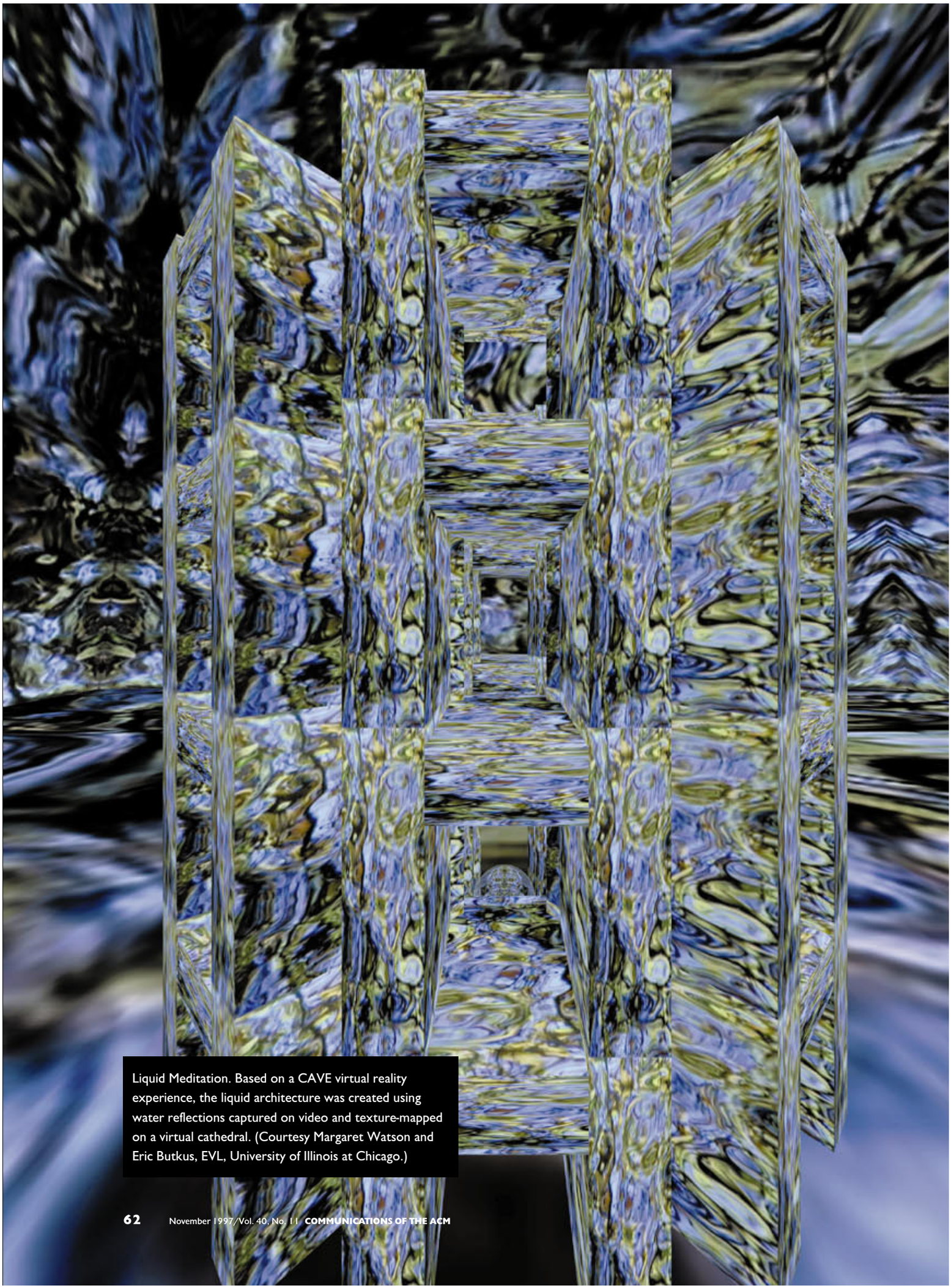


Liquid Meditation. Based on a CAVE virtual reality experience, the liquid architecture was created using water reflections captured on video and texture-mapped on a virtual cathedral. (Courtesy Margaret Watson and Eric Butkus, EVL, University of Illinois at Chicago.)

A Nationwide Parallel Computing Environment

How to get there? Through a scalable software environment coupling the small parallelism (a few processors) of the desktop to the large-scale parallelism (thousands of processors) in Alliance machines.

Since its beginning in 1985, the National Science Foundation's Supercomputer Centers Program has provided access to high-performance computing to the U.S. computational science and engineering community. Access initially meant classic shared memory parallel vector processor (PVP) computers, followed in 1989 by the first distributed massively parallel processing (MPP) machines. Access motivated the user community to an exponential increase in high-performance use that has continued to this day (see Figure 1).

The greatest gain in computing performance since 1989 has been through the move to large-scale parallelism—enabled at NCSA by a Thinking Machines CM-2 added in 1989 and a 512-processor parallel distributed memory CM-5 in 1992. Attesting to the value of this transition, the number of NCSA projects using more than 1,000 processor hours per year (in Cray X-MP units) increased from 10 to 100 after the CM-5 became generally accessible. Today, the MPP tradition of ever-increasing performance gains continues by way of such new MPP machines as the Silicon Graphics/Cray Research T3E and the IBM SP series, now available from other centers in the NSF program.

