Hardware: Developing the Metacomputer

Engineers will have easy, transparent, desktop access to a variety of powerful computing resources

Mark A. Stadtherr, Haruna N. Cofer, and Kyle V. Camarda, University of Illinois

Since the mid-1980s, computational power has increased by factors approaching two orders of magnitude for all types of hardware, from desktop personal computers to state-of-the-art supercomputers. And this trend of dramatically increasing performance, which actually dates back decades *(1,2)*, should continue into the next century. So, chemical engineers in 2001 should have markedly mightier machines at their command. To provide a sense of what is likely to happen, we will look at the expected growth in computational power, the technologies underlying it, the techniques required to exploit it, and its impact in chemical engineering applications.

Today, for scientific and engineering computing, the marketplace can be divided roughly into three overlapping categories: personal computers, workstations, and high-performance computers. As used here, personal computer refers to single-user desk-based systems typified by the Apple Macintosh, IBM PC, and various PC compatibles. Workstation covers a broad category of faster desk-based machines, which have multiuser capability but which often are dedicated to a single user — machines like Sun Sparcstations, IBM RS/6000s, and HP 9000s. And the high-performance-computer category encompasses machines that employ some form of advanced computing architecture such as vector processing or parallel processing — these may be state-of-the-art supercomputers like the CRAY C90, or Thinking Machines CM-5; multiprocessing workstations such as the Silicon Graphics VGX; or multiprocessing personal computers like the Compaq Systempro.

As their needs demand, engineers and scientists now may use machines in any or all of these categories. And, because advances in networking to provide inter-machine communication *(3)* have paralleled gains in computer speed, access to these different sources of computational power often is available right from the users' desks. Indeed, what is evolving is a network of heterogeneous computational resources from which engineers can draw needed processing power and other resources. Current networks of this sort represent a rapidly developing form of computational system that has become known as a *metacomputer*.

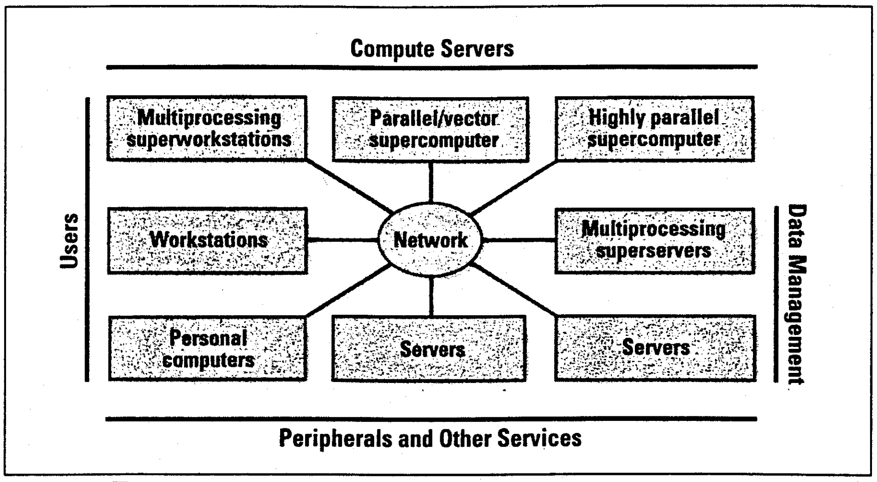
The metacomputer

This is defined as a network of heterogeneous computational resources linked by software so that is it virtually transparent to the users *(4,5)*. In other words, users ideally will be unaware that they are using any computer beyond those on their desks — even though they may, in fact, be using computational power and other resources coming from various parts of the metacomputer system, which may well be working in parallel on different parts of a problem.

To make an analogy, consider the national electrical power network. When you plug your desktop computer into an electrical outlet, you actually are plugging it into a network over which electrical power is distributed. You do not know or care whether the electricity you are using was generated at a nearby power plant or one several hundred miles away. Nor do you know, though perhaps you may care, whether fossil fuels, nuclear fission, or solar energy were used in generating that power. Similarly, a desktop computer could be plugged into a nationwide (or company-wide) metacomputer that provides computational power, as well as data access and storage, and other resources, in such a way that you do not know, or perhaps care, the actual source of that power. In an effort to minimize the elapsed wall-clock time, software may direct various parts of a problem to the appropriate source of computational power based on availability and the type of computation involved. Efforts are underway at various commercial and academic enterprises, including the National Science Foundation supercomputer centers, to develop the software needed to make a large-scale metacomputer operational.

The metacomputer also may be thought of as the ultimate in Client/Server Computing (CSC) systems *(6)*. Client programs, typically running on a user's personal computer or workstation, request services from server programs running on other machines. These servers may, in turn, become clients of other servers. Layers of software between the clients and servers cause the different processors to cooperate without the intervention of the user. Servers may provide several services, or a service may be provided by several servers. Today, common services include file storage, databases, peripheral sharing, and mail. The computational server, a key feature of the metacomputer, is largely lacking, though. This is due in part to the lack of a standardized way to query a compute server, as there is, for example, with Structured Query Language (SQL) and relational database servers.

The metacomputer is evolving from today's CSC systems. One possible metacomputer configuration that affords powerful computational and data-management resources is shown schematically in Figure 1. Users interact primarily with desk-based personal computers or workstations. Numerically intensive computations that cannot be done in a timely manner by the desk-based machines are handled automatically by one or more high-performance compute servers, depending on the nature of the computation. Large files and information retrievals from big databases are dealt with by multiprocessing superservers, and a wide variety of other functions are performed by other servers. Later, we will discuss in more detail a possible metacomputer system for a chemical engineering environment.


Compute Servers
Multiprocessing superworkstations
Parallel/vector supercomputer
Highly parallel supercomputer
Users
Workstations
Network
Multiprocessing superservers
Data Management
Personal computers
Servers
Servers
Peripherals and Other Services

■ *Figure 1. Schematic of a possible metacomputer configuration.*

In this article, we will primarily focus on the parts of the metacomputer that provide floating-point computational power and, to a lesser extent, on the superservers that provide data-management services. We will look at these in the context of our three overlapping machine categories.

