Experiences with the IBM SP1

by W. D. Gropp
E. Lusk

One of the first IBM parallel processing computers—the SP1™—and the largest, with 128 nodes, was installed in 1993 at Argonne National Laboratory. It took only days, not months, to prepare for and migrate applications to this parallel supercomputer, demonstrating that high performance, parallelism, and portability can coexist. This paper describes the early experiences with the SP1 at Argonne, which provide lessons for supercomputer system designers and users alike. We explore what features of software technology and system architecture enabled immediate and successful use of the SP1. The paper concludes with a brief indication of why the move to the SP2™ software environment using the SP2 communication adapters, the use of the emerging Message-Passing Interface standard, and the continued use of the SP1 processors have been successful.

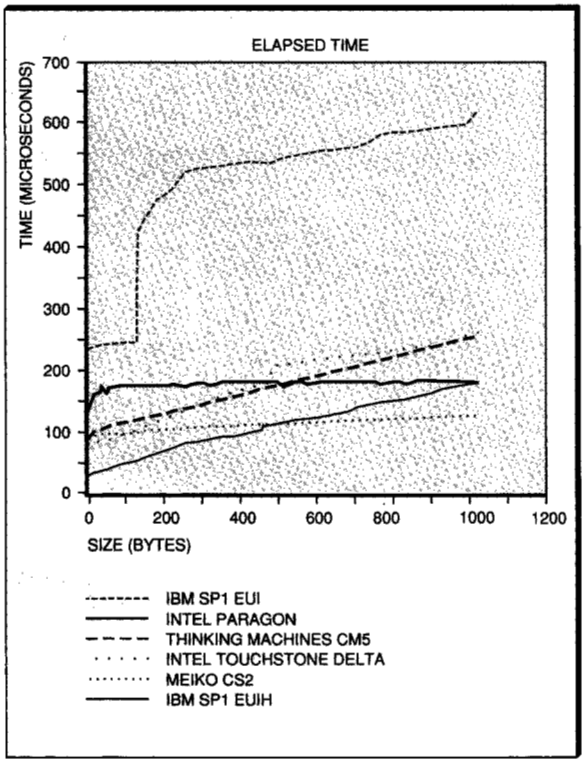
In 1991–1992, the Mathematics and Computer Science Division at Argonne National Laboratory in Chicago, Illinois, began making plans for a change in its research focus in high-performance computing. Since 1983 it had carried out research in many areas of parallel computing and had operated the Advanced Computing Research Facility (ACRF), home to early versions of parallel computers from a variety of manufacturers. Work with these machines had been responsible for significant contributions to the understanding of parallel algorithms for many scientific problems and to the development of tools for enhancing the portability of these algorithms. But it was time for a new direction. The new direction was to involve application scientists in the use of this parallel computing knowledge, in order to demonstrate the cost-effectiveness of parallel computing for large-scale scientific problems.

Such a goal required a different machine acquisition strategy from that of previous years. Although some forward-looking application scientists had used the machines of the ACRF to acquaint themselves with parallel computing issues, most had kept to their traditional supercomputers, simply because the research machines of the ACRF, suitable as they were for computer science research, did not have the computing power necessary for delivering research results in the new field of computational science.

Argonne initiated its new machine acquisition strategy by reviewing all the then-current parallel computer vendors and then choosing the newly available IBM Scalable POWERparallel Systems*. The specific machine acquired, the SP1*, had 128 nodes (larger than what was officially available at that time), with custom-designed input/output hardware. It was anticipated that the SP1 would provide an environment to which existing tools and applications could be ported quickly, providing early evidence of its usability as a "first-class scientific instrument." (In this paper, the term *port* is used to refer to the programming changes necessary to allow an application program that runs on one type of computer to run on another type of computer.)

 0018-8670/95/$3.00 © 1995 IBM

Figure 1 Communication performance for small messages for a variety of machines



All timing and performance results discussed in this paper were preliminary results. Little time was made available for single-user benchmarks; hence, many of the results are from runs on relatively small numbers of processors. The results also reflect the use of the SP1 communication adaptors (further described in References 1 and 2).

Several of the sections that follow mention the external user interface (EUI) and the high-performance external user interface (EUIH), terms that are no longer current. EUI was the name of the custom message-passing library that came with the SP1. Due to performance problems (see Figure 1), the EUI was replaced almost immediately with a user-space implementation developed at IBM Research, named EUIH.

This paper focuses on the first few months of experience with the IBM SP1 and concludes with a brief observation about the migration to the newer SP2* software, while still using the SP1 hardware. The projects reported here consist of tools projects (summarized in Table 1), reflecting both the porting of existing tools to the SP1 and in some cases the development of new ones, and application projects (summarized in Table 2), many of which use one or more of the tools.

Successful migration to the SP1

It took only a matter of days and not months to prepare for and move applications to the SP1 parallel computer. What was it that made the IBM SP1 such a success? The experiences reported in this paper provide support for some of the expected reasons, but also for a few unexpected ones.

The single most important reason that the machine was effectively used so quickly was that so many advanced users of parallel computers had already converted their codes to a portable form in order to run their applications in a variety of existing environments. They were supported by a number of tools that were geared to provide portability among parallel systems. It has not been found (by this community) that parallel algorithms must be completely different on each machine in order to achieve efficiency. Although some machine-specific tuning can make a performance difference, the tuning need not affect the overall design of the program. Efficient parallel algorithms can be designed and coded in a portable way. A variety of portability tools contributed to the speed with which applications were moved to the SP1 machine; they are described in the next section. Use of the message-passing model in the design of the SP1 meant that the machine was not very different from existing parallel machines, and so both tool writers and users found themselves in a familiar environment.