However, in all cases, previous shared memory applications have had to be rewritten from scratch to fully exploit the highly parallel architecture. The re-coding has presented a barrier so great that many users have yet to experience the benefits of highly parallel computing.

In 1995, NCSA began a program to solve this

Ken Kennedy, Charles F. Bender, John W.D. Connolly, John L. Hennessy, Mary K. Vernon, and Larry Smarr

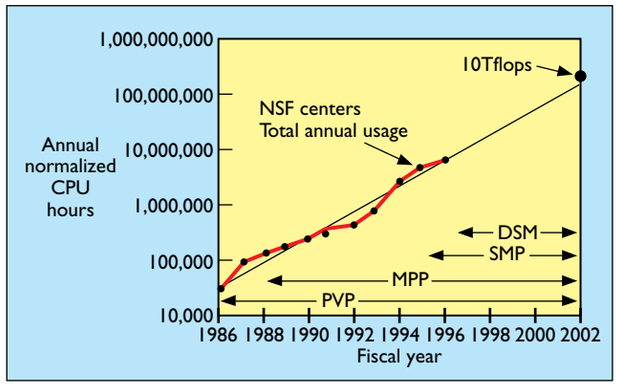


Figure 1.
Exponential growth in U.S. academic use of high-performance computers at NSF Supercomputer Centers. If this trend continues for the next five years, 10Tflops will be needed to meet it.

problem by moving to two new architectures—shared memory symmetric multiprocessing (SMP) and distributed shared memory (DSM). Instead of buying a mature MPP system, NCSA decided in 1995 to keep its CM-5, abandon its PVP computers, and move to a cluster of Silicon Graphics Power Challenge SMP units. In collaboration with the University of Illinois at Urbana-Champaign, NCSA also began to evaluate new DSM machines from Hewlett-Packard, and at the beginning of fiscal 1997, NCSA became the first place in the world to take delivery of the Silicon Graphics Origin 2000 DSM system.

NCSA's goal was to bring down the cost of shared memory computing by moving from specialized vector processors to RISC microprocessor SMP, then bring the benefits of highly parallel programming to a shared memory environment on DSM systems. Much of the community's understanding of how applications perform on DSM architectures derives from the DARPA-funded Stanford University DASH project, which was motivated by the realization that programming distributed-memory machines without cache coherency is too difficult when compared with their bus-based SMP counterparts implementing cache-coherent shared memory. Lessons learned from DASH implementation in DSM-architecture machines should ensure an easy-to-use programming model in the form of a single large address space, automatic hardware- and software-enabled cache coherence, and low internal latency for high-memory bandwidth.

Supercomputers have now become "scalable servers" at the top end of product lines that start with desktop workstations. Market-driven companies, like Silicon Graphics, Hewlett-Packard, and IBM, have a much broader focus than just building the world's fastest computer, and the software of the desktop computer has a much greater overlap with that of the supercomputer than it used to, because both are built from the same cache-based microprocessors.

The new NSF National Computational Science Alliance, with its NCSA Leading Edge site and Advanced Hardware partners, are the "supernodes" of the National Technology Grid. In addition to providing access to the supernodes, the Alliance is also connecting them via high-speed networks to form either homogeneous virtual power subgrids of similar scalable machines or to create U.S. national-scale heterogeneous metacomputers (see Stevens et al. in this issue). Using new developments in Web- and Java-based technologies, the Alliance is also forming "knowledge grids" of consulting, training, and performance expertise to create an unprecedented level of support for end users.

Because U.S. user community application codes are spread out across a variety of PVP, MPP, SMP, and DSM machines, users need methods to conserve their software development investments as hardware changes occur. Specifically, the Alliance's Enabling Technologies (ET) team on Parallel Computing is developing standard, portable programming environments and system interfaces to make it easier for users to port, develop, and optimize their codes on Alliance architectures. This ability becomes even more crucial as the 10-year NSF Supercomputer Centers Program comes to an end and the new NSF Partnerships for Advanced Computational Infrastructure (PACI) program begins and users from some of the current supercomputer sites make the transition to the Alliance and the National Partnership for Advanced Computational Infrastructure (NPACI).

This article lays out the five-year plans for both production and experimental Alliance computing hardware, then details a number of software initiatives meant to make the national Grid much more usable to a wider set of users.[1]

Parallel High-Performance Computers

Thousands of U.S. computational scientists and engineers expect the NSF Leading Edge centers to continue increasing their capacity exponentially. This expectation is quantified by Ostriker and Norman in this issue to include not only computers capable of sustaining teraflops performance but a central memory of hundreds of GB and a balanced I/O and storage capability (see also Reed et al. in this

issue). Achieving such performance in the next three years is not a matter of buying the latest hardware but a common effort among users, computer scientists, and leading-edge vendors.

One Alliance goal is to help make shared-memory parallel computing the norm, even at the desktop, over the next five years. A powerful ally in the pursuit of this goal is the rapid market move toward affordable shared-memory multiprocessors—especially in the commercial world in the form of scalable servers. The installed base of MPP machines is in the hundreds, while the installed base of servers is in the tens of thousands, workstations in the millions, and PCs in the hundreds of millions. As all these machines become increasingly parallel, demand for parallel languages and software can be expected to increase by orders of magnitude. Already, the two- or four-processor PC or workstation is no longer a novelty.