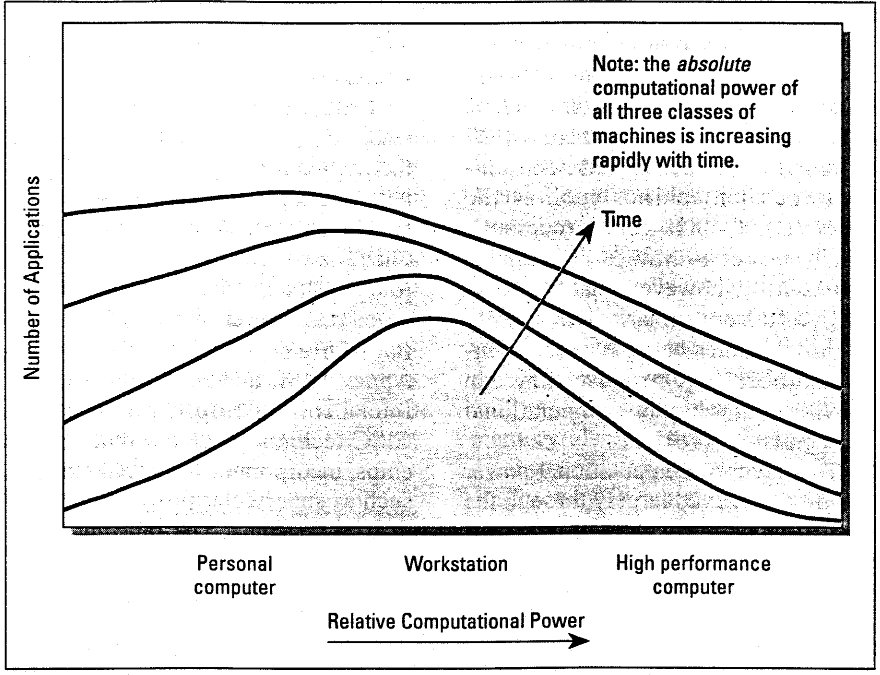
Personal computers

An increasing number of chemical engineers now are finding that their computational needs can be satisfied simply by using a personal computer. This is not surprising because today's personal computers provide power comparable to, if not exceeding, most of the mainframes and minicomputers widely used in the past (and still soldiering on in some installations). To put this into perspective, we'll employ the widely used LINPACK-100 benchmark *(7)*, which indicates the performance of a machine in solving a dense system of linear equations of order 100 using the LINPACK software. Though there are many other benchmarks used, this one has been run on a very wide variety of machines, and represents performance on a problem common in scientific and engineering applications. On this benchmark, personal computers such as the Gateway 2000/486DX2/66 and Apple Macintosh Quadra 950 perform at 2.4 and 2.0 megaflops (millions of floating point operations per second), respectively — as compared to older mainframes and minis such as the CDC Cyber 175 at 2.1 megaflops (Mflops), IBM 4381-23 at 1.2 Mflops, and DEC VAX 8700 at 0.99 Mflops. Older personal computers, like the IBM PC/AT (with 80287 math coprocessor) at 0.012 Mflops, perform two orders of magnitude slower, and even relatively recent machines, such as the Compaq 386/20 (with 80387 math coprocessor) at 0.13 Mflops, are one order of magnitude slower.

This increase in the computational power of personal computers is due not only to faster clock speeds, but to improvements in cache memory and intramachine communications. Such advances will continue to occur. Machines now are beginning to appear based on the Intel Pentium (P5) microprocessor, the latest extension of the widely used 80x86 architecture. Running initially at 60 or 66 MHz, the Pentium chip is twice as fast as a 486DX2/66, and breaks through the 100 million instructions per second (MIPS) barrier into territory previously reserved to workstations. An improvement of another factor of two or so can be expected, as clock speeds of 100 MHz and beyond are anticipated. The Pentium offers even greater performance increases in the speed of floating-point operations, providing roughly four to five times the speed of the fastest 486. This is done through the use of dedicated, as opposed to general-purpose, arithmetic units, and rudimentary on-chip parallelism. For example, the dedicated multiplication unit gives a result in three clock cycles as compared to 12 to 15 for the 486. The high-end personal computer clearly is beginning to overlap the traditional workstation market and, conversely, low-end workstations are invading the traditional personal computer market, as discussed in more detail below.

Today, engineers can solve on personal computers many problems that not long ago would have required a much larger system. Such problems include flowsheet simulations, statistical analyses, on-line data acquisition and control, process scheduling and planning, parameter estimation, optimization, and various mathematical-modeling applications. These problems might be solved using mainframe-oriented software packages that have migrated down to the personal computer level, or by using personal-computer-oriented software such as spreadsheets. In many cases, the personal computers require only about the same amount of central processor unit (CPU) time as the larger computer systems, and often provide turnarounds in less actual wall-clock time when compared to a busy timesharing system.

Thus, the major impact of personal computers has not been to expand the types of problems that can be solved, but rather to make the necessary level of computational power much more widely and cheaply available, and in an environment rich in easy-to-use post-processing tools such as word processing, graphics, and communications.



■ ***Figure 2. Distribution of chemical engineering applications based on relative computational power required.***

Many of the chemical engineering problems solved today on personal computers represent computationally diverse applications — that is, applications for which computational requirements vary widely depending on problem size and complexity, and for which more computational power always can be used. For example, in flowsheet simulation, we can solve simple steady-state problems in seconds on a personal computer, yet complex dynamic simulations may require several hours or days on even the fastest supercomputer. In such applications, personal computers are capable of solving an ever-expanding scope of problems on the low end, high-performance computers are needed on the high end and are pushing forward the range of problems that can be attacked, while workstations are used to handle the large bulk of problems in the middle. This profile of problem distribution and its possible evolution with time is depicted schematically in Figure 2. Such an evolving profile may be considered characteristic of chemical engineering computing as a whole.

Workstations

For many engineers, workstations have become their computational workhorses. (Remember that we are distinguishing workstations from personal computers by their multiuser timesharing capabilities made possible through the use of operating systems such as UNIX or VMS. Workstations also generally are capable of providing for substantially more central (random access) and auxiliary (disk) memory than personal computers.) Low-end workstations provide a level of computational power comparable to high-end personal computers, while high-end workstations often are an order of magnitude faster. For example, on the LINPACK-100 benchmark, an HP 9000/735 provides 41 Mflops and an IBM RS/6000-980 runs at 38 Mflops. This is two orders of magnitude faster than older, widely used workstations such as the SUN-3/260, which runs at 0.46 Mflops (with floating point accelerator) on this benchmark. High-end workstations now can provide computational speed comparable to older vector supercomputers, such as the CRAY 1S, which runs at 27 Mflops on this benchmark, and current vector machines such as the CONVEX C-3810 (one processor), which runs at 44 Mflops. It is important to note, however, that the LINPACK-100 benchmark is not really fair to machines based on vector/parallel processing, because they can provide much higher computational rates on larger problems.