The use of off-the-shelf hardware and software components (the IBM RISC System/6000* and the Advanced Interactive Executive* system, AIX*) meant that the machine came with many proven industrial-strength pieces, despite being very new. This was particularly true of the "single-node" software components, such as AIX, the FORTRAN and C compilers, and the dbx debugger.

The use of high-performance single nodes as a fundamental part of the design also eased the porting of code and contributed to good initial results for applications. The large memory on each node (128 megabytes in our case) meant that a first stage in the porting process could be achieved by running

Table 1 Parallel tools discussed in this paper

BlockSolve	Parallel sparse, symmetric linear systems
Chameleon	Lightweight and portable message-passing system
FORTRAN M	Parallel extensions to FORTRAN
MPI	Message-Passing Interface draft standard
Parallel UNIX tools	Parallel versions of cp, kill, etc.
PCN	Program Composition Notation (a coordination language)
PETSc	Portable Extensible Tools for Scientific Computing
p4	Portable message-passing and shared-memory library
SBR	Numerical parallel linear algebra
xsp linfo	Graphic display of EUIH message-passing system

Table 2 Applications discussed in this paper

Computational electromagnetics	Model 3D, arbitrary geometry magnets
Mesoscale weather model	Continent-sized weather model
Nuclear structure	Monte Carlo computation
Parallel community climate model	Global climate model
Phylogenetic trees	Program to construct phylogenetic trees from sequence data
Superconductivity	Modeling of flux vortices in high-temperature superconductors (three applications)
Theorem prover	Distributed associative-commutative theorem prover

the application sequentially on one node. On machines where the same aggregate memory and computing power is achieved with a larger number of smaller nodes, this is often impossible, and so applications must be ported in one step, which requires solving several problems at once.

Some features of the SP1 that contributed to our decision to acquire it turned out to be much less important than the above design aspects. This was particularly true of the new parallel software interface to the machine. The custom EUI message-passing library was not used by most applications. Since applications used portability libraries, only the tools developers needed to know the EUI interface. Environment tools such as the log file display tool (vt) and the parallel version of dbx (pdbx) did not play a role, since during most of the period described here, they did not run on 128-node machines. In addition, such tools were too tightly coupled to parts of the system we were not using. For example, no library for producing vt trace records came with the system, so vt could not be integrated into portable tools. Finally, the job-scheduling software (LoadLeveler*) was perhaps too completely an off-the-shelf component from the workstation network world; once the SP1 became heavily used (which was soon) we needed a scheduler more appropriate for a true supercomputer.

Programming packages and tools

This section describes both new tools developed for the SP1 and existing tools ported to the SP1, as well as several numerical libraries and programming packages that were ported to the new system.

Graphics tools. Researchers in Argonne's Mathematics and Computer Science Division (MCS) developed a tool, named xsp linfo, that displays the usage of the SP1 when using the EUIH message-passing system. This tool, written using tcl and tk, shows the partitions in use, each partition being displayed in a different color, and a list of partitions containing the username, size, and amount of time the partition has been in use. An example display is shown in Figure 2.

In addition, each node is represented by a button. Pushing this button with the mouse can bring up an x-terminal window (xterm) on that node or show the load average (depending on which mouse button is used). The load average reflects all processes, not just EUIH jobs. The display also shows the number of the node the mouse is pointing to and the time when the display was last updated. Using the tool xsp linfo, users can see how much of the machine is in use with EUIH jobs and see where their

Figure 2 A typical display of the EUIH usage of the SP1

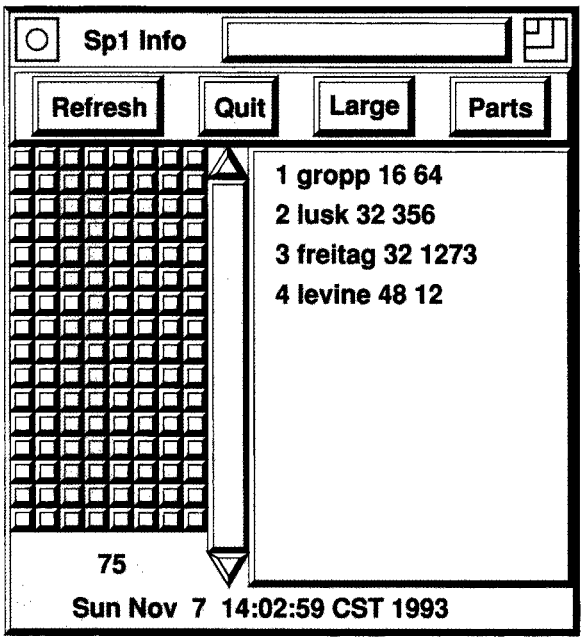


Table 3 Parallel UNIX commands

UNIX	Parallel
cp	pcp
ps	pps
ls	pls
find	pfind
if ('test') action	ppred
kill	pkill
a.out	prun . . . a.out

EUIH jobs are running. It also provides a convenient way to open x-terminal windows on the SP1.

Parallel UNIX tools. Because each node of the SP1 ran a separate copy of AIX and contained a private disk, users quickly discovered the need to (1) run various UNIX** tools (such as ps, ls, and cp) on a set of nodes and (2) to filter the output to show just the needed data. For example, on a uniprocessor, a typical query "Is a file present?" is usually answered by using ls filename. On the 128-processor SP1, however, running ls generated so much output that it was difficult to be sure that the file was present on all processors. (Piping into the word count program, wc, was too severe; if the file was missing somewhere, the user wanted to know where.) MCS scientists developed programs to answer such questions in a scalable way, by providing both an easy way to make a specific inquiry across the parallel machine (e.g., the program ppred) and a graphical display of the output of these parallel commands (the program pdisp).