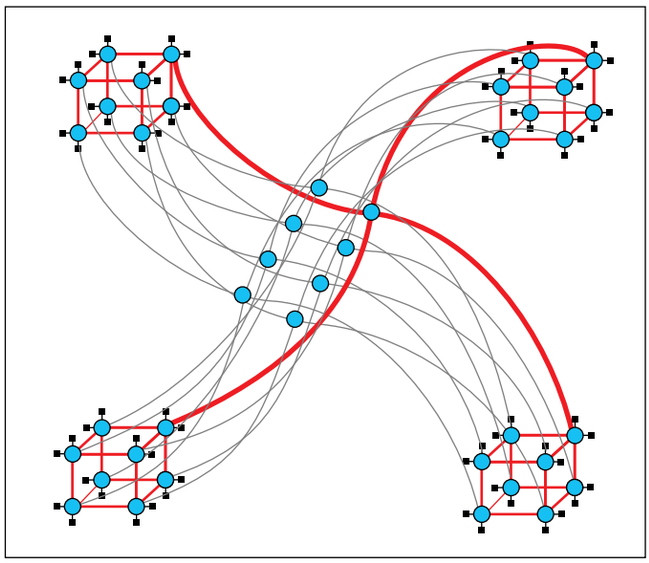
An Alliance goal is to help make shared-memory parallel computing the norm, even at the desktop, over the next five years.

Parallelism made available to an exponentially larger installed base will completely change the dynamics of parallel software development and use. Instead of being a niche market, parallel computing will become the general market. The role of the Alliance is to help develop a scalable software environment that will couple the small parallelism (a few processors) of the desktop to the large-scale parallelism (thousands of processors) in the leading-edge machines of the Alliance. Toward this end, the leading edge center at NCSA will develop a large-scale DSM machine with Silicon Graphics, moving from 32 processors under a single memory image in 1996 to from 1,024 to 2,048 processors by 2002.

Distributed shared memory. One specific approach to extending the shared memory multiprocessor to large-scale parallelism is to use DSM machines, available today from Silicon Graphics and Hewlett-Packard. These emerging high-performance computers use the same microprocessors as desktop SMP machines. Because the individual microprocessors have a cache between the main shared memory and the processing unit, if special features in software or hardware are not added, cache conflicts can result during a program's execution. This is the case for traditional MPP machines and is the major reason message passing is used as a programming paradigm so that cache inconsistencies are manually avoided. In DSM machines, cache coherency is maintained, so users can program at a higher level as if they were using a true shared-memory machine, although efficiency still requires that close attention be given to data locality. Because processors may fetch variables from local or remote memories, memory access is non-uniform. Since the cache contents are kept coherent, such architectures are sometimes referred to as Cache-Coherent Non-Uniform Memory Access (ccNUMA).

In practice, cache coherency is supported by the vendor up to a certain level of parallelism, say, 128 processors in the Silicon Graphics Origin2000. To achieve higher levels of parallelism, users can cluster the DSMs, just as one clusters single processors or SMPs. In October 1996, NCSA installed the world's first 128-processor Silicon Graphics/Cray Origin system. A second 128-processor Origin was added in June 1997. This cluster doubled to 512 processors in October 1997 and will double again to 1,024 processors in 1999. The Origin cluster builds on NCSA's two-year experience with its cluster of Silicon Graphics SMP Power Challenges (consisting of 10 Power Challenge SMPs, each with 16 processors). The Origin2000 is built up from two-processor SMP modules through a novel interconnection fabric allowing the bisection bandwidth to increase linearly with processor count (see Figure 2). The new

[1]This work is supported by the NSF's PACI program, effective October 1, 1997.



COURTESY JOHN BROWN, SILICON GRAPHICS

Figure 2. Silicon Graphics/Cray Origin2000 CrayLink Scalable Interconnect Network.

CrayLink Interconnect Network removes the bus-based bottlenecks limiting efficient parallel scaling to 8–12 processors on the Silicon Graphics Power Challenge SMP machine.

In a proof-of-principle effort, researchers at the University of Minnesota's Laboratory for Computational Science and Engineering generated the highest-resolution 3D simulation ever of the convection in a rotating model star by running a new communication-latency-tolerant code for parallel computer architectures on NCSA's Origin array (see Figure 3). The simulation used two 64-processor systems interconnected by 100MB/sec High Performance Parallel Interface (HiPPI) channels.

In a nine-day run on NCSA's 128-processor Silicon/Cray Origin2000 using more than 25,000 cpu-hours, the Minnesota simulation calculated more than 18,000 time steps—advancing 57 million active computational cells from a grid with 169 million cells overall—to simulate the convection process in a rotating star. This astrophysical gas dynamics simulation was computationally intensive, consisting of more than 3.5 million billion floating-point operations and generating more than 2TB of archived information. The researchers' code prevents latency from slowing a computation, because the code overlaps the computation with an exchange of information among the other processor groups. Rather than eliminating latency, the code hides it. Hiding latency is easier as the size of a problem increases, because the amount of time spent computing increases faster than the time spent exchanging data between processor groups. In large problems, like solar convection, the time spent computing exceeds by far the time spent exchanging data.