This level of computational power has been achieved largely through the use of Reduced Instruction Set Computing (RISC) technology *(8,9)*, as opposed to Complex Instruction Set Computing (CISC). The principles underlying RISC were developed beginning in the mid-1970s, and today are implemented in a number of designs, the most widely used of which is the SPARC design used by Sun Microsystems. The basic idea in RISC is to use a small instruction set consisting of simple, specific, and usually fixed-length instructions executable in a very small number of clock cycles. Several RISC instructions may be required to perform a function that can be done with a single CISC instruction. The CISC instruction, however, must be designed to handle all possible cases of a function and, thus, on many specific cases may take many more machine cycles than the required sequence of RISC instructions. In other words, RISC lays open the underlying intrinsic machine functions, permitting optimizing compilers to generate the best code for specific cases, whereas in CISC the general case must always be implemented. The use of RISC facilitates so-called superscalar processing, a term generally implying a high degree of pipelining — in which, like an assembly line, several functions may be operating simultaneously on tasks which are in different stages of completion — and a degree of on-chip parallel processing, such as the use of parallel integer and floating-point pipelines, and parallel load/store units.

Today, machines based on RISC technology are beginning to appear in the high-end personal computer market. For example, DEC's AXP personal computer is based on its Alpha chip, and the next generation Macintosh and other machines will be based on a PowerPC chip *(10)*, developed in a joint venture involving Apple, IBM, and Motorola. And the Intel Pentium chip, while based on CISC technology as are other 80x86 chips, incorporates advanced features, such as superscalar processing, found in RISC designs. The blurring of the high-end personal computer and low-end workstation markets is being driven in part by the availability of new operating systems, such as Windows NT, that can run on either CISC or RISC machines, and can handle software developed for either. New graphical-user-interface (GUI) based versions of UNIX *(11)*, such as Solaris 2.2, that will run on 80x86 machines, will reinforce this trend, and may lessen the resistance to UNIX in the personal computer market that developed with earlier 80x86 versions of UNIX.

Two new versions (P6 and P7) of 80x86 CISC technology are expected to follow the Pentium chip this decade, and improvements in RISC technology will proceed as well. Thus, we can expect a continuing battle between RISC and CISC machines, as well as between operating systems that can run on either platform. Future improvements in both RISC and CISC processors will be due to a large extent to extensions of superscalar capabilities and extensive use of on-chip parallelism. Today, advances in workstation performance also are increasingly coming from the use of multiple processors, both for numerically intensive computing and for large server applications such as database management. These advances are discussed in more detail below.

Chemical engineers are using workstations to solve a very wide variety of problems. Among these are process scheduling and planning based on sophisticated optimization techniques such as mixed integer linear and nonlinear programming; mathematical modeling problems based on finite differences, finite elements, and other advanced numerical methods; and many other of the computationally diverse problems discussed above. In some cases, these are problems that not long ago would have had to be solved on a supercomputer. So, the impact of developments in workstations is to put much more computing power in the hands of many more users. Thus, more interesting and important problems can be solved and in a more timely fashion.

There, however, remain chemical engineering problems of sufficient size and complexity for which the turnaround on workstations may be too slow, or which simply are intractable on current workstations. So, there is a need for high-performance computing, typically involving some form of parallel processing.

High-performance computing

Today, parallel processing is rapidly entering the mainstream of computer technology. Though single processor performance will continue to improve, the most immediate way to improve a system's performance is through the use of multiple processors, as opposed to waiting for the next generation of single processors. Furthermore, physical limits on single processor speeds will eventually be reached. Because, in principle, parallel processing has no upper limit in speed, it represents the inevitable future of computing, not only for the fastest state-of-the-art supercomputers, but also for desk-based machines and servers. In fact, it is the booming network- and database-server market that is driving parallel processing into the mainstream today. As this booming market forces prices down, inexpensive parallel computing hardware increasingly will show up as a computational tool on engineers' desks, while more powerful parallel machines will be incorporated as computational servers in the evolving metacomputer.

The basic ideas involved in parallel computing are described in a number of sources (for instance, Refs. 12–15), and for more detail a number a good texts are available (such as Refs. 16–18). Most machines available today can be categorized as either Single Instruction/Multiple Data (SIMD) or Multiple Instruction/Multiple Data (MIMD), with the latter predominating. SIMD machines are capable of executing the same instruction simultaneously on multiple data values. The SIMD architecture is generally found in relatively-special-purpose array processors, such as the Connection Machine CM-2. MIMD machines can execute different instructions simultaneously on different data streams, and serve in more general-purpose machines.

Most MIMD computers can be classified either as distributed-memory or shared-memory. In a distributed-memory machine, each processor has its own local memory and cannot address the local memory of other processors. Thus, results from one processor needed by another must be passed as messages through some interconnection network. These machines also are known as multicomputers, because each processor and its local memory can essentially be thought of a complete computer in its own right. Clusters of workstations connected in a local area network represent one extreme of the distributed-memory MIMD approach. This clustered workstation approach is best suited to applications that can be almost completely decomposed into independent parts whose processing requires little or no communication with other processors. For applications involving more message passing, the considerably faster communication that can be achieved by locating the processors in a single cabinet is important. Distributed-memory systems have the advantage of being scalable to a very large number of processors. They, however, have a reputation of being difficult to program, in part because of the difficulties in partitioning data among the local memories, and moving information to and from them. Furthermore, for many applications, performance does not scale with the number of processors; so the ability to accommodate a large number of processors is not always important. Thus, the shared-memory approach is the most widely used today.

Applications best suited to MPPs now are those "embarrassingly" parallel applications.

In shared-memory machines, all processors address a common memory, and thus communicate indirectly by reading from and writing to memory. When many processors share the same memory, contention for memory is a problem that must be addressed. State-of-the-art supercomputers such as the CRAY C90 use extremely high memory bandwidths (over 250 gigabytes/s total) to achieve a uniform memory access time over 16 processors. It is not often appreciated that the largest part of the cost of such machines is attributable to such fast uniform memory. Memory contention can also be addressed by giving each processor its own local memory cache, which essentially holds data until the shared memory is free from contention. This raises the problem that a variable in the shared memory may simultaneously have different values in different caches. There are various schemes *(19)* for handling this cache coherency problem. Cache-based systems are simple examples of nonuniform memory access systems, which in general may have several levels of physically distributed (though still shared) memory interconnected to the processors in various ways. Increasingly, machines are appearing that provide both the scalability of the distributed-memory model and the ease of programming of the shared-memory model — by combining hardware and software features of each. For example, the CRAY T3D is scalable to 2,048 DEC Alpha RISC microprocessors, each with its own local memory, but this memory also is globally addressable by any processor.