Table 3 lists the prototype implementations of some parallel versions of popular UNIX commands. These routines (actually shell scripts) use recursive subdivision to execute the UNIX commands in parallel. They proved particularly important in distributing executable and shared input data to the local disks on the nodes. (This project, conceived as an immediate need on the SP1, subsequently developed into a more substantial effort.[3])

FORTRAN M. FORTRAN M[4] is a small set of extensions to FORTRAN that supports a modular approach to the construction of sequential and parallel programs. FORTRAN M programs use channels to plug together processes that may be written in FORTRAN M or FORTRAN 77. Processes communicate by sending and receiving messages on channels. Channels and processes can be created dynamically, but programs remain deterministic unless specialized nondeterministic constructs are used.

FORTRAN M was ported to the SP1, with run-time support added to allow communication via the Transmission Control Protocol/Internet Protocol (TCP/IP) over either the switch or the Ethernet. Few difficulties were encountered beyond the normal small differences in FORTRAN compilers. (Since FORTRAN M is a preprocessor that produces FORTRAN 77 code, it is heavily dependent upon the FORTRAN 77 compiler of the target machine.) Several FORTRAN M applications were also run on the SP1, including a parallel chromatography simulation and a parallel smog model.

Chameleon. Message passing is a common method for writing programs for distributed memory parallel computers. Unfortunately, the lack of a standard for message passing has hampered the construction of portable and efficient parallel programs. In an attempt to remedy this problem, a number of groups have developed their own message-passing systems, each with its own strengths and weaknesses. Chameleon[5] is a second-generation system of this type. Rather than replacing

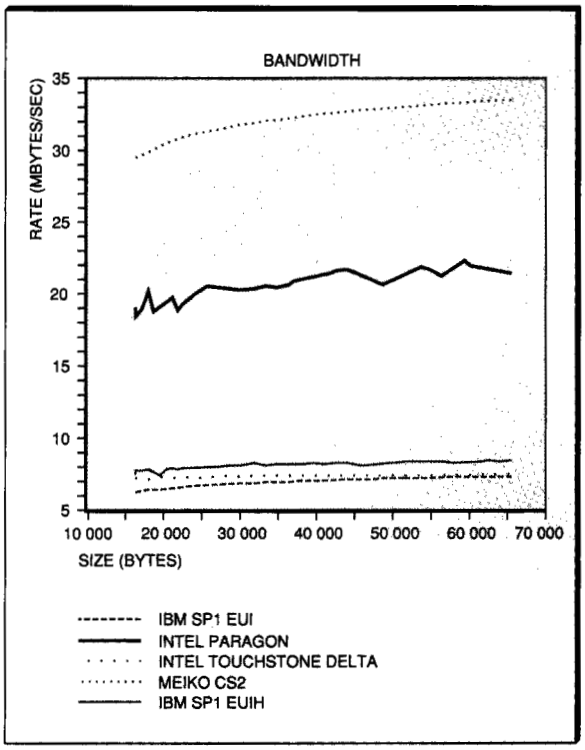
these existing systems, Chameleon is meant to supplement them by providing a uniform way to access many of these systems. Chameleon's goals are to (1) be very lightweight (have low overhead), (2) be highly portable, and (3) help standardize program startup and the use of emerging message-passing operations such as collective operations on subsets of processors. Chameleon also provides a way to port programs written using other message-passing libraries (such as PICL or Intel NX) to other systems, including collections of workstations. This feature was used by the global climate model (discussed later) to port to the SP1.

Chameleon was ported to the SP1 with no problems other than the need to statically link FORTRAN programs. Both an EUI and EUIH port were provided, as well as a TCP/IP port using the p4 message-passing system (discussed later) that uses TCP/IP. The EUIH port provided a simplified startup mechanism that eliminated the need for having the user invoke the program with the shell script cotb0.

Using a timing program in Chameleon called twin, MCS researchers tested communication between pairs of processors. The program selects message sizes adaptively in order to capture discontinuities in the behavior of the message-passing system. Figure 1 compares the performance on several different machines. The SP1 shows a distinct discontinuity at 128 bytes. This is due to the fact that (the preproduction version of) EUI switches to a different protocol for longer messages, thereby significantly adding to the latency of longer messages. The Intel Touchstone DELTA results also show a discontinuity (at 480 bytes); this reflects the message packet size (minus the header) used on the DELTA. The performance for long messages for a variety of machines is shown in Figure 3. These results show that one can expect good DELTA performance compared with other massively parallel processors for communication-intensive programs that use EUIH.

Successive band reduction. Successive band reduction (SBR) is an approach for orthogonally reducing matrices to condensed form, thereby allowing for the use of matrix-matrix (BLAS-3) kernels. Special instances of SBR are the tridiagonal and Hessenberg reductions used in various eigensolvers. In addition, SBR supports general band reduction, which is needed for banded eigenvalue scenarios. A modified SBR approach is also used in the PRISM (parallel research on invariant subspace methods)

Figure 3 Communication performance for long messages for a variety of machines. All SP1 results use the switch. Data for the CM5 is unavailable for messages of this length.



project for the development of a scalable parallel eigenvalue solver. The SBR code is intended for distributed memory multiple-instruction, multiple-data (MIMD) parallel machines. PRISM uses the Chameleon programming system, discussed in the previous section, and support for parallel operations on disjoint node subsets that is critical in exploiting the multiple levels of parallelism in the algorithm.

