OCCASION

This publication has been made available to the public on the occasion of the 50th anniversary of the United Nations Industrial Development Organisation.

DISCLAIMER

FAIR USE POLICY

CONTACT

Please contact publications@unido.org for further information concerning UNIDO publications.

For more information about UNIDO, please visit us at www.unido.org

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
Vienna International Centre, P.O. Box 300, 1400 Vienna, Austria
Tel: (+43-1) 26026-0 · www.unido.org · unido@unido.org

Parallel Computers and Their Industrial Applications

Boleslaw K. Szymanski
Department of Computer Science & Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

1 Challenges for Parallel Computers

Parallel computing has become a critical technology for manufacturing processes. It is also quickly gaining importance in sciences, medicine and the drug industry. Large-scale computer modeling impacts decision making in banking and finance, military and government. The Industrial Revolution of the 18th Century had freed humans from the enslavement of manual labor and had transformed craft and handiwork into the industries of today. Likewise, the Computer Revolution which we are witnessing now has been freeing the labor force from routine mental tasks which were and often are still done by assistants, clerks and low-level managers. Parallel computers form an important component of this revolution. They empower decision makers, such as high-level managers and chief scientists, with the ability to gather, access, and synthesize information, as well as to simulate real-life processes to measure the impact of social, economical and design decisions. The quality of the simulations and synthesized information is strongly dependent on the applied computational power. Today, even the largest uniprocessor computers are too slow for the most challenging problems of this kind.

In the United States, the quest for higher-speed machines is fueled by computationally intensive problems with profound economical and social impacts referred to as Grand Challenges [3]. It is difficult to list all Grand Challenge problems because so many areas of science and engineering are potential sources of such problems. The short list typically includes:

- High-resolution weather forecasting crucial for agriculture, disaster prevention, etc.
- Pollution studies that include cross-pollutant interactions, important in environmental protection.
- Global modeling of atmosphere-ocean-biosphere interactions to measure the long-term impact of human activities on the stability of the global ecosystem.
- Human genome sequencing that will assist in recognizing, preventing and fighting genetic diseases.

- The design of new and more efficient drugs to cure cancer, AIDS and other diseases.
- High-temperature superconductor design that can revolutionize computer design, electrical devices, etc.
- The aerodynamic design of aerospace vehicles (airflow modeling) and improvements in automotive engine design (ignition and combustion modeling) that can lead to more efficient use of depletable fossil fuels in transportation.
- The design of quantum switching devices important for building more powerful computers.

