamics:* modeling the behavior of geologic oil reservoirs, the transport of groundwater, and the characteristics of wind tunnels.

These, and most other scientific applications, rely heavily on a relatively small core of linear algebra routines. Thus, performance of an application on a particular machine is related to the suitability of the machine for solving linear algebra problems. The use of linear system solvers has long been a standard technique for comparing computer system performances. Performance data for solving linear systems on the largest "massively parallel" computers known to the authors are included in Table 1 [6]. For comparison, an eight-processor Cray Y–MP is capable of solving an order 1000 linear system at a rate of 2.144 Gflops [6].

5 Conclusion.

The largest, most powerful computers available today are parallel systems. The fastest programs in applications areas that require the solutions of large, complex problems are parallel programs. In spite of the difficulties still associated with parallel computing, it is clear that the technology has reached a level of sophistication that makes its use worthwhile, and even essential, in scientific computing today.

References

[1] A. L. Beguelin, J. J. Dongarra, G. A. Geist, R. J. Manchek, and V. S. Sunderam, *Heterogeneous network supercomputing*, Supercomputer Review, (September 1991).

[2] G. G. Bell, *The future of high performance computers in science and engineering*, Communications of the ACM, 32 (1989), pp. 1091–1101.

[3] T. Blank, *The MasPar MP–1 architecture*, in Proceedings of COMPCON 90, IEEE Computer Society Conference, February 1990.

[4] B. L. Buzbee and H. J. Raveche, *Conference on forefronts of large-scale computational problems*, Parallel Computing, 1 (1984), pp. 307–315.

[5] W. J. Dally and C. L. Seitz, *Deadlock-free message routing in multiprocessor interconnection networks*, IEEE Transactions on Computers, C–36 (1987), pp. 547–553.

[6] J. J. Dongarra, *Performance of various computers using standard linear equations software*, Tech. Report CS–89–85 (Revised 9–13–91), University of Tennessee, 1991.

[7] G. C. Fox, *Parallel computing comes of age: Supercomputer level parallel computations at Caltech*, Concurrency: Practice and Experience, 1 (1989), pp. 63–103.

[8] M. T. Heath, *Visual animation of parallel algorithms for matrix computations*, in The Fifth Distributed Memory Computing Conference Proceedings, D. W. Walker and Q. F. Stout, eds., Los Alamitos, California, 1990, IEEE Press, pp. 1213–1222.

[9] D. W. Hillis, *The Connection Machine*, MIT Press, 1985.

[10] R. W. Hockney and C. R. Jesshope, *Parallel Computers*, Adam Hilger, Ltd., Bristol, England, 1981.

[11] D. J. Kuck, *A survey of parallel machine organization and programming*, Computing Surveys, 9 (1977), pp. 29–59.

[12] R. H. Kuhn and D. A. Padua, eds., *Tutorial on Parallel Processing*, IEEE Computer Society, 1981.

[13] S. L. Lillevik, *The Touchstone 30 Gigaflop DELTA Prototype*, in The Sixth Distributed Memory Computing Conference Proceedings, IEEE Computer Society Press, April 28 – May 1, 1991, pp. 671–677.

[14] J. F. Palmer, *A VLSI parallel supercomputer*, in Hypercube Multiprocessors, 1986, M. T. Heath, ed., SIAM, 1986, pp. 19–26.

[15] D. A. Patterson, *Reduced instruction set computers*, Communications of the ACM, 28 (1985), pp. 8–21.

[16] J. C. Peterson, J. O. Tuazon, D. Lieberman, and M. Pniel, *The Mark III hypercube-ensemble concurrent computer*, in Proceedings of the 1985 International Conference on Parallel Processing, IEEE Computer Society Press, August 1985, pp. 71–73.

[17] J. Rattner, *Concurrent processing: A new direction in scientific computing*, in Conference Proceedings of the 1985 National Computing Conference, vol. 54, AFIPS, 1985, pp. 157–166.

[18] C. L. Seitz, *The Cosmic Cube*, Communications of the ACM, 28 (1985), pp. 22–33.