Table 4 shows some early performance results of the SBR code when applied to the Hessenberg reduction of a full matrix. N is the matrix size, nb is the block size used in the reduction of the matrix to b subdiagonal bands, Time is the elapsed time in seconds, and Gflops (gigaflops) is the sustained double-precision performance in billions of floating-point operations per second on 16 nodes of the SP1. These runs used the EUIH transport layer and the vendor-supplied blas.a library. As the table indicates, codes exploiting matrix-matrix ker-

Table 4 Results for successive band reduction (SBR) code applied to a full matrix

Matrix Size (N)	Block Size (nb)	Subdiagonal Bands (b)	Time	Gflops
1000	1	20	19.9	0.233
1020	15	15	11.5	0.412
2000	1	20	164.1	0.227
2000	10	10	63.1	0.582
3000	1	10	512.5	0.246
3000	15	15	165.4	0.798
4000	20	20	346.0	0.849

nels ($nb > 1$) perform much better than codes based on matrix-vector kernels ($nb = 1$). In particular, using a blocked algorithm, one can reduce a 3000×3000 matrix to a bandwidth of 15, roughly in the same time that one can tridiagonalize a 2000×2000 matrix, even though the former operation takes three times as many floating-point operations.

Message-Passing Interface. Message-Passing Interface (MPI) is a message-passing standard library interface developed by a group of parallel computer vendors and users. A partial implementation of the standard as of May 1993 was implemented and run on the SP1. Subsequently, the MPI standard was completed (see References 6, 7), and multiple versions were made available on the SP1. All made efficient use of the High-Performance Switch and were comparable to IBM's fastest message-passing library products. A portable version that allows porting from a very large number of machines and workstation networks to the SP1 is described in Reference 8.

Program Composition Notation. Program Composition Notation (PCN)[9,10] is a system for developing and executing parallel programs. It comprises a high-level programming language, tools for developing and debugging programs in this language, and interfaces to FORTRAN and C that allow the reuse of existing code in multilingual parallel programs.

The network version of PCN (net-PCN) was ported to the SP1. Run-time support was added to PCN to allow communication to use TCP/IP over either the SP1 High-Performance Switch or the Ethernet. Several PCN applications were also run on the SP1, including the massively parallel mesoscale model (discussed in a later section).

Libraries for scientific computing. Two libraries that were designed at Argonne to assist in the development of parallel scientific software were ported to the SP1.

Portable Extensible Tools for Scientific Computing. Portable Extensible Tools for Scientific Computing (PETSc) is a package of routines aimed primarily at the solution of partial differential equations. PETSc is designed to match advanced algorithms to new and existing applications by taking an object-oriented approach to the design of the routines. For example, the iterative accelerators that are part of PETSc[11] allow the user to specify all of the vector operations as well as matrix-vector product and preconditioning. Thus, these iterative methods can be used with nontraditional vectors, such as vectors generated by adaptive refinement algorithms, or vectors distributed across a distributed memory parallel computer. PETSc also includes a number of packages that aid in writing parallel programs. One of these is BlockComm, a package for communicating blocks of data between processors. Another is a parallel general (nonsymmetric) linear system solver using iterative methods.[11]

All of the parallel communication in PETSc is done with Chameleon; hence, porting PETSc required no special effort, with the exception of the FORTRAN library. A version of PETSc that can take advantage of the IBM Engineering and Scientific Subroutine Library (ESSL) was developed; because PETSc is object oriented, users can take advantage of these changes by relinking rather than rewriting code.

BlockSolve. BlockSolve[12] is a software library of applications used for solving large, sparse systems of linear equations on massively parallel computers. The matrices must be symmetric but may have an arbitrary sparsity structure. BlockSolve is a portable package that is compatible with several different message-passing paradigms including EUIH but not EUI.

Researchers used Chameleon to port BlockSolve easily to the IBM SP1. (It should be noted, however, that BlockSolve need be compiled only with the options compatible with the message-passing paradigm and architecture on which it will be used.)

The p4 system. The p4 parallel programming system[13,14] is a library of message-passing applications that runs on a wide variety of parallel computers and workstations and has been used routinely on networks of RISC System/6000 machines.

The C part of p4 compiled and linked without alteration on the SP1, using all parameters from the RISC System/6000 version. Programs written in C that had been compiled and linked for the RISC System/6000 network using p4 also ran unchanged; the phylogenetic tree application (see section later in this paper with same name) is in this category.

After the switch and related software were installed, existing p4 applications were able immediately to use the switch via the TCP/IP interface, even without EUI. When EUI was available, p4 was quickly ported to EUI and then to EUIH. The only changes necessary to move from the EUI version involved switching to FORTRAN calling sequences.

Applications

Successful porting of a programming package to a parallel machine was once considered a sufficient test of the machine. However, as parallel machines are increasingly being acquired for production computing, it is more important to test them with real existing (as opposed to model) applications. Using the portability tools described previously, MCS researchers quickly ported and ran a wide variety of applications.

Computational electromagnetics. Computational electromagnetics is widely used in industrial, research, and defense applications. However, many important problems are intractable with conventional techniques and vector supercomputers. For practical applications, the problem size (such as the number of degrees of freedom and the number of unknowns) must be dramatically increased, turnaround time must be reduced, and solution accuracy must be improved. A promising approach for overcoming these limitations is the use of integral equation methods (IEMs) implemented on massively parallel computers.

CORAL is a program that has been used to solve nonlinear three-dimensional magnetostatics problems by exploiting such methods. Argonne researchers developed a parallel version of CORAL, using Chameleon for the message-passing parts of the program. The systems of linear equations were solved using the parallel iterative methods in Argonne's Parallel Simplified Linear Equation Solvers package, which provides easy access to state-of-the-art methods for solving systems of linear equations (see the previous section on libraries for scientific computing).