Future generations of Silicon Graphics/Cray hardware will improve the speed and number of individual processors in a single-memory image DSM machine, culminating in about five years in a 2,048-processor DSM architecture capable of over 10Tflops. At the same time, the high-level DSM design will remain the same. For users, this means a smooth software upgrade path to shared-memory programming on future DSM architectures. The same architecture also scales downward to workstations and servers with 2, 4, 8, . . . processors using the same technology as the Origin. This scalability provides "desktop-to-teraflop" scalability by having the same software and hardware architecture on remote user workstations and departmental servers as on the teraflop Leading Edge facility at NCSA.

As a result of NCSA's moves toward SMP/DSM clusters, the five years centered on 1997 are seeing an

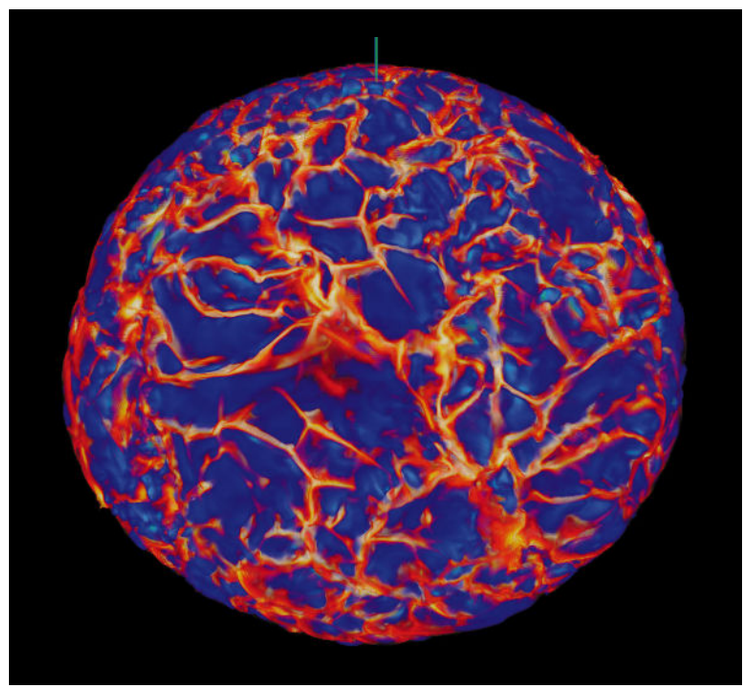


Figure 3. Velocity structures in a rotating model star with convection in equilibrium in its unstable outer half. (Simulation by D. Porter, S. Anderson, J. Habermann, and P. Woodward on two 64-processor Silicon Graphics Origins at NCSA, 1997.)

unprecedented rate of growth in shared-memory-machine compute capacity (see Figure 4). For the first 10 years of NCSA, shared memory capacity grew at only 24% per year compounded. This low rate is because parallelism grew only slightly from two to eight processors, and the growth in peak speed of vector processors was fairly slow. In January 1995, NCSA shut down its vector SMPs from Cray Research and replaced them with microprocessor

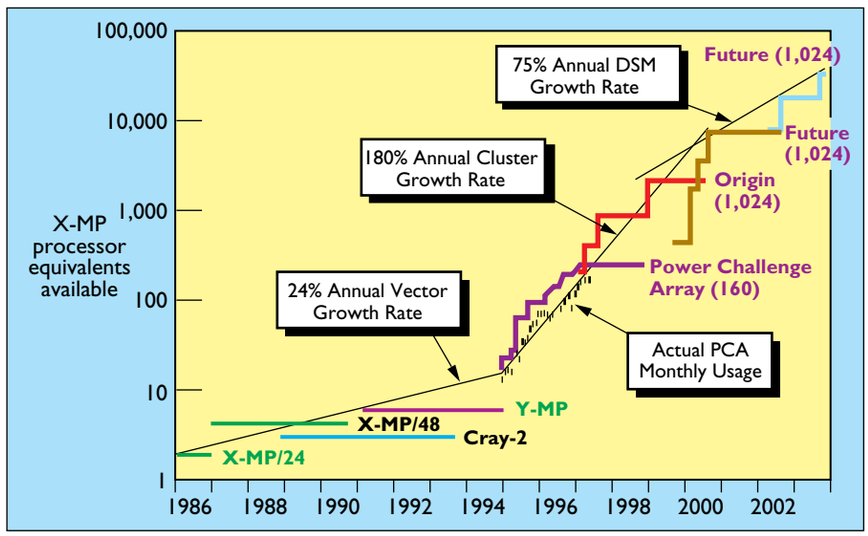


Figure 4. Past, present, and planned high-performance computing capacity at NCSA, the Alliance's leading-edge site (capacity normalized by LINPACK benchmarks).

SMPs from Silicon Graphics with the same number of processors—eight. However, in the same five years, NCSA's shared-memory compute capacity has been soaring at an unbelievable 180% per year compounded growth rate. The reason is simple and twofold: The microprocessors themselves had a much faster rate of growth in speed (60% per year) than the vector processors, and the parallelism increased rapidly from 8 to 1,024. The synergy between these two effects enabled this great increase in speed to happen with roughly fixed capital investment per year, which is also indicative of the superior price/performance ratio of microprocessor-based computers.

However, users will have even less time to adapt their codes to the new machines than they had in the past because of the increased capacity growth rate. From this opportunity and dilemma comes the strategy of creating the focused Alliance Application Technologies (AT) teams to work closely with computer scientists in the Parallel Computing team to take advantage of the new shared-memory parallelism early in product lifetimes, thus enhancing the support and training the Alliance provides the broader user community.