The way in which processors are interconnected in distributed-memory or nonuniform access, shared-memory systems is important, because it affects the amount of communications delay (latency) in accessing data from memory or from other processors. In general, there are competing goals in the design of interconnection networks. Typical goals are to provide a network in which the path from any one processor to any other is always relatively short, the total number of interconnections is relatively small, and the number of interconnections per processor remains constant as the number of processors scales upwards. A number of interconnection schemes are diagrammed in the literature *(12,13,15)*. Popular schemes today include hypercubes, meshes, and toroidal meshes, as well as switching networks based on crossbars, trees, or other multistage switches.

There are basically three types of parallel computing hardware that predominate commercially today: scalar superserver (superworkstation), parallel/vector supercomputer, and highly parallel. So, let's look at the typical features of each, and their areas of application in chemical engineering.

Scalar superservers

These machines are generally MIMD, shared-memory, and cache-based. They rely significantly on virtual memory, and involve a relatively small number of CISC or RISC microprocessors. For instance, the Sun SPARCcenter 2000 may be configured with 2 to 20 processors. Machines of this type with more processors are appearing; for example, Cray Research is expected to unveil a new SPARC-based superserver that will have a substantially larger number of processors than the current 8-processor CRAY S-MP superserver. A large commercial market is emerging *(20)* for such machines as servers in client/server networks. These servers can perform different services in parallel; they also can use multiple processors to perform services, like database management and transaction processing, that are frequently parallelizable easily.

The need to efficiently manage large amounts of process data is a significant problem in chemical engineering. Today, process data often are not readily usable, and it may be days before reports on a day's operation are available. There is also an increasing demand for process data to document compliance with environmental, safety, and quality-assurance standards. Thus, superservers may find significant use in the process industries for data-management applications. Chemical engineers also will employ these machines as computational tools, as an inexpensive way to extend the range of problems that can be solved using workstations. Today, though, tools such as compilers for parallelizing applications on such superworkstations are generally not as well developed as those for parallel/vector supercomputers — but such tools can be expected to migrate down to this level.

Parallel/vector supercomputers

Such supercomputers generally can be characterized as MIMD, shared-memory machines with uniform access to large amounts of real memory. They involve a relatively small number of processors, typically 4 to 16, each of which is a powerful vector processor. The number of processors in these machines is expected to reach 64 by the year 2001. Each vector processor is usually highly pipelined, and has the ability to perform a significant number of arithmetic and load/store functions in parallel. Examples of such parallel/vector machines include the CRAY Y-MP, NEC SX-3, and CONVEX C-3800 families of machines. While not too many years ago machines of this sort were still regarded as fairly special purpose, today they have emerged as general-purpose computational workhorses, and are applied in solving a wide variety of large-scale scientific and engineering problems. This is due in part to the development of good programming tools, such as vectorizing and parallelizing compilers, and the extensive number (over 1,000) of application codes now available. Furthermore, while state-of-the-art supercomputers still cost over $10 million, entry-level supercomputers such as the desk-side CRAY EL92 cost two orders of magnitude less, and are providing wider and wider access to the power of this technology.

While hardware capabilities have been advancing rapidly, the software often lags behind.

There are many chemical engineering applications that utilize the power of these machines. From a fundamental standpoint, chemical engineers frequently today are interested in modeling complex processes involving interacting transport phenomena, often with simultaneous chemical reaction. These include problems in reactor design, combustion, mixing, and separations. Using vector/parallel supercomputers, many such problems today can be solved with high resolution using realistic three-dimensional models and in a reasonable time frame. Also, chemical engineers increasingly are interested in modeling processes and materials on a molecular or microscopic level, using the techniques of computational chemistry and computational materials science. At many installations, these problems represent the largest application of vector/parallel supercomputing, and are leading to the design of new pharmaceuticals, pesticides, catalysts, polymers, and other products.

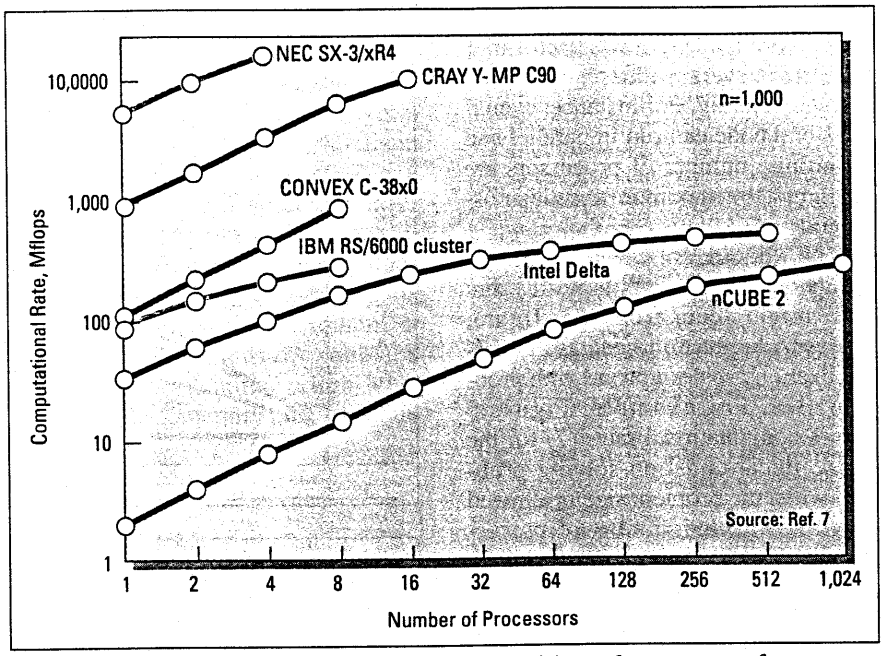
From a process engineering standpoint, there also are a number of applications of vector/parallel supercomputers — though these generally are in an earlier stage of development than those discussed above. Among such applications are dynamic flowsheet simulation, real-time plantwide optimization and control, process synthesis, and process scheduling. By allowing the use of more accurate process models — for instance, more realistic treatment of trace components — these machines lead to processes that are safer, cleaner, and that produce a higher quality product. And more accurate models foster a better understanding of process fundamentals, and thus reduce the need for overdesign, which in turn results in better resource utilization and lower costs. Furthermore, the rapid turnaround that can be provided by vector/parallel machines improves engineering productivity, meaning faster times to market, more timely response to changes in business, economic, and regulatory conditions, and more important deadlines met. For instance, Helling and coworkers *(21)* have noted that it was the use of a CRAY supercomputer to run process simulations that enabled a critical project deadline to be met in optimizing a multiproduct chemical plant at Dow.