Table 5 Solution time (in seconds) for four problems

Number of Processors	Matrices of Order			
	579	972	1629	2278
1	952	3127	—	—
2	507	1245	—	—
4	279	767	2571	—
6	236	—	—	—
8	—	454	1637	3416

Preliminary timings on the IBM SP1 are shown in Table 5. The table shows solution time in seconds as a function of the number of processors for four different problems. The first column is the number of processors used. The other columns report total solution time (in seconds) as a function of the number of processors used for solving four different nonlinear problems with matrices of order 579, 972, 1629, and 2278, respectively. These results show good performance and speedup even on relatively small problems.

Massively parallel mesoscale model. Mesoscale models of the atmosphere provide short-range weather forecasts, detailed "what if" scenarios for climate change, and research platforms for modeling the generation of storms. One widely used mesoscale model is the massively parallel mesoscale model (MPMM), a fine-grained dynamic decomposition of the Penn State NCAR (National Center for Atmospheric Research) mesoscale model version 5. In MPMM, each set of four horizontal grid points is represented as a parallel process running under PCN (discussed previously), providing a transparent mechanism for redistributing load between physical processors.

MPMM achieved 700 Mflops (millions of floating-point operations per second) on 64 processors of the IBM SP1. This performance is equivalent to five times the performance of a single CRAY Y-MP** processor. The SP1 offered several features critical to efficient performance for real-time forecasting and data assimilation—in particular, the high-bandwidth input/output and the large memory capacity

on the processing nodes. The SP1 also provided a better development environment than other parallel systems, in part because each node has a full UNIX environment.

Monte Carlo calculations of nuclear ground states. Researchers in Argonne's Physics Division, in collaboration with V. R. Pandharipande of the University of Illinois at Urbana, are computing the properties of light (up to 40 neutrons and protons) nuclei using realistic two- and three-nucleon interactions. This research involves developing many-body methods for reliably computing the properties of a nucleus for complicated forces that are strongly dependent on the spins and charge states of the nucleons.

Argonne researchers quickly ported and ran a wide variety of applications.

The current approach involves variational calculations in which one assumes a form for the quantum-mechanical wave function describing a nucleus and then computes the energy of the nucleus for a given force model. Multidimensional (12 to 120 dimension) integrals are computed by using Monte Carlo methods. The integrand is expressed in terms of large complex vectors describing the spin and charge states of the nucleons. These calculations must be repeated many times to find the best set of parameters for the assumed form of the wave function. The longer a given calculation is allowed to proceed, the smaller the statistical error from the Monte Carlo integration, and hence the more refined the determination of the best parameters.

The first calculations done on the SP1 used a new nuclear interaction and obtained much better (when compared with experiment) results for the binding energy and density profile of oxygen than previously obtained. The better density results were specifically attributable to the detailed variational searches made possible by the SP1. The speed of the SP1 also enabled the researchers to calculate calcium (40 nucleons) for the first time.

The SP1 was the first parallel processor used for these calculations. Previously, the work had been performed on single processors of the most powerful Cray computers available. Earlier parallel computers could not be used because of their small memories; the calculations require up to 65 megabytes of memory.

The Argonne package p4 was used to implement the message-passing part of the program. Runs on 128 nodes achieved speedups of 123, or computational rates of 5.9 Gflops. (One run using 160 nodes achieved 6.5 Gflops, but there were also other users on some of the nodes.)

Parallel community climate model. The parallel community climate model (PCCM2) is a message-passing implementation of the NCAR (National Center for Atmospheric Research) Community Climate Model 2 and is intended for global climate prediction. Development of the parallel model is supported by the Department of Energy CHAMMP program as a collaborative project with Argonne, Oak Ridge, and NCAR.

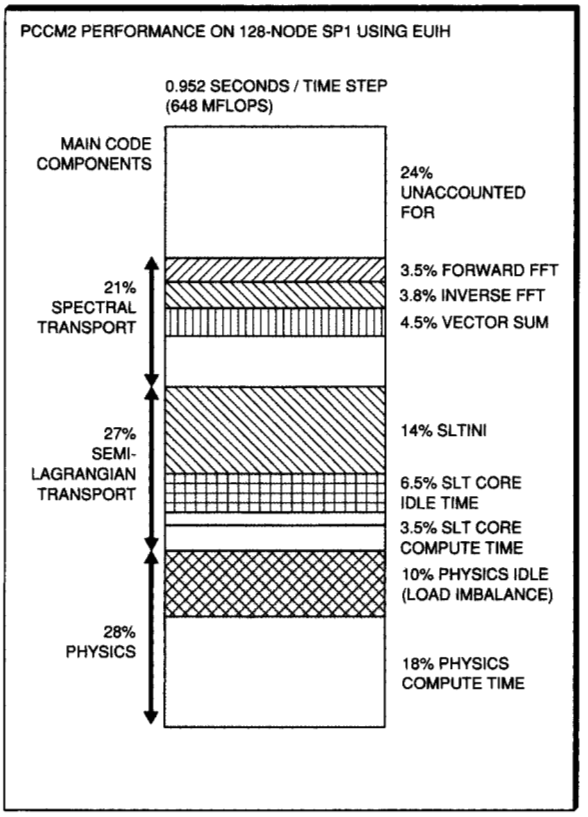
In September 1993, PCCM was officially validated with respect to the sequential version of the community climate model. The SP1 was used extensively in the validation work because its nodes are identical to workstation platforms running the previously validated sequential version.