Finally, the shared-memory programming model used on these machines will ease the transition from sequential machines as well as from vector and small-microprocessor SMP machines. Although performance tuning is still necessary, just getting the program running will not be the heroic effort it was for early message-passing architectures.

Many of NCSA's Industrial Partners have strong relationships with third-party commercial engineering and analysis code vendors. The Alliance plans to leverage these relationships to bring DSM-enhanced versions of commercial codes to market—many months or years before they would be available without such partnerships with commercial clout. In the Alliance's first year, beginning October 1997, there is particular concern with helping academic users running codes on classic MPP architectures (Thinking Machines CM-5, Cray T3D/E, and IBM SP) move to the cluster of Origin computers. For an example of how the DSM architecture allows a new type of computational science—dynamic adaptive mesh refinement in cosmological evolution—that would be very difficult on a traditional MPP, see Ostriker and Norman in this issue. Portable languages and performance analysis tools are critical for achieving these goals and will be further developed by the Parallel Computing team.

Architectures at Advanced Hardware sites. In addition to the NCSA architectures being made available to the national user community, Alliance partners are also making available cycles on their machines, extending and complementing NCSA's architectures.

Having a variety of architectures guards against putting all our eggs in one basket in a fast-changing technological environment. These architectures also provide important facilities for comparing application performance across architectures while serving as a testbed for computer scientists developing portable computing languages and environments. User support, including documentation and training

activities, will be shared across the Alliance Advanced Hardware sites, increasing the number of staff working with the national user community.

The following sites provide Alliance teams and users access to a variety of production parallel systems, including vector and microprocessor SMP, MPP, and DSM architectures:

- NCSA and the University of Illinois at Urbana-Champaign, using Silicon Graphics/Cray Power Challenge Array and Origin2000 cluster; Hewlett-Packard/Convex SPP-1200 and -2000; and NT /Intel SMP cluster
- Boston University, using Silicon Graphics Cray Origin2000
- University of Kentucky, using a Hewlett-Packard/Convex SPP-1200
- Maui High Performance Computing Center (operated by the University of New Mexico), using a large IBM SP with single-processor and SMP nodes
- Ohio Supercomputer Center, using a Cray Triton 94; a Cray J90/16; a Cray T3E-LC512; a small IBM SP; Silicon Graphics Power Challenge; and Hewlett-Packard/Convex SPP-1200

Many Alliance partner sites also have smaller versions of these machines. For instance, there are smaller Origins at Princeton University, Argonne National Laboratory, Indiana University, and the University of Utah. Plans for linking them to form a national-scale metacomputer are discussed by Stevens in this issue.

Experimental Architectures

The Alliance is also experimenting with emerging architectures, giving the early adopters in the national user community access to architectures years before they become mainstream computing commercial offerings. It also gives computer scientists access to large-scale versions of these new machines so they can develop the software needed to take advantage of them. Two of these experimental architectures involve harnessing clusters of NT/Intel SMP machines and coupling high-speed parallel graphics with parallel computing to create a visual supercomputer.

NT SMP array. Commodity PC hardware and software technologies are an increasingly important vehicle for computational science. The Alliance plans to explore the viability of Microsoft Windows NT SMP clusters as a high-performance computing resource. The testbed for this development is the University of Illinois at Urbana-Champaign Computer Science Department NT cluster [5]. So far, experiments have produced a layer of low-latency, high-bandwidth messaging software running on a cluster that can outperform tightly integrated MPP systems on certain problems. NCSA will deploy a production cluster of NT SMPs in 1998, by first bringing a variety of computational disciplines to this new operating system, later bringing scalability to high-speed networked clusters of SMPs. In order to exploit and expand these new technologies, NCSA has formed a partnership with Hewlett-Packard and Microsoft to create an NT/Unix interoperability testbed to allow investigation of scalable NT computing, bringing Unix programming tools to NT platforms and enterprise management technology.

Performance analysis on both the NT cluster and the Hewlett-Packard SPP-2000 Unix ccNUMA machine on user codes will prepare the way for the next-generation Hewlett-Packard/Convex SPP-3000 system, a cache-coherent version of clustered Intel/Hewlett-Packard Merced SMP. The Merced microprocessor will be software backward-compatible with both the Hewlett-Packard PA-RISC 8000 and the Intel Pentium Pro. The Alliance will work with Hewlett-Packard and Microsoft to run both NT and Unix on the SPP-3000 and to run NT on clustered Merced SMPs. Improvements in the processor and interconnection technologies should lead to a teraflops-capable NT shared-memory system within five years. Such a production system should be a critical national asset, because it will provide a single technical computing environment for the large number of computational science users switching to NT/Intel workstations and a scalable commercial/financial server for NCSA Industrial Partners interested in data mining, financial engineering, and decision-support applications.