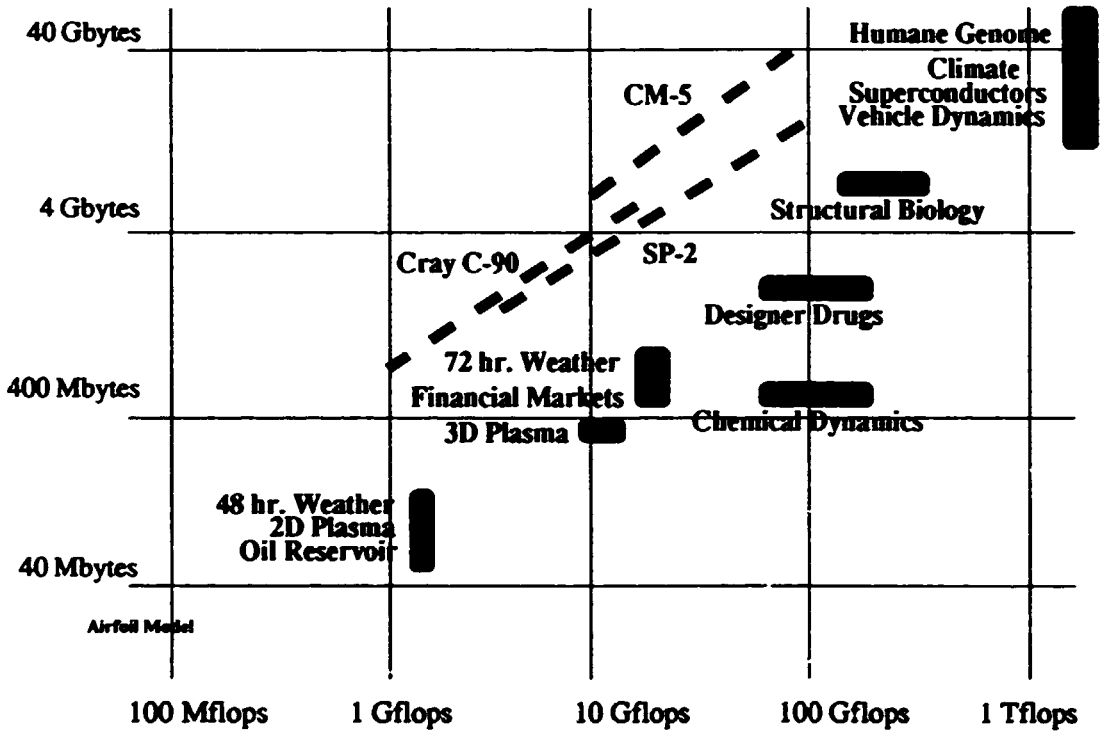

Figure 1: Computational Speed and Memory Required for Grand Challenge Models versus Current Parallel Machines

It is estimated that to achieve acceptable response time for these problems, in the order of several hours, will require a machine with performance of teraflops = 10^{12} floating point operations per second. Today's parallel computers approach a tenth of a teraflops, i.e., about 100 gigaflops (1 gigaflops = 10^9 flops). However, such speed is achieved only on certain very large, highly localized, finely tuned, often idealized applications. The demand for computational speed and memory for some applications from the above short list of Grand Challenges are shown in Figure 1. The position of some current parallel machines discussed in this article is also marked in Figure 1.

In this article, first the need for the parallel computers is justified on technological grounds. Then, the recent developments in parallel computer architectures are discussed, followed by a brief review of their software and limitations. Finally, the industrial applications are summarized and discussed.

2 Need for Parallel Computers

In recent years, it has become increasingly difficult to improve the performance of a uniprocessor based on the time-honored von Neumann model. By laws of physics, the speed of signal transmission in a computer cannot exceed the speed of light in the transmission media, about 3×10^7 m/sec for silicon. Consequently, it takes 10^{-9} of a second for a signal to propagate in a silicon chip of an inch in diameter. However, one signal propagation can support at most one floating point operation. Hence, a sequential computer built with a chip of such a size can provide at most 10^9 flops = 1 gigaflops, i.e., one-thousandth of the needed teraflops. The immediate conclusion is that the only feasible path to a teraflops computer leads through massively parallel machines (MPPs).

An interest in parallel computing systems is not new and can be traced back as far as the 1920s. However, as late as the early 1970s , major criticism of parallel processing was based on Grosch's law which states that the computing power of a single processor increases in proportion to the square of its cost. Recent careful analysis of Grosch's law showed that it is valid only within one technology. Economy of scale for mass-produced memory and RISC (Reduced Instruction Set) processors makes them a few orders of magnitude less expensive than custom designed chips for mainframes and traditional vector supercomputers. The improving computer chip technology enables the placement of ever-faster processors with ever-increasing amounts of memory on a single wafer. Hence, introduction of RISC technology made Grosch's law obsolete. Massively parallel computers built from a large number of RISC processors provide a superior performance-to-price ratio compared to computers based on the powerful, custom-designed CISC (Complex Instruction Set) processors.

The traditional vector supercomputers are built of a limited number of powerful processors connected to large shared memory. In addition, they explore array operation parallelism through vector coprocessors. As discussed below, because of the shared memory, the number of processors in such a parallel system cannot easily be increased and is limited to about 16. In contrast, massively parallel computers have processors with local memories. The processors are connected directly to each other by a network. The cost of such a parallel computer is roughly proportional to the requested number of processors. Therefore the size of the computer installation is more limited by costs than technical considerations. The massively parallel computers have three advantages over traditional vector supercomputers:

1. An accelerated rate of advance of peak processing power. In the last decade, microprocessor performance has increased four times every three years, following the rate of integrated circuit logic density improvement. By contrast, the clock rates of vector machines have improved much more slowly, doubling every seven years [2]. These trends are expected to continue for at least the 1990s.