The model is patch decomposed in two horizontal dimensions. Spectral transport of all prognostic variables except moisture is accomplished by parallel fast Fourier transforms (FFTs) in the zonal dimension and Gaussian quadrature in the meridional dimension, approximating Legendre transforms. The spectral transport mechanism of PCCM2 is communication-intensive because interchange of data is not confined to the nearest neighbor. A semi-Lagrangian transport scheme is used for transport of moisture. Modules that compute atmospheric processes such as convection, radiative transfer, and precipitation are collectively known within the model as physics. Physics is perfectly parallel in PCCM because there are no horizontal data dependencies; however, physics does present the largest source of inefficiency from load imbalance.

PCCM2 was implemented by using Chameleon through a compatibility library to PICL, the message-passing package under which the code was originally developed. Before being run on the IBM SP1, PCCM2 was run on the Intel Touchstone DELTA and Paragon computers.

PCCM ran at approximately 650 Mflops on the full SP1 (128 processors) communicating over the EUIH switch interface. Figure 4 shows the distribution of run time over the three main components of the code: spectral transport, semi-Lagrangian transport, and physics. Interprocessor communication accounted for most of the time spent in the forward and inverse FFTs, the parallel vector-sum (part of the Legendre approximation), and initialization for the semi-Lagrangian transport (SLTINI). The largest computational part of the code is physics, and the effect of load imbalance can be seen as well. Despite good overall floating-point performance (.65 Gigaflops), a considerable fraction of the time was still taken up with communication and load imbalance possibly due in part to the absence of gang scheduling on the SP.

Figure 4 Distribution of work in a PCCM2 environment



Phylogenetic trees. To better understand organisms and the evolution and function of the ribosome, Argonne researchers are developing a phylogenetic tree that relates the sequences of a small molecular subunit called ribosomal RNA in prokaryotic and eukaryotic microorganisms. This work is conducted in collaboration with scientists from the University of Illinois Ribosomal Database Project. A maximum likelihood technique is being used, which determines the greatest statistical probability of the sequence of evolution. The technique, originally implemented by Felsenstein,[15] was enhanced by University of Illinois researchers, along with researchers at Argonne and Kobe University in Japan, to enable it to analyze large numbers of organisms on both uniprocessors and massively parallel systems.

Using the enhanced technique, and capitalizing on the large memory and exceptional speed of the system, the researchers were able to construct a phylogenetic tree with more than 2000 organisms. Since it was not practical to simply make one huge run and produce a reliable tree, the tree was computed in steps. Initially, a small tree of 473 organisms was composed. New sequences were then inserted into the tree, and the resultant tree was "optimized" by performing thousands of local maximum likelihood computations (which can produce local rearrangements within the tree).

Argonne researchers also used the SP1 to gather data needed to evaluate tools used in conjunction with the maximum likelihood computation. These tools, developed at the University of Illinois, compute varying rates of change that occur in phylogenetic computations. Over 450 runs were made on the SP1, each of which consumed between 12 and 36 hours on single nodes.

The phylogenetic tree application unquestionably consumed the largest number of hours on the SP1 during its first few months at Argonne. The work was reported in the *Journal of Bacteriology*,[16] with a full reproduction of the tree.

Superconductivity. Argonne is exploring various aspects of the numerical solution of high-temperature superconductors. Three projects made early

Table 6 Solution times (seconds per iteration) for three problems

Number of Processors	Grid Sizes $130^2 \times 10$		$258^2 \times 10$	$514^2 \times 10$
	EUIH	Sockets	EUIH	EUIH
1	7.67	15.3	—	—
2	4.00	8.19	—	—
4	2.16	4.88	8.36	—
8	1.22	3.89	4.66	18.67
16	.60	3.69	2.19	8.27
32	.34	4.11	1.18	4.43
64	.20	4.36	.65	2.23
96	.17	—	.48	1.57

use of the SP1: the study of the vortex-glass state, numerical simulation of vortex dynamics, and analysis of the properties of vortex structures.

Elastic string model. MCS Division computer scientists developed a code for the numerical simulation of the planar motion of a one-dimensional elastic filament (single vortex) under tension. With this elastic filament model, avalanche-type behavior was observed when the applied forces were in the neighborhood of a critical transition value. The researchers developed other elastic filament models to explore this so-called self-organized criticality phenomenon. Numerical studies required the accumulation of statistics from a large number of events, each involving the solution of a stochastic differential equation subject to a random initial perturbation.

The most difficult calculations occurred for very small applied forces when the system was in a "glassy" or "creep" state. The slow dynamics of the creep state required extremely large amounts of computer time to establish the asymptotic behavior. The calculations were characterized by large numbers of ensembles (~12 000), each corresponding to a random realization of a potential tilted by an applied force. Even for small forces, a very large number of spatial points (~75 000) were needed to resolve the potentials. Since each realization is independent of the other realizations, a large number of these jobs was able to be run in parallel. This work was initially started using a BBN TC2000** and a Sun SPARCstation** network. The programs were then ported to the SP1 system, realizing a significant improvement in execution time. In particular, for some of the most difficult calculations, with very small applied forces, use of the SP1 reduced solution time from approximately five days (on Sun SPARC workstations) to approximately 17 hours.

Vortex dynamics. To study the formation and subsequent evolution of magnetic flux vortices and the influence of random impurities on vortex pinning, MCS researchers developed a computer model based on the three-dimensional time-dependent Ginzburg-Landau (TDGL) equations.

The three-dimensional domain is subdivided into an array of cells. The code solves numerically for the complex-valued order parameter (identified with the vertices of each cell) and the gauge field (identified with the edges of each cell). The resulting equations of motion are solved by using a single-time-step forward Euler procedure. The primary data structures used are four complex, three-dimensional arrays whose values are updated each time step according to the equations of motion.