Visual supercomputers. The term "visual supercomputing" refers to the integration of parallel computing and parallel computer graphics through a DSM architecture. The Alliance goal is to bring the same high performance to visual analysis that scalable computing brought to computation. The Alliance intends to create a National Immersive Analysis Facility at NCSA offering such capabilities as allowing parallel graphics engines to direct all their polygon-generation capabilities to one screen, using a flexible composited tiling approach, to produce well over a million polygons/sec on a single screen in real time. The same set of parallel engines can drive up to 16 high-resolution screens, unified into a "Great Wall of Power" fed by shared-memory, 100MB/sec Fibre Channel disks (stored data) or the vBNS net-

work (computed data). The visual supercomputer will provide up to 100 times improvement over current Power Challenge surface rendering from data on disk, due to the large memory available over the large number of parallel processors. Finally, the highly parallel large-memory visual supercomputer can accelerate implementations of the newest algorithms for volume rendering using wavelet compression, scattered data concepts, and transparency/shading as the algorithms emerge from the graphics community.

Parallel Software Development

The Parallel Computing team will identify user codes that can reach higher performance by exploiting the features of DSM, using these codes to drive R&D of new compiler technology and libraries. The team will then work with the vendors to ensure that these technologies are made widely available in commercial form. The ultimate goal is to produce a programming environment supporting construction of codes portable across all Alliance architectures while achieving the highest possible performance on the DSM configurations.

Languages provide the most visible interface between a programmer and a parallel machine. Designing a parallel language is a matter of aesthetics and practicality; it must be understood and easy to use by humans but implemented with high efficiency on complex hardware. For example, three Alliance projects provide a shared-memory-like interface: High Performance Fortran (HPF), High Performance C++ (HPC++), and TreadMarks. These languages are currently available, but efficient implementation on DSM architectures and strategies for avoiding blocking on memory still need to be developed. As DSM architectures become more complex, hiding this complexity through compiler or run-time analysis will be critical for providing performance to users.

HPF. As is an extension of Fortran 90, HPF supports machine-independent scalable parallel programming by enabling programmers to specify the assignment of data to processors for major data structures. The compiler generates the required synchronization and communication. Major compiler optimizations include minimization of communication and generation of computation/communication overlapping using machine primitives. Rice University has developed many of the fundamental compilation strategies in this area and is now extending these techniques to DSM systems [9].

Key technical challenges include optimizing for locality, tailoring codes to various processor-to-node ratios, addressing the problem of false sharing of cache lines, and effective use of prefetch. Of particular interest is implementation of adaptive, distributed computations on DSM systems in which the hardware support for dynamic remote access may provide advantages over message passing. HPF 2.0 includes new distributions intended for such problems.

The Parallel Computing team will work with AT team codes on Alliance DSM architectures to get early examples of optimized programming to the national community. These extensions to Fortran are very attractive to the Cosmology, Nanomaterials, and Scientific Instrumentation teams, partly because they already have vectorized Fortran and Connection Machine Fortran codes to parallelize for DSM architectures. For example, spin dynamics codes used by the Nanomaterials team are essentially pointwise updates of a data structure suited to HPF.

HPC++. An HPC++ industry-university-government consortium is building a programming model and software tools for building object-oriented applications on high-performance computing platforms. These include MPP systems, shared-memory multiprocessors, and heterogeneous networks of compute servers. HPC++ contains a set of extensions and class libraries for C++ that helps users build meta-applications from smaller data-parallel applications through an improved technology for coupling systems. team members have demonstrated that HPC++ can be used to encapsulate parallel applications written in other languages, such as HPF and Fortran with MPI, into objects and used as components to build large distributed applications.

HPC++ supports the notion of concurrent objects of the same type but not necessarily of the same size. The programmer specifies how the objects communicate with one another and with their parent objects at a high conceptual level, improving the code's readability and portability. In addition, object-oriented features can be layered on many other programming paradigms, improving the modularity and maintainability of the resulting applications. Compilation issues include integrating efficient run-time libraries for HPC++'s dynamic features with HPF-like static optimization techniques. Indiana University led the standardization process for HPC++ and is further developing the language and its compilers for the Alliance [7].

TreadMarks. TreadMarks is a run-time system providing a global shared-address space across the machines on a network of Unix workstations; that is, it efficiently emulates a DSM system while running

on a distributed-memory system. It can be either used as implementation support for HPF and HPC++ or called directly by application codes. Alliance researchers will explore software DSM implementations via enhancements to the TreadMarks system; planned research includes integration of compiler and run-time techniques and use of high-level synchronization operators and multithreading, especially on multiprocessor nodes. Rice University will lead development of a Windows NT version, thus providing a software development environment on PC networks with a DSM interface compatible with Alliance DSM machines. Users could then use the NT clusters for development or for production runs of their code. Existing shared-memory codes could therefore be moved to NT clusters with minimal effort [1].

Although performance tuning will still be necessary, just getting the program running will not be the heroic effort it was for early message-passing architectures.

Software Performance Analysis for Parallel Computers

Although languages like HPF and HPC++ support portable high-level programming models, programmers may need to customize code to a particular machine to achieve acceptable performance. Fortunately, this tuning can often be done using machine-independent constructs (for instance, by judiciously choosing HPF distribution directives). To achieve this ideal, however, high-level languages need sophisticated tools for program construction and tuning. Such tools are analogous to the language-level symbolic debuggers and profilers now considered essential on sequential computers. Such parallel software is not widely available today because HPF and HPC++ are further removed from the hardware than sequential languages were; translating machine behavior into language terms is much more difficult—a problem the Alliance is attacking.