2. An improvement in the performance-to-cost ratio. In 1993, this ratio was between two

to eight times higher for MPPs than for the vector supercomputers.

3. Scalability of the machine. The smallest configurations of MPPs are usually low priced to entice initial purchase (in 1993, the least expensive MPPs costed below $100.000). The initial configuration of the MPP can be upgraded incrementally as the needs and available funds arise.

The clear conclusion is that only massively parallel computers can deliver the much needed teraflops level of performance.

3 Architectures of Parallel Machines

In Flynn's well-known classification of parallel computational models [5], the von Neumann model is characterized by a single stream of instructions controlling a single stream of data (SISD). To achieve parallelism, multiple data streams can be introduced, thus creating a SIMD model. A further extension adds multiple instruction streams which leads to multiple instruction multiple data streams (MIMD) architectures. The last category splits into two classes on the basis of a memory access mechanism. One class, the shared-memory architecture, is characterized by the existence of a single global memory. Each processor has equal access to this memory. The other class, the distributed-memory architectures, have processors with local memories. Each processor has direct access to its own memory and indirect access to the memory of other processors. The indirect access is typically supported through a message-passing mechanism that enables processors to communicate with each other.

On SIMD machines, all processors execute the same statement but each operates on a different piece of data (data parallelism). The major task of a programmer is to identify data that can be distributed among the processors. However, programming itself is relatively simple because each processor executes the same sequential program. If, for a particular program step, a processor does not have any data assigned to it or the executed step does not apply to its data, the processor remains idle. For this reason, SIMD machines are efficient only in such applications that have enough operations applicable to a large number of data pieces.

An example of a massively parallel SIMD system is the MP-2 computer produced by MasPar. The MP-2 is built around an array of processors with a single circuit board containing 1024 processors. Each processor has 64 Mbytes of local memory. A processor can communicate with its eight nearest neighbors interconnected in a two-dimensional grid. Another means of communication is supported by the hypercube interconnected network and a global router that can deliver messages from any processor to any other processor. The Array Control Unit (ACU) controls the operations and communication of all the processors in the array. The front-end of the machine is a standard UNIX workstation with standard and high speed input/output subsystems. The front-end handles traditional serial processing. The MP-2 includes up to a 16,364 array processor and delivers up to 6.3 gigaflops with 32-bit precision arithmetic. MP-2 runs an operating system that is a derivative of the UNIX system and has optimizing compilers for MPL (a variant of the data parallel C language) and Fortran.

Programming for MIMD machines is more complex than programming for sequential or SIMD machines. For MIMD shared-memory architectures, the most difficult programmer's task is to map the program onto processors. Synchronization and data exchange can be efficiently implemented through blocks of shared memory. Programming of such machines is less difficult than for distributed-memory machines thanks to the global address space that makes any data uniformly accessible from any processor. The challenge is in the hardware support for shared memory. As the number of processors increases, so does the traffic in the network connecting processors with the memory. If the memory requests from the different processors are directed to the same memory bank, memory access is done sequentially slowing down the processors. Consequently, it is believed that shared-memory machines cannot support massive parallelism. The currently available architecture in this class is the Cray C-90 series which represents a traditional vector supercomputer with limited interprocessor parallelism [8]. Its largest configuration consists of 16 processors, each with a performance of 1 gigaflops and shared memory of 8 gigabytes. Each processor can have two vector pipes and two functional units active in a cycle, thus producing four vector results per clock unit. This parallelism of operations within each processor can be multiplied by 16 available processors resulting in the peak performance of 16 gigaflops. The Cray C-90 runs under the UNICOX operating system and has vectorizing compilers for Fortran and C.