The dynamics were initiated from a doped state in the presence of an external magnetic field whose strength was adjusted so that the material was in the mixed state. The material was modeled with a pair of planar defects (twin boundaries) running diagonally through the sample. Inhomogeneities in the twin boundaries were modeled with random point defects embedded in the twin boundaries.

To parallelize the program, the researchers partitioned the array of cells (grid) among the processors. Each processor was responsible for updating all the cells in the subgrid contained in its memory. The update step for each cell required values from neighboring cells. In order to communicate these values between processors, the BlockComm package (previously discussed) was used.

Table 7 Comparison of the CPU and elapsed times for three different cases

Number of Processors	Intel DELTA		IBM SP1 (EUIH)		IBM SP1 (p4)	
	CPU	Elapsed	CPU	Elapsed	CPU	Elapsed
1	—	—	203.89	205.46	203.90	205.29
2	307.67	308.00	86.09	86.75	81.26	118.44
4	160.26	160.00	37.54	37.61	33.29	112.46
8	79.33	80.00	21.33	21.50	17.94	196.09
16	43.13	43.00	12.75	12.97	—	—

Preliminary timings on the IBM SP1 are shown in Table 6. The first column is the number of processors used. The other columns report time per iteration (in seconds) as a function of the number of processors for grid sizes of $130 \times 130 \times 10$, $258 \times 258 \times 10$, and $514 \times 514 \times 10$, respectively. EUIH in a column indicates IBM's EUIH message-passing software was used in conjunction with the High-Performance Switch. For comparative purposes, the smallest problem was also run using a version of the p4 message-passing software that used UNIX sockets to communicate over an external Ethernet network that also connects the processors.

Vortex structures. To find optimal vortex solutions within the three-dimensional anisotropic Ginzburg-Landau model, MCS researchers developed a parallel code that uses the limited-memory BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm.[17]

Parallelization was achieved through a simple three-dimensional domain decomposition scheme in which the global domain was partitioned across an arbitrary number of processors. The communication between processors was carried out by using the Chameleon parallel software package. The portability of the Chameleon primitives enabled the code to be run on a variety of parallel platforms, using several different parallel communication paradigms, without any coding changes.

The code was used to study various properties of uniaxial superconductors such as the lower critical field and the anomalous "vortex-chain" state. Tables 7 and 8 give performance comparisons for a selection of three different cases: (1) the Intel Touchstone DELTA, (2) the IBM SP1 running EUIH, and (3) the IBM SP1 running p4. All cases are for 100 BFGS iterations with a constant *global* domain size of $32 \times 32 \times 32$ for Table 7, and a constant *local* domain size of $16 \times 16 \times 16$ for Table 8. The superlinear speedup in the SP1 results in Table 7 was most likely caused by cache effects. Also note the CPU time column from the SP1 (p4) results; these show very good performance in a time-shared environment, even though the elapsed time performance is relatively poor. Table 8 shows the performance as the local domain size is held fixed and the number of unknowns grows proportionally with the number of processors. The results suggest that the local domain is too small (only 4096 mesh points) for the SP1. This conclusion is consistent with the faster speed of the processors with respect to the communication than for the Intel DELTA, and emphasizes why the large per-node memory is an important feature of the SP1.

Table 8 Comparison of the CPU and elapsed time for two cases

Number of Processors	Intel DELTA		IBM SP1 (EUIH)	
	CPU	Elapsed	CPU	Elapsed
1	73.71	74.00	15.91	16.01
2	76.27	76.00	17.71	17.90
4	77.85	78.00	19.33	19.49
8	79.33	80.00	21.46	21.57
16	80.57	81.00	22.75	22.98

Parallel theorem prover. A parallel distributed memory theorem prover (the distributed associative-commutative theorem prover, or dac) was ported to the SP1 with no problems. The code had initially been developed on the Sequent Computer Corporation Symmetry system and a network of Sun Microsystem processors using p4. Therefore, once p4 itself was ported to the SP1, dac simply had to be recompiled and linked.

The code was used on the SP1 to solve a benchmark problem in theorem proving, namely, that a ring where $x^3 = x$ for all x is commutative. Single-node performance was excellent, despite the lack of floating-point operations in dac.