The Pablo project, led by the University of Illinois at Urbana-Champaign, has pioneered performance analysis and visualization technology for distributed-memory machines [12]. Pablo is a portable, extensible, and scalable software performance analysis infrastructure that includes instrumentation at multiple hardware and software levels, real-time data reduction, and data analysis tools. Pablo has been extended to SvPablo—a graphical source code browser and performance visualizer that integrates the project's dynamic performance instrumentation software with the Portland Group's commercial HPF compiler (PGI HPF). SvPablo can integrate dynamic performance data with information recorded by the HPF compiler to describe mapping from the high-level source to the low-level, explicitly parallel code.

To capture dynamic performance data, the PGI HPF compiler emits code with embedded calls to the Pablo performance instrumentation library. During execution of the instrumented code, the Pablo library maintains statistics on the execution of each generated construct on each processor and maps these statistics to constructs in the HPF original source code. Because these are statistics, rather than detailed event traces, Pablo can measure the performance of HPF codes that execute for hours or days on hundreds of processors. After execution, Pablo creates a single performance file; the SvPablo browser then provides a hierarchy of color-coded performance displays, including high-level routine-profile and source-code scroll boxes, that contain statistics and detailed per-processor metrics for each processor and each routine or line. Future enhancements under development include scalability predictions based on symbolic expressions derived from compiler-generated code.

Communication and Scientific Libraries

A key component in scientific programming is efficient, standard library implementations of com-

monly used algorithms. Examples abound within the Alliance AT teams:

- Chemical Engineering uses sparse linear algebra routines in process simulation.
- Nanomaterials relies on dense and sparse linear algebra to solve electron structures.
- Advanced optimization methods, used by all AT groups to solve design problems, rely on efficient underlying linear algebra.
- Scientific Instrumentation uses Fast Fourier Transforms as a key part of image processing.

Another key aspect of libraries is support for particular communication patterns, so, for example, all application areas use codes relying on MPI, and all scientific subroutine libraries rely on efficient data movement at a low level.

The Alliance is updating standard scientific libraries to make porting applications easier and developing new algorithm encapsulations for situations in which library interfaces are awkward. Existing versions of scientific libraries generally use MPIs but need more efficient low-level communication libraries for maximum advantage on DSM systems.

Scientific applications often use standard numerical algorithms that can be encapsulated into libraries and reused in many different settings. Reuse justifies the optimization effort needed on each target architecture. Within the Alliance, scientific library work includes:

- The ScaLAPACK group at University of Tennessee is extending its distinguished record in numerical linear algebra to DSM architectures [4].
- Argonne is extending the PETSc library algorithms and data structures, providing much of the numerical infrastructure for solving partial differential equations on DSM machines [2].
- The University of Houston is concentrating on improving the FTPACK library and extending and improving interfaces of several libraries for data-parallel operation [8].

Because message-passing systems were the first scalable parallel architectures, much work concentrates on communication libraries. Such libraries also make sense on DSM systems, since any access to memory on a remote node can be viewed as communication. Alliance programmers will use these libraries directly as another programming language and indirectly through scientific libraries. Two projects are under way:

- The University of Houston is developing techniques for managing data motion through good communication scheduling and data allocation on DSMs [10].
- The SUMAA3d project at Argonne is improving DSM application performance by saving and optimizing repeated irregular communications patterns and improving data partitioning.

Advanced Application Support

Several AT teams require advanced methods and tools that work well on parallel computers, particularly on DSMs. For example, the Environmental Hydrology team is studying environmental processes, including the environmental health of Chesapeake Bay and flood management on the upper Mississippi River, both modeled as complex systems of linked processes. Because phenomena occur on many temporal and spatial scales, adaptive methods are required to accurately solve the systems in reasonable time. The Cosmology and Nanomaterial teams also use adaptive methods for similar mathematical (albeit different physical) reasons.

The Chemical Engineering team designs and optimizes manufacturing processes (see McRae in this issue), with scales ranging from individual reactors to cooperating sets of chemical plants. In addition to adaptive methods for submodels, the high-level designs require efficient numerical optimization procedures to rationally choose alternatives for evaluation.

The Nanomaterials and Environmental Hydrology teams can also use optimization methods for their long-term goals of designing new processes. And the Molecular Biology team, which uses optimization to solve protein structures, is collecting its software into the Biology Workbench, an environment allowing biologists to find the most effective available computational methods and apply them to their problems.

Parallel adaptive methods. The Parallel Computing team is developing technologies to support advanced algorithmic methods, such as adaptive meshing efficiency on DSM architectures. Adaptive mesh refinement is a technique for handling problems requiring much greater accuracy of solutions in a few areas of the computational domain that are not known until run time. These methods work by creating new data structures (typically trees or meshes) at run time to refine the high-accuracy areas. Because the methods are inherently hierarchical, it is natural and efficient to implement them by building a layered approach, allowing users to concentrate on the abstractions at the top of the hierarchy.

The University of Texas at Austin's Distributed Adaptive Grid Hierarchy (DAGH) system runs on MPP systems to compute adaptive methods in NSF-funded Grand Challenge projects (see Figure 5) [11]. The Parallel Computing team will expand that work as part of the Alliance. Distributing the adaptive grid hierarchy across processors in a load-balanced way while maintaining data locality is an important challenge in computer science. The DAGH library solves this problem for distributed-memory MPP systems. In addition to data distribution, DAGH also supports visualization and interactive steering of adaptive computations.