Programming for distributed-memory machines inherits all the problems of the shared-memory programs and is further complicated by the the need for data distribution. Each processor has the direct access to the local memory only. Non-local data must be negotiated with the owner processes using communication. The synchronization imposed by the wait for a communicated data can significantly slow the performance of a computer. Subsection 4.1 discusses the Fortran extensions that allow the programmer to define data distributions. Another effort to ease the programmer's burden is to support non-local data access through hardware as done by the Kendall Square Corporation in the so-called all-cache KSR-1 machine. Although the memory of KSR-1 is distributed, the address space of the program is global. If the accessed data is not in local memory, the operating system suspends the process and brings the data to the processor. The efficiency of such a solution is being evaluated by the KSR-1 users [8].

The MIMD architecture, also capable of SIMD execution mode, is exemplified by the CM-5 computers produced by Thinking Machines Corporation [8]. The CM-5 machine consists of processing nodes (the configuration can vary from 32 to 16,384 processors), a number of control processors, a data network, a control network and a diagnostic network. Each processing node is a RISC processor with 32 Mbytes of memory and a 128 megaflops vector processing unit. Input and output are provided via a high-bandwidth interface. The data network is interconnected into a fat-tree and provides high-performance, point-to-point data communication between the processors. Unlike an ordinary binary tree, the channel capacities of a fat-tree increase as the tree is traversed from leaves to root. The control and diagnostic networks are implemented as binary trees; the first one provides cooperative operations such as broadcast and synchronization whereas the second one supports testing system integrity as well as detection and isolation of errors. The data parallelism in CM-5 can be implemented in either SIMD mode, multiple SIMD mode or synchronized MIMD mode. The reported performance of the 1024 processing node configuration was about 50 gigaflops. Theoretically, a full configuration of 16,384 processing nodes could reach teraflops

range but, with current pricing, such a machine would be prohibitively expensive.

The biggest promise of wide commercial use, in the opinion of the author, is the recently announced (end of the year, 1993) scalable SP-2 computer produced by IBM Corporation. It is an MIMD computer based on the RISC System/6000 processors. The system consists of three major components: the number of the RISC System/6000 processors, the high-performance switch, and the control processor. Each processor can perform at 250 megaflops. The high-performance switch is a multistage network with optical links. The switch is capable of a 40 Mbyte/sec processor-to-processor data transfer with a latency of about 3 microseconds. The software approximately doubles this latency. The predecessor of this machine, the SP-1, is about half as fast as the SP-2. Both systems run under AIX operating system and support PVM message-passing protocols. The Cornell Theory Center in Ithaca announced recently the replacement of its SP-1 machine with the 512-processor SP-2 computer with a peak performance of more than 100 gigaflops in 1994. The Theory Center plans to use the new system to introduce commercial users to scalable computing for such applications as modeling sedimentary basins to predict where oil is present, interactive access to large data sets, aerospace engineering, dissolution of natural gas, turbulent combustion and orthopedic biomechanics.

The size of the high performance computing market worldwide is about $2 billion (excluding sales of the IBM add-on vector hardware). The large share of this market is held by Cray Research which accounts for roughly 40% of sales. On the other hand, many MPP vendors have sales below $100 million. Clearly, the MPP industry is still in the early stages of development and it is very likely that some existing companies will disappear and new ones will emerge. However, in the opinion of this author, the direction of development towards MPP system will intensify.

4 Programming Models and Languages

While the use of parallel computers has been increasing, their popularity has been hampered by the level of effort required to develop and implement the needed software. Parallel software often must be tuned to a target architecture to execute efficiently. Thus, it often requires costly redesign when ported to new machines. Different categories of parallel architectures have led to a proliferation of dialects of standard computer languages. Varying parallel programming statements for different language dialects limit parallel software portability.

Parallel computation can be viewed as an interwoven description of operations that are applied to data values, and of data movement and synchronization that dictate the form of data accesses and computation order. The traditional programming languages, like Fortran, C, or C++, provide for description of data movements and synchronization through ad hoc architecture-dependent extensions. Examples are various synchronization constructs such as busy-wait, locks or barriers used in programs for shared-memory machines, send and receive with different semantics employed by programs for message-passing architectures, and dimension projection and data broadcast popular in programs for SIMD computers.