For many of these applications, vector/parallel supercomputing has allowed chemical engineers to break through computational barriers and to formulate and solve new problems that would not otherwise have been solvable, at least not in a reasonable time frame. And, indeed, this will continue to be a major factor in chemical engineering computing. While many problems solved today on supercomputers eventually will be solvable on workstations, the continuing impact of high-performance machines will be to push forward problem-solving frontiers.

Highly parallel machines

Today, these machines are generally MIMD, though they may be able to emulate SIMD operation. They typically have large real memories that are physically distributed but that, in some cases, may be globally addressable; RISC microprocessor chips are used in each node, perhaps with vector-processing arithmetic accelerators. The number of processors may range from the tens to the thousands. Some examples include the nCUBE and Intel iPSC families of machines, based on hypercube configurations; the Intel Delta, based on a two-dimensional mesh; the Thinking Machines CM-5, based on a tree architecture; and the CRAY T3D, based on a three-dimensional toroidal mesh. Large machines of this type are frequently referred to today as Massively Parallel Processors (MPPs). MPPs represent a still-emerging area of technology. Programming tools and problem-solving algorithms are in relatively early stages of development, and not a large number of application codes are available.

Applications best suited to MPPs now are those "embarrassingly" parallel applications, such as signal processing and image processing, that are inherently parallel and require little interprocessor communication. Monte Carlo methods, widely used in chemical engineering for molecular-level simulations, are among such applications. Some chemical engineering applications, such as problems in computational fluid dynamics and reactor modeling that are based on finite-element or finite-difference methods, have been run successfully on MPPs — but, in general, it is much more difficult to exploit parallelism on these problems than in the inherently parallel applications. Many codes of this type today are still best suited to parallel/vector supercomputers. Some such applications will eventually migrate to MPPs as better codes and problem-solving algorithms are developed. Other applications will remain best suited to parallel/vector machines, or to parallel machines with a smaller number of processors. The difficulty, as we will discuss, is that in many applications, at least using today's numerical algorithms and codes, the number of processors that can be effectively used on a problem is limited by the sequential content of the computation.



■ *Figure 3. Computer performance in solving a dense system of linear equations of order n = 1,000.*

Performance

To gauge the speed of some high-performance computers, let's return to the problem of solving dense systems of linear equations, first looking at systems of order 1,000. Figure 3 shows some results *(7)* for a variety of high-performance machines. If full advantage was taken of parallel processing in this application, then the curves for each machine would increase linearly from the same initial slope, indicating a doubling of the computational rate each time the number of processors is doubled. It is noteworthy that in general as the number of processors increases, the efficiency with which parallelism is used decreases. This is common in many applications, and is often explained in terms of Amdahl's Law.

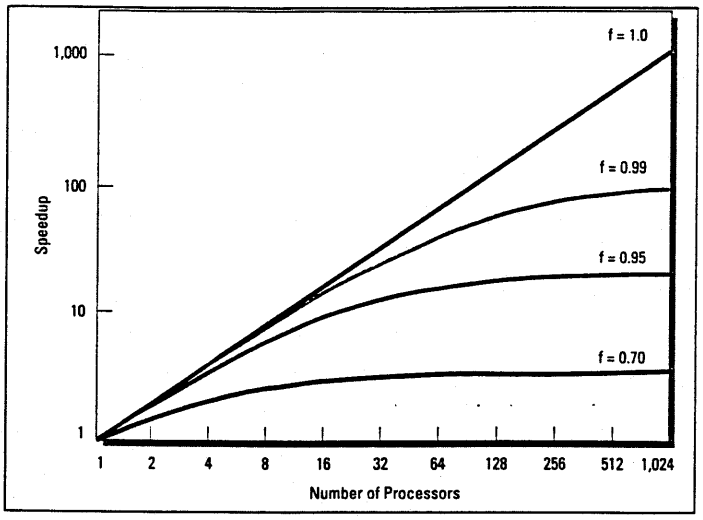
Amdahl's Law is based on a simple model of parallel computing, and essentially provides an upper bound on the speedup possible from a given percentage parallelization. It is given by:

$$S = P/[P(1-f)+f]$$

where S represents an upper bound on the speedup, P is the number of processors, and f is the fraction of utilized code performed in parallel. Speedup is defined here as the ratio of the time it takes to execute a job on one processor to the time it takes to execute the same job on P processors. The lesson of Amdahl's Law is that even with an infinite number of processors, if there is just a small amount of code that cannot be run in parallel, the potential speedup will be greatly limited. Figure 4, a plot of *S* vs. *P* for several values of *f*, shows this very clearly. For instance, even if 95% of a code can run in parallel and an infinite number of processors are available, the maximum speedup possible is only 20.

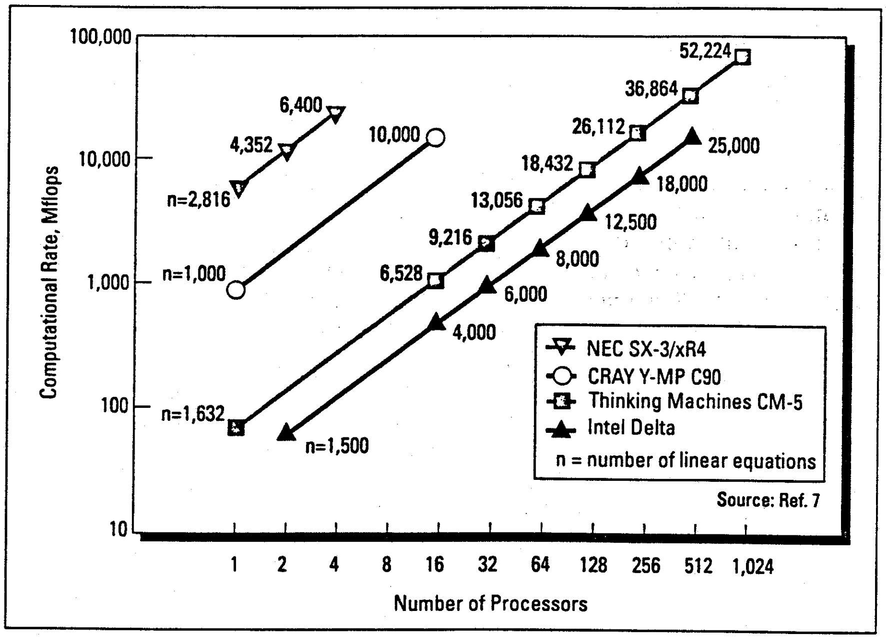
The outlook for the use of large numbers of processors is not as dim as it might seem from that figure, however, because in general the parallelization *f* scales upward with problem size. For example, Figure 5 shows computational rates *(7)* for the dense linear-equation-solving problem when the problem size is allowed to grow with the number of processors used, as is likely in most applications. Note that the shared-memory parallel/vector supercomputers achieve their high computational rates at much smaller problem sizes than the highly parallel machines. The parallelization *f* also depends strongly on many other aspects of the problem, including its formulation, the solution algorithm used, the coding of the algorithm, the compiler used, and ultimately the machine itself. Thus, there are many opportunities for improving the use of parallel computing, and all of these are active areas of research.