Numerical optimization. Numerical optimization is used to pick an optimal design from infinite alternatives. These methods work by starting from a baseline design, finding a step in design space to a better design with which to replace the baseline, and then iterating this search procedure until the judgment is made that there is too little further improvement in the scoring function to warrant further investigation. Several search strategies are possible. Newton-type methods base their choice on the gradient of the objective function; transformation systems like Automatic Differentiation in Fortran mechanically produce the code to compute the exact gradient (if it exists) of an input program, making such searches feasible [3]. When derivatives do not exist, parallel direct search and similar methods use sophisticated search patterns to choose "good" sets of test points. All of these methods have proved their worth on applicable problems. Rice University is leading the effort to extend them to Alliance DSM architectures.

Conclusions

To succeed, this multiyear parallel computing development effort has to unite the best computer and computational scientists with application developers to address the bottlenecks in problem solving using current hardware and software technologies. Since most of these systems use DSM hardware, a relatively new technology, the Parallel Computing team's efforts must adapt the software research of the past decade to this new hardware paradigm. At the same time, the team's efforts will need to focus on the challenges of applying scalable parallelism in general-purpose computer systems to scientific and engineering problem solving. C

References

1. Amza, C., Cox, A., Dwarkadas, S., Keleher, P., Lu, H., Rajamony, R., Yu, W., and Zwaenepoel, W. TreadMarks: Shared-memory computing on networks of workstations. *IEEE Comput. 29*, 2 (Feb. 1996), 18–28.
2. Balay, S., Gropp, W., McInnes, L., and Smith, B. Efficient management of parallelism in object-oriented numerical software libraries. In *Modern Software Tools in Scientific Computing*, E. Arge, A. Bruaset, and H. Langtangen, Eds., Birkhauser Press, Cambridge, Mass., 1997, pp. 163–202.
3. Bischof, C., Carle, A., Khademi, P., and Mauer, A. Adifor 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Comput. Sci. Eng. 3*, 3 (Fall 1996), 18–32.
4. Blackford, L., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., and Whaley, R. *ScaLAPACK Users' Guide*, SIAM Press, Philadelphia, 1997.
5. Chien, A., Pakin, S., Lauria, M., Buchanan, M., Hane, K., Giannini, L., and Prusakova, J. High-performance virtual machines (HPVM): Clusters with supercomputing APIs and performance. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing* (Minneapolis, March 7–11, 1997), SIAM Press, Philadelphia, 1997.
6. Epema, D., Livny, M., van Dantzig, R., Evers, X., and Pruyne, J. A worldwide flock of condors: Load sharing among workstation clusters. *J. Future Generations of Comput. Sys. 12* (1996).
7. Johnson, E., Gannon, D., and Beckman, P. HPC++: Experiments with the parallel standard template library. In *Proceedings of the 1997 International Conference on Supercomputing* (Vienna, Austria, July 7–11, 1997), ACM Press, New York, 1997.
8. Johnsson, S., Jacquemin, M., and Krawitz, R. Communication efficient multi-processor FFT. *J. Comput. Phys 102*, 2 (Oct. 1992), 381–397.
9. Koelbel, C., Loveman, D., Schreiber, R., Steele, G., and Zosel, M. *The High-Performance Fortran Handbook*. MIT Press, Cambridge, Mass., 1994.
10. Nesson, T., and Johnsson, S. ROMM routine: A class of efficient minimal routing algorithms. In *Proceedings of the 1994 Parallel Computer Routing and Communication Workshop*. (1994), Springer-Verlag, New York, 1994.
11. Parashar, M., and Browne, J. Object-oriented programming abstractions for parallel adaptive mesh refinement. In *Proceedings of Parallel Object-Oriented Methods and Applications (POOMA)* (Santa Fe, N.M., Feb. 28–March 1, 1996).
12. Reed, D. Experimental performance analysis of parallel systems: Techniques and open problems. In *Proceedings of the 7th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation* (Vienna, Austria, May 1994).

Ken Kennedy (ken@cs.rice.edu) is the Noah Harding Professor of Computer Science and director of the Center for Research on Parallel Computation at Rice University in Houston, a member of the Alliance Executive Committee, and co-leader of the Parallel Computing team.

Charles F. Bender (bender@osc.edu) is director of the Ohio Supercomputer Center in Columbus, a member of the Alliance Executive Committee, and chair of the Alliance Partners for Advanced Computing Services group.

John W.D. Connolly (connolly@ukcc.uky.edu) is director of the Center for Computational Sciences at the University of Kentucky in Lexington, a member of the Alliance Executive Committee, and chair of the Alliance Resource Allocation Board.

John L. Hennessy (jh@mojave.stanford.edu) is dean of the School of Engineering at Stanford University , a member of the Alliance Executive Committee, and chair of the Alliance ET Committee.

Mary K. Vernon (vernon@cs.wisc.edu) is a professor of computer science and industrial engineering at the University of Wisconsin in Madison and a member of the Alliance Executive Committee.

Larry Smarr (pls@ncsa.uiuc.edu) is Director of the NCSA at the University of Illinois at Urbana-Champaign and was recently appointed to the Presidential Advisory Committee on High-Performance Computing and Communications, Information Technology, and the Next Generation Internet.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage; the copyright notice, the title of the publication, and its date appear; and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© ACM 0002-0782/97/1100 $3.50

View publication stats