To counter this trend to proliferation of language constructs and variants, there has recently been a strong push towards standardization of programming models and languages. Examples are the High Performance Fortran (HPF) language, the Parallel Virtual Machine

(PVM) communication primitives library, and the Message Passing Interface MPI standard. There is also a trend towards an object-oriented paradigm represented by several experimental languages based on C++. Many operating systems for parallel machines are derivatives of UNIX; therefore, next to Fortran the most popular language available on parallel machines is C with extensions. However, since its introduction in the 1950s, Fortran has been the language of choice for scientific and engineering applications that have driven sales of parallel machines so far. Fortran compilers are available on virtually all computers ranging from personal computers to workstations to parallel computers. The newest version of Fortran that was designed as a standard for parallel processing is discussed below.

4.1 High Performance Fortran

Fortran has evolved over the period of its existence by incorporating such features as array operators, dynamic storage allocation, and enhanced support for modular programming. To exploit the full capabilities of modern parallel architectures, the programmer must be able to define additional features of the programs, such as [7]:

- data mapping among processors,
- placement of data within a single processor,
- specification of control parallelism,
- specification of parallel sections of code.

The Fortran extensions that enable the user to provide this kind of information in the source program are called High Performance Fortran (HPF). They were developed by a group of users between 1991-1993 [7]. HPF is intended as a platform for portable parallel programming. It is widely assumed that major vendors of parallel computers and third-party compiler and system software developers for parallel processing will adopt HPF.

HPF includes features for mapping data to parallel processors, specifying data parallel operations and interfacing HPF programs to/from libraries and other languages. HPF uses compiler directives if the extension cannot change the program semantics and explicit language extensions otherwise. The parallelism in an HPF program can be expressed by array operators, FORALL and DO INDEPENDENT loops and EXTRINSIC and library procedures. Since communication in a program is an overhead that lowers the parallel execution efficiency, HPF puts much of the burden of defining communication on the compiler. The user supplies very high-level data mapping strategies and the compiler generates the needed communication.

4.2 Message Passing Interface

MPI is intended to be a standard message-passing interface for applications running on MIMD distributed-memory computers and workstation networks. The design of MPI has been a collective effort involving researchers in the United States and Europe from many organizations and institutions. MPI supports point-to-point and collective communication routines. It provides constructs for defining process groups, communication contexts and

application topologies. Since MPI was introduced in November of 1993, it is difficult to measure its impact on portability of programs written with its use. It is hoped by its authors [10] that it will be useful in building libraries of mathematical software for MIMD machines. MPI's design allows heterogeneous implementations and definitions of virtual communication channels. The design was also influenced by the need for ensuring that it could be implemented efficiently in a multi-threaded environment.

5 Limitation of Parallel Processing

System performance defines the computational problem sizes that can be handled within acceptable time and cost limitation. Performance is impacted by the four factors that may be of varying importance in different applications. The first is the raw computational speed (processor and memory clock time, number of arithmetic operations per second). The second is the memory of the machine (loading/unloading the data to/from the disk increases the program execution time). The third factor is the rate of input/output operations, i.e., the rate at which data can be loaded into and produced by the machine. The fourth factor is the synchronization and communication delay. It is relevant only for parallel computers. The synchronization delay arises when a processor idles because it waits for other processors to finish the corresponding stage of computation. The communication delay results from a processor's wait for receiving the requested data. In both cases part of the computational power of the machine is lost.