Today, the "holy grail" of high-performance computing is the sustainable teraflop (one trillion floating point operations per second). Theoretical peak speeds of a teraflop certainly seem within reach today, and it is anticipated that the sustainable teraflop will be achieved this decade for at least some applications. Because speeds are now at the 100 gigaflops level, this prediction is consistent with the observed two orders of magnitude increase in computational power per decade. Though it is sometimes assumed that most performance gains will be made in MPP systems, the performance of vector/parallel machines will continue to improve at about the same pace, through improvements in intraprocessor parallelism and number of processors. Teraflop performance and beyond is called for in the solution of so-called grand challenge problems — fundamental problems in science and engineering that will have broad scientific and economic impact, a selection of which are discussed in Ref. 22.



■ *Figure 4. (left) Plot of Amdahl's Law for a constant-size problem for different fractions of parallelization* **f.**

■ *Figure 5. (below) Computer performance in solving a dense system of linear equations of increasing order* **n.**

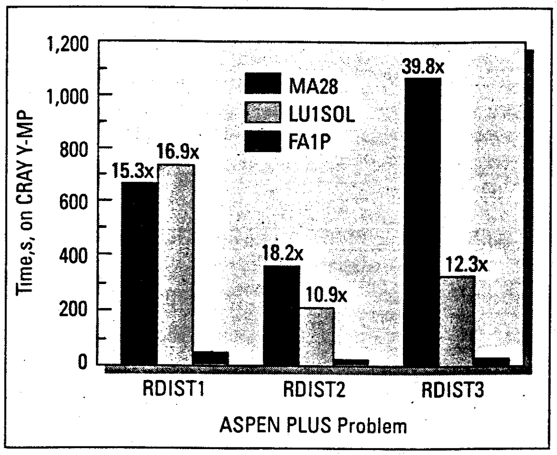


Software and solution strategies

While hardware capabilities have been advancing rapidly, the software to utilize these capabilities often lags behind. This is especially true today in the MPP area, and to a lesser extent in parallel/vector machines. New languages, such as Linda, are evolving to address the needs of parallel computing, as are parallel dialects of traditional languages such as Fortran-77. Relatively new languages, such as Fortran-90, that provide for array-valued operations facilitate the use of vectorizing and parallelizing compilers, because those operations often are easily vectorized or parallelized. Indeed, given the availability of such compilers, especially for parallel/vector machines, Fortran, a very old yet in its latest incarnation still new language, likely will remain a mainstay of scientific and engineering computing.

Of course, compilers cannot vectorize or parallelize codes that implement solution algorithms that are not amenable to vector or parallel processing in the first place. Most problem-solving strategies today evolved based on a serial-computing paradigm in which minimization of serial-operation count was a critical factor in determining whether or not a method was successful and survived. This is no longer an appropriate criterion by which to judge numerical algorithms, inasmuch as a larger operation count may be preferred if those operations can be performed in parallel. New problem-solving strategies are being developed in which parallelizability and vectorizability are the desired traits. One example of the impact of the solution algorithm is in process simulation. Zitney *(23)* has considered the simulation of reactive distillation problems using ASPEN PLUS, and looked at the effect of using different algorithms for solving the large sparse-linear-equation system that is a key step in the computation. Two of the codes used, MA28 from the Harwell subroutine library, and LU1SOL from the University of Illinois *(24)*, are based on traditional serial algorithms, which do not vectorize well. The third code, FA1P, uses a different algorithm *(25)* that does vectorize well, though involves on these problems a significantly higher operation count. Figure 6 shows the effect of using FA1P on the total simulation time required to solve three reactive distillation problems. Performance improvements of over an order of magnitude were obtained simply by using an algorithm more amenable to vectorization. Studies along the same lines *(26)* involving dynamic-process-simulation problems solved using SPEEDUP have also resulted in dramatic performance gains.

Vectorizing and parallelizing compilers are only effective in recognizing relatively low-level parallelism, usually at the DO-loop level. There, however, often are opportunities for higher-level parallelism. These usually must be recognized by the algorithm developer, typically based on knowledge of a specific problem or class of problems that cannot be imparted to a compiler. High-level parallel tasks generally will be larger



■ ***Figure 6. The sparse matrix algorithm used can markedly affect solution time for reactive distillation problems.***