A parallel machine with p processors, each with speed of m megaflops, can theoretically achieve the peak performance of $p*m$. However, rarely can the algorithm be divided equally among processors. There are usually parts of it that can be done one step at a time. The so-called Amdahl law defines the limit on the speedup of a parallel execution due to the residual sequentiality of the program [1]. Speedup for a p-processor system over a uniprocessor system is defined as

$$S_p = \frac{T_1}{T_p}$$

where T_1 is the execution time for the best serial algorithm on a single processor and T_p is the execution time for the parallel algorithm using p processors. Let α be the so-called Amdahl's fraction, i.e., the ratio of the execution time spent in sequential parts of the algorithm to the total execution time on a single processor. Then,

$$S_p = \frac{1}{\alpha + (1-\alpha)/p} < \frac{1}{\alpha} \qquad (1)$$

Amdahl's formula (1) suggests that no matter how many processors are available to participate in the computation, the speedup is limited by $1/\alpha$. For example, if 5% of an algorithm cannot be parallelized, the maximum speedup will not exceed 20, no matter how many processors are used.

In most engineering and scientific algorithms, the fraction α is not a constant but a decreasing function of the problem size n. Algorithms for which $\alpha(n)$ asymptotically reaches 0 while n increases are called *effective parallel algorithms*. Such algorithms, if applied to large enough problems, are capable of achieving speedup nearly equal to the number of used

processors. To substantiate this point, let $S < p$ denote the desired speedup, arbitrarily close to p. Since the achieved speedup S_p is given by formula (1), then

$$S_p = \frac{1}{\alpha(n) + (1 - \alpha(n))/p} \geq S$$

Hence, it follows that

$$\alpha(n) \leq \frac{p - S}{S(p - 1)} \tag{2}$$

For an effective parallel algorithm $\alpha(n)$ is asymptotically decreasing to 0. Hence, it is always possible to select such a large problem size n_0 that for problems larger than n_0, $\alpha(n)$ satisfies inequality (2). Therefore any problem with size $n > n_0$ will achieve speedup greater or equal to S.

In a message-passing system, a significant fraction of the total execution time is often spent on communication between processors. To examine the effect of communication overhead on the speedup in such systems, let c denote the fraction of the total execution time spent on communication that is not overlapped with computation. If t is the sequential execution time and α is Amdahl's fraction then, with p processors, the total execution time (which includes the time spent on communication) is

$$\frac{\alpha t}{1 - c} + \frac{(1 - \alpha)t}{p(1 - c)}$$

Thus, the speedup in this case is

$$\frac{1 - c}{\alpha + \frac{1-\alpha}{p}} = \frac{p(1 - c)}{1 + \alpha(p - 1)}$$

Since $p \geq 1$, the speedup is limited by

$$S_p \leq p(1 - c) \tag{3}$$

As in the case of Amdahl's fraction, the communication fraction of the execution time c is often a function of the problem size, say $c(n)$. A parallel algorithm is *communication effective* if $c(n)$ asymptotically reaches 0 with the growth of n. The conclusion is that for large applications using effective parallel algorithms the speedup can be very close to the number of the used processors despite the communication and synchronization delays.

Large applications running on a single processor can exceed the memory and cache limit of the machine. The resulting excessive paging or cache miss ratio lead to the poor performance of such application. On a parallel machine, each processor runs only a fragment of the application. Hence, the cache and memory of the processor might be sufficient to achieve low paging and cache miss ratio. As a result, the application can achieve super-linear speedup on a parallel computer, meaning that the sum of execution times on all parallel processors is smaller than the total execution time on a single processor, i.e.,

$$S_p = \frac{T_p}{T_1} > p$$

Such speedups have been reported for large irregular computations [9].

6 Industrial Applications

Industrial applications of parallel computations are limited mainly by the relatively high cost of solving computationally intensive problems. The use of a parallel computer must be justified by the economical significance of the results. As the performance-to-price ratio and reliability of new generations of parallel computers increase, the range of applications will follow. A large part of the cost of parallel processing results from the high cost of program development. The new standardization efforts described in the previous section have a potential of fostering software portability and reuse, thus further contributing to the decline of the cost of parallel computing. The next few years most likely will witness widespread commercialization of parallel computers. Today, the range of applications is already impressive and there is a clear trend towards an increased involvement of industry in parallel processing, as evidenced by Table 1 [2]. In 1992, the worldwide installed parallel computer

Year	Government	Academia	Industry
early 1980s	70	5	25
late 1980s	60	15	25
1993	40	20	40

Table 1: The Percentage of Parallel Computer Installations for Different Users Categories

base (of U.S. vendors only) was nearly equal between Academia and Government (129 in total) and Industry (122). In academic centers, the usage of parallel computers by industrial users nearly doubled between 1991 and 1992 (last two years for which data is available).

The cooperation between academic centers and industry is strong in the United States. For example, the Scientific Computing Research Center at Rensselaer Polytechnic Institute brings together 35 faculty, 50 graduate students and many researchers from 16 organizations including such industrial leaders as Alcoa Technical Center, Dassault Systems, General Electric Company, General Motors Corporation, Grumman Aircraft, and IBM Corporation. The spectrum of investigated problems covers computational fluid dynamics, engineering structural analysis, human joints dynamics, and epidemiological modeling.

Some of the largest customers of parallel computers are commercial aerospace companies. They have been using computational fluid dynamics to analyze airflows for spacecraft and planes. An interesting application of this method was made by Boeing to predict airflow in aircraft cabins using a Cray parallel computer. The design of an airplane's environmental control system involves the specification of air supply and return and analysis of airflow speed and distribution. The computer simulation eliminates the majority of candidates and the full-size airplane cabin mockups are used only in the final selection. The expense and the time required for testing the large number of candidate airflow systems has been thus largely reduced. Computational fluid dynamics is also useful in the analysis of airflow for cars. For example, Nissan Motor Company reported that it saves on wind tunnel tests by using a model of an unsteady, three-dimensional viscous incompressible flow program on the Cray C90 computer achieving the performance of 7 gigaflops.

Oil reservoir modeling uses cross-well seismic data to build and run the model. Even with current parallel supercomputers, large models use a computational unit of several-hundred feet which may contain few separate wells. However, there are important fluid events at the scale of a foot, such as the mixing of elementary fluids. Companies like Mobil Corporation and Amoco Oil Company use parallel computers for exploiting existing as well as searching for new reservoirs. Shell Oil Company reported that it built parallel versions of several petroleum reservoir simulators, but those have not been put into use because of the difficulty in providing requisite network and job support. In contrast, selected geophysical application programs have been used successfully in corporate settings. In [4], British Petroleum Exploration Inc. reported the accurate modeling of a complex reservoir to predict potential gas and oil production, the rate of production and the impact of operating decisions on recovery and economics. When operating under constrained computer resources, the model of a reservoir must be simplified and projections conservative. More reliable projections obtained with the use of the parallel computer increased predicted recovery and reduced the time and cost of the study. The economic benefits far outweighed the cost of using the parallel computer.

Challenged by the international market demand and increasingly complex production requirements, a growing number of heavy industries worldwide are exploring parallel processing usage to optimize manufacturing processes. A division of the German industrial conglomerate Mannesmann used the Cray parallel computer for optimizing pipe and tube milling [11]. Structural and civil engineering problems are solved at Mitsui Construction Company in Japan. Ford Motor Company purchased the Cray Y-MP C90 system for structural analysis, crash simulation and other problems related to automotive design and engineering. A smaller system, the Cray Y-MP 4E, was also installed in the PSA Peugeot-Citroen in Velizy, France for similar applications.

Computational chemistry uses parallel computers to study problems such as the prediction of relative stability of different molecules, the identification of transition states, and reaction intermediates based on the model of heat formation. A commercial application also includes an analysis of the effect of a molecular structure on the flexibility of polymers [6]

Designing, installing and operating a power transmission network and ensuring its stability and reliability are complex challenges. Each network is a dynamic system subjected to oscillations which can lead to costly equipment failures, network separation and eventually blackouts. Events such as lightning bolts, ice storms and tornadoes disturb and threaten to disrupt power network operation. The growing demand for electrical power and the complexity of interconnected, expanded networks require prudent operational planning and the ability to predict a power system's behavior under various conditions. Hydro-Quebec in Canada uses the Cray computer for testing and predicting the network's operation under the various contingencies to decide the proper improvements for the network structure.