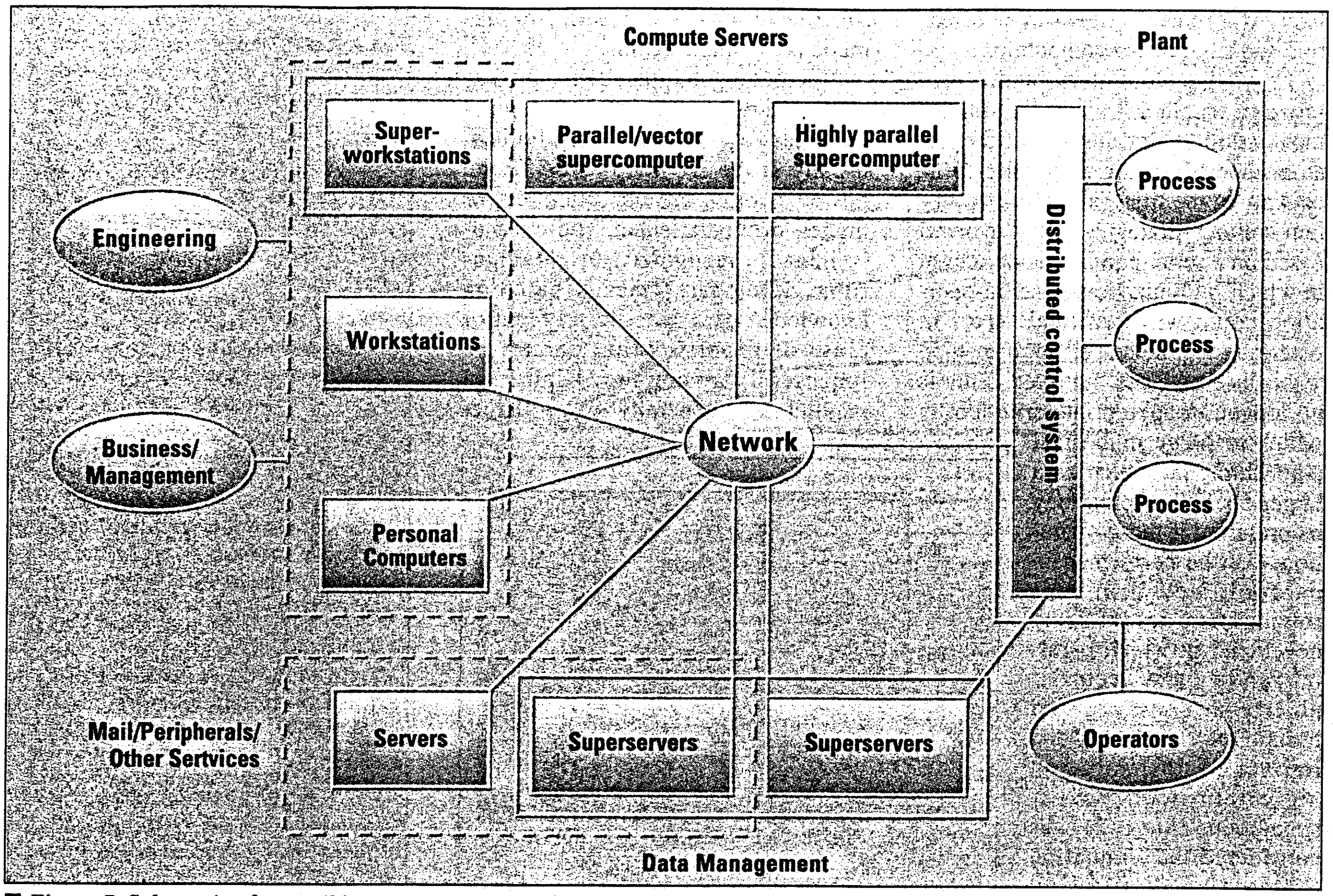
than on the low level; thus, there are opportunities for low-level parallelism within the high-level tasks. In fact, exploitation of low-level parallelism within the high-level parallel tasks allows a multilevel concurrency that significantly enhances the overall parallel performance of an algorithm. Use of such hierarchical parallelism often is the key to achieving good performance, because the Amdahl's Law limits apply only on each level in the hierarchy. For instance, consider a case in which 64 processors are divided into 8 clusters of 8 processors each, with a high-level strategy used to parallelize computations across the clusters, and low-level parallelization used within each cluster. If the low-level parallelization is 95%, then the Amdahl's law speedup within each cluster is about 5.9; if the high-level parallelization is also 95%, then 8 clusters perform about 5.9 times faster than one cluster. This results in an overall speedup of about 35, over twice the speedup of 15.4 achievable with 95% parallelization on a single level of 64 processors. One example of the use of a high-level parallel strategy is in the simulation of equilibrium-stage separation columns *(27)*. If we are to truly exploit, as opposed to simply use, the power of high-performance computing, a rethinking of the strategies used to solve many chemical engineering problems is necessary.

The computational hardware of the future will be determined in a competitive marketplace, and not by who wins the teraflop contest. Keys will be

M. A. STADTHERR is on the faculty of the department of chemical engineering at the University of Illinois at Urbana-Champaign (217/333–0275; Fax: 217/244–8068; e-mail: m-stadtherr@uiuc.edu). His research interests are focused on the development of strategies for effectively exploiting high-performance computer architectures in solving process-engineering problems in simulation, optimization, operations, and design. He holds a BChE from the Univ. of Minnesota and a PhD in chemical engineering from the Univ. of Wisconsin. A member of AIChE, he is active in the Institute's Computing & Systems Technology Division.

H. N. COFER currently is a graduate research assistant at the National Center for Supercomputing Applications at the University of Illinois, where she provides support, consulting, and training for chemical engineering applications. She received a BS from Rice Univ. and an MS from the Univ. of Illinois, both in chemical engineering. She now is working toward a PhD in Prof. Stadtherr's research group.

K. V. CAMARDA now is a graduate research assistant in the department of chemical engineering at the University of Illinois, where he is working toward MS and PhD degrees under Prof. Stadtherr. Previously, he worked at the Univ. of California at San Diego (UCSD) Medical Center on a pharmaceutical engineering project. He received a BS in chemical engineering from UCSD.



■ *Figure 7. Schematic of a possible metacomputer configuration in a chemical engineering environment.*

not only affordability, but also applicability — namely, the availability of a large number of application codes that effectively exploit the technology. The real winner in such a marketplace should be the users, provided they make sure the criteria used in the contest are theirs and not the vendors, some of whom may emphasize high theoretical peak-performance rates at the expense of applicability.

The chemical engineering metacomputer

One possible metacomputer configuration for a chemical engineering environment is shown schematically in Figure 7. This metacomputer ties end users responsible for various functions to the process and to powerful computational and data-management resources. Engineers and other users interact directly with desk-based client machines ranging in power from low-end personal computers to high-end multiprocessing superworkstations. Computations, such as complex modeling, scheduling, or planning problems, that cannot be handled adequately by the users' desk-based machines are passed transparently to high-performance compute servers that may be based on parallel/vector or highly parallel architectures. Sufficiently powerful client machines also may be authorized to act as compute servers when they are not otherwise occupied. A plant's distributed control system also will be a client for the compute servers, which will provide the real-time computational power to enable the use of realistic models in advanced model-based control strategies, model-based fault diagnosis, and plantwide optimization. Process data are reconciled and managed using multiprocessing superservers, which also provide other database services. Thus, past and present process data are available immediately, and can be managed and used easily.