Weather forecasting uses grids of the size of two-hundred miles by three-hundred miles, too large to register local rains, storms, etc. To reliably model storms, the Center for Analysis and Prediction of Storms at Oklahoma University developed the ARSP (Advanced Regional Prediction System). ARSP has been ported to the massively parallel computer CM-5 with 1024 processors [12]. A high speedup of 907 times over a single processor was achieved with the overall performance of about 50 megaflops. Still, for the problem size corresponding to a regional weather prediction, the simulation runs about one-fifth as fast as the weather

changes (so after one hour of simulation, the prediction could be made for four hours in advance). It is estimated that a teraflops machine would be able to produce a four-hour forecast in about 2.5 minutes.

There is a growing trend among Wall Street securities firms to utilize the advanced computer simulations to track and model global financial markets. Among them, Prudential Securities is a pioneer of parallel computing, currently using the Intel 32-node hypercube computer IPSC/360. Dow-Jones News Retrieval acquired two Connection Machines from Thinking Machines Corporation to improve the performance of their commercial document retrieval systems.

Parallel computing and related technologies of computer networks, database management systems and graphics are changing the scale and scope of data that companies and governments can manage and analyze. This process involves not only computer expertise but also finance, marketing and management. Parallel computing helps organizations produce information in three major domains:

1. changes in production, with greater emphasis on managing the data as a strategic resource.

2. improved control over relationships with customers and clients.

3. development of new kinds of information.

In recognition of this trend, ORACLE has been making its database system available to a growing number of parallel machines. The impact of paraliel computers on this kind of applications should rapidly grow.

References

[1] Amdahl, G.M., "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. 30 AFIPS Conference*, AFIPS Press, pp. 483-485, 1967.

[2] Branscomb, L. (Chair), *From Desktop to Teraflop: Exploiting the U.S. Lead in High Performance Computing*, NSF Blue Ribbon Panel on High Performance Computing, Washington, DC, August, 1993.

[3] Committee on Physical, Mathematical and Engineering Sciences. *Grand Challenges: High Performance Computing and Communications*, NSF/CISE Report, Washington, DC, 1991.

[4] Cullahm, W.E., Deskin, R.H, Handyside, D.D., Karaoguz, O.K., and Li, K.-M., "Improved Financial and Operational Forecasting with Large-Scale Reservoir Models," *Cray Channels*, vol. 14, no. 3, pp. 26-31, 1992.

[5] Flynn, M., "Some Computer Organizations and Their Effectiveness," *IEEE Transactions on Computers*, vol. C-21, pp. 948-960, 1972.

[6] Graffunder, S.K., "Barrier-Breaking Performance for Industrial Problems on the Cray C916," *Proc. Supercomputing'93*, Portland, OR, November, 1993, IEEE Computer Science Press, Los Alamitos, CA, pp. 516-519.

[7] High Performance Fortran Forum, *High Performance Fortran Language Specification, version 1.0*, Center for Research on Parallel Computation, Rice University, Houston, TX. Revised May, 1993, to appear in *Scientific Programming*, vol. 2, no. 1, 1994.

[8] Hwang, K., *Advanced Computer Architectures*, McGraw Hill, New York, 1993.

[9] Lewis, T.G., and El-Rawini H., *Introduction to Parallel Computing*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[10] The MPI Forum, "MPI: A Message Passing Interface," *Proc. Supercomputing'93*, Portland, OR, November, 1993., IEEE Computer Science Press, Los Alamitos, CA, pp. 878–883.

[11] Pehle, H.J., and Thieven, P., "Advances in Metal Forming Simulation at Mannesmann," *Cray Channels*, vol. 14, no. 1, pp. 6–9, 1992.

[12] Sabot, G., Wholey, J.B., and Oppenheimer, P., "Parallel Execution of a Fortran 77 Weather Prediction Model," *Proc. Supercomputing'93*, Portland, OR, November, 1993, IEEE Computer Science Press, Los Alamitos, CA, pp. 538–545.