By providing access to common models and databases, as well as the computational power to use them effectively, the metacomputer will foster teamwork and integration across company functions. Such integration now is relatively common in the discrete-parts-manufacturing industry, and in that context is referred to as Computer-Integrated Manufacturing (CIM). Similar integration *(28,29)* is considerably more difficult in the process industries, and today is at a relatively early stage of development. (There isn't even a standard acronym for it.) These integration efforts basically are aimed at the development of corporate strategies and enabling hardware and software for getting the right information and decision-making tools to the right place at the right time in order to arrive

Literature Cited

1. National Research Council, "Frontiers in Chemical Engineering: Research Needs and Opportunities," National Academy Press, Washington, DC (1988).
2. "Supercomputing in Chemical Engineering," *Chem. Eng. Progress*, **85**(10), p. 17 (Oct. 1989).
3. **Pekny, J. F.,** "Communications: Profiting From an Information Explosion," *Chem. Eng. Progress*, **89**(11), p 51 (Nov. 1993).
4. **Latta, S., and L. Lane,** "Designing a Metacomputer to Increase Research Power," *access* **5**(5), Natl. Ctr. for Supercomputing Appl., Univ. of Illinois, Urbana-Champaign, p. 4, (Sept. 1991).
5. **Khokhar, A. A., V. K. Prasanna, M. E. Shaaban, and C.-L. Wang,** "Heterogeneous Computing: Challenges and Opportunities," *Computer*, **26**(6), p. 18 (June 1993).
6. **Berson, A.,** "Client/Server Architecture," McGraw-Hill, New York (1992).
7. **Dongarra, J. J.,** "Performance of Various Computers Using Standard Linear Equations Software," Report CS-89-85, Computer Sci. Dept., Univ. of Tennessee, Knoxville, TN (May 31, 1993). [Note: an up-to-date Postscript copy of this report can be obtained on-line by sending the message *send performance from benchmark* to netlib@ornl.gov.]
8. **Stallings, W.,** "Reduced Instruction Set Computers (RISC)," 2nd ed., IEEE Comp. Soc. Press, Los Alamitos, CA (1990).
9. **Ryan, B.,** "RISC Drives PowerPC," *BYTE*, **18**(9), p. 79 (Aug. 1993).
10. **Thompson, T.,** "PowerPC Performs for Less," *BYTE*, **18**(9), p. 56 (Aug. 1993).
11. **Ricutti, M.,** "Full-Strength UNIX Finally Comes to PCs," *Datamation*, **39**(14), p. 47 (July 15, 1993).
12. **Vegeais, J. A., A. B. Coon, and M. A. Stadtherr,** "Advanced Computer Architectures: An Overview," *Chem. Eng. Progress*, **82**(12), p. 23 (Dec. 1986).
13. **Fox, G. C., and P. C. Messina,** "Advanced Computer Architectures," *Sc. Amer.* **257**(4), p. 67 (Oct. 1987).
14. "Understanding Computers: Speed and Power," Time-Life Books, Alexandria, VA (1987).
15. "Encyclopedia of Computer Science," 3rd ed., A. Ralston and E. D. Reilly, eds., Van Nostrand Reinhold, New York (1993).
16. **Hwang, K., and F. A. Briggs,** "Computer Architecture and Parallel Processing," McGraw-Hill, New York (1984).
17. **Almasi, G. S., and A. Gottlieb,** "Highly Parallel Computing," Benjamin/Cummings, New York (1989).
18. **Moldovan, D. I.,** "Parallel Processing: From Applications to Systems," Morgan Kaufmann, San Mateo, CA (1993).
19. **Vaughannichols, S. J.,** "Catch as Cache Can," *BYTE*, **16**(6), p. 209 (June 1991).
20. **Davis, D. B.,** "Multiprocessing Software Plays Catch-Up," *Datamation* **39**(11), p. 77 (June 1, 1993).
21. **Helling, R. K., R. D'Souza, and P. D. Glover,** "Achieving the Optimum of a Multiproduct Chemical Plant," *Proc. 2nd Intl. Conf. on Foundations of Comp.-Aided Proc. Op.* (FOCAPO-II), CACHE Corp., Ann Arbor, MI (1993).
22. "Grand Challenges 1993: High Performance Computing and Communications," Fed. Coordinating Coun. for Sc., Eng., and Tech. (1993). [available from the Natl. Sc. Found., Comp. and Inf. Sc. and Eng. Dir., 1800 G. Street NW, Washington, DC 20550].
23. **Zitney, S. E.,** "Sparse Matrix Methods for Chemical Process Separation Calculations on Supercomputers," *Proc. Supercomputing '92*, IEEE Comp. Soc. Press, Los Alamitos, CA (1992).
24. **Stadtherr, M. A., and E. S. Wood,** "Sparse Matrix Methods for Equation-Based Chemical Process Flowsheeting: II. Numerical Phase," *Comput. Chem. Eng.*, **8**, p. 19 (1984). [Note: up-to-date information on the LU1SOL code is available from Prof. Stadtherr.]
25. **Zitney, S. E., and M. A. Stadtherr,** "Frontal Algorithms for Equation-Based Chemical Process Flowsheeting on Vector and Parallel Computers," *Comput. Chem. Eng.*, **17**, p. 319 (1993).
26. **Zitney, S. E., K. V. Camarda, and M. A. Stadtherr,** "Impact of Supercomputing in Simulation and Optimization of Process Operations," *Proc. 2nd Intl. Conf. on Foundations of Comp.-Aided Proc. Op.* (FOCAPO-II), CACHE Corp., Ann Arbor, MI (1993).
27. **O'Neill, A. J., D. J. Kaiser, and M. A. Stadtherr,** "Strategies for Multicomponent Equilibrium-Stage Separation Calculations on Parallel Computers," *AIChE J.*, to be published in 1994.
28. **Venkatasubramanian, V., and G. M. Stanley,** "Integration of Process Monitoring, Diagnosis, and Control: Issues and Emerging Trends," *Proc. 2nd Intl. Conf.on Foundations of Comp.-Aided Proc. Op.* (FOCAPO-II), CACHE Corp., Ann Arbor, MI (1993).
29. **Macchietto, S.,** "Bridging the Gap — Integration of Design, Operations Scheduling and Control," *Proc. 2nd Intl. Conf. on Foundations of Comp.-Aided Proc. Op.* (FOCAPO-II), CACHE Corp., Ann Arbor, MI (1993).

Acknowledgments

The authors' research work in the use of high-performance computing in chemical process simulation, optimization, and design is supported in part by the National Science Foundation under grant DDM-9024946. The authors also acknowledge the support of the Dow Chemical Co., and the National Center for Supercomputing Applications at the University of Illinois.

at the right decision. Today, many firms are making significant efforts and progress along these lines. Such efforts, together with advances in computational power and the techniques to use it, are leading to the chemical engineering metacomputer of the future.

A powerful opportunity

The impact of future hardware and software at all levels — personal computer, workstation, or high performance, working alone or especially when working together as a metacomputer — is to provide more computing power in the hands of more people less expensively than today. For chemical engineers this means new fundamental discoveries, leading to new and better products, and processes that are cleaner, safer, more efficient, and less costly. It also means enhanced productivity and integration, resulting in speedier response to changes in economic, regulatory, and technological conditions, and faster time to market with new products and processes. Taking good advantage of the opportunities to exploit advances in computational power will be necessary to maintain and strengthen competitiveness in the global market. CEP

To receive a free copy of this article, send in the Reader Inquiry Card in this issue with No. 162 circled.