Figure 5 Bandwidth for an SP1 with SP2 communication adaptors

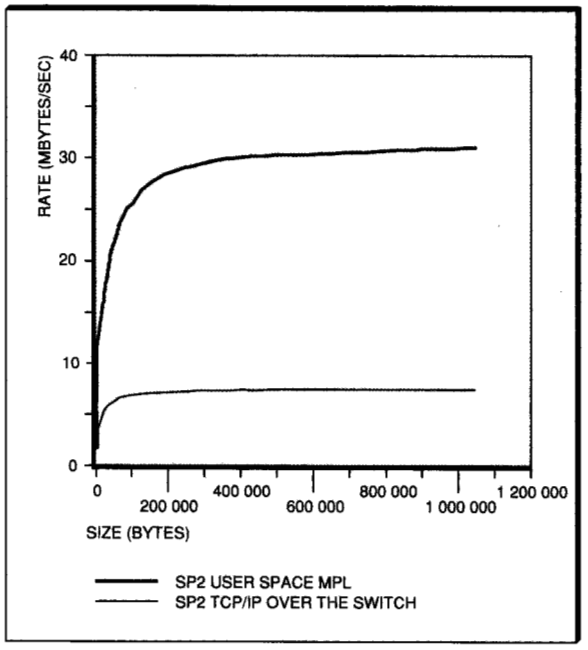
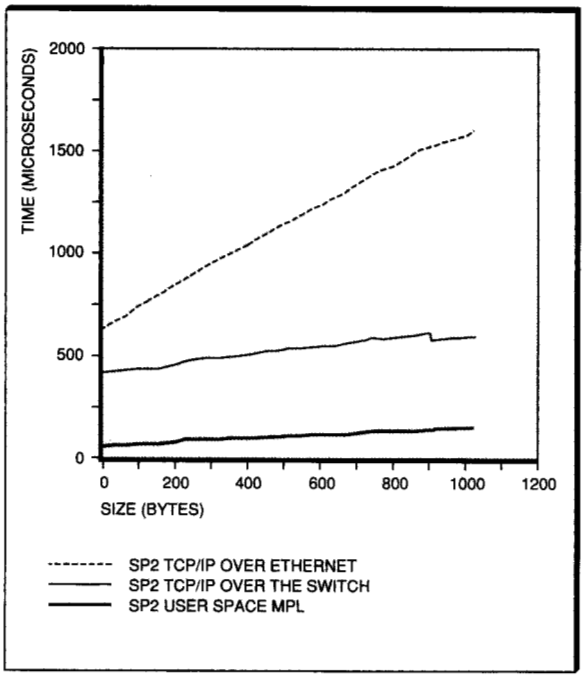


Figure 6 Latencies for small messages on SP1 with SP2 communication adaptors



Summary

The SP1 proved to be an excellent system for a variety of applications. Having a full version of UNIX on each node allowed for easy ports. The large memory on each node provided one-node execution as a useful step in the porting process. Furthermore, the use of tools such as Chameleon and p4 made it possible to port diverse applications quickly.

After the first few months, the pace of moving new applications to the machine increased. Many of the early applications quickly shifted to "production" mode, no longer functioning as research projects in computing but delivering new scientific results in their application areas. The work in structural biology and in nuclear physics is particularly noteworthy.

Since the time covered by the original reporting of these experiences in an Argonne technical report,[18] development of the SP1 facility at Argonne and the programming environment has flourished. In terms of hardware, the biggest change has been the installation of the SP2 communication adaptors, allowing much greater bandwidth (see Figure 5). We now run the SP2 software although we still have SP1 nodes. Figure 6 shows latency for short messages on the upgraded machine. The architecture has proven flexible enough to accommodate custom changes, such as integration of our Model 970Bs, large RISC System/6000s originally designed to be connected only by fiber channel and Ethernet to the SP1 nodes, into the switch "fabric" by installing SP2 communication adapters in them. (This makes our machine a sort of "home-brew" SP2.) The SP2 communication adapters are faster (up to five times faster) and "smarter" (such as allowing overlap between communication and computation using direct memory access) than the SP1 communication adapters.

This new release 2 implementation of MPL has replaced EUIH, providing high performance in the product software, but those using the portability libraries did not see any changes in their codes. Since the time covered by our earlier experiences, the MPI (Message-Passing Interface) standard[7] has emerged as an industry-wide portable library specification, and most new applications at Argonne are being written in MPI. Two implementations of MPI

are available to users on our SP1. Both are as efficient as the "native" MPL library.

Utilization of the machine has been enhanced by a new job scheduler developed at Argonne,[19] which ensures exclusive access to nodes for users for enhanced parallel performance. Some of its features are being considered for later versions of the IBM scheduling software.

Finally, visibility of the machine and its documentation have been enhanced by making use of the World Wide Web in the Internet. Current information on our machine and all aspects of its operation can be found at http://www.mcs.anl.gov/sp1.

Acknowledgments

Many people contributed to the individual sections of this document. We thank in particular Christian Bischof, Kimmo Forsman, Lori Freitag, Lauri Kettunen, Gary Leaf, David Levine, William McCune, John Michalakes, Ross Overbeek, Mario Palumbo, Steven Pieper, Paul Plassmann, Xiaobai Sun, and Steven Tuecke.

Cited references

1. C. B. Stunkel, D. G. Shea, B. Abali, M. G. Atkins, C. A. Bender, D. G. Grice, P. Hochschild, D. J. Joseph, B. J. Nathanson, R. A. Swetz, R. F. Stucke, M. Tsao, and P. R. Varker, "The SP2 High-Performance Switch," *IBM Systems Journal* **34**, No. 2, 185–204, (1995, this issue).
2. M. Snir, P. Hochschild, D. D. Frye, and K. J. Gildea, "The Communication Software and Parallel Environment of the IBM SP2," *IBM Systems Journal* **34**, No. 2, 205–221 (1995, this issue).
3. W. Gropp and E. Lusk, "Scalable UNIX Tools on Parallel Processors," *Proceedings of the 1994 Scalable High Performance Computing Conference*, IEEE (1994), pp. 56–62.
4. I. Foster, R. Olson, and S. Tuecke, *Programming in Fortran M*, Technical Report ANL-93/26, Revision 1, Argonne National Laboratory (1993).
5. W. D. Gropp and B. F. Smith, *Chameleon Parallel Programming Tools Users Manual*, Technical Report ANL-93/23, Argonne National Laboratory (March 1993).
6. The Message-Passing Interface Forum, "MPI: A Message Passing Interface," *Proceedings of Supercomputing '93*, IEEE Computer Society Press (November 1993), pp. 878–883.
7. The Message-Passing Interface Forum, "MPI: A Message-Passing Interface Standard," *International Journal of Supercomputer Applications* **8**, No. 3/4 (1994).
8. W. Gropp and E. Lusk, *An Abstract Device Definition to Support the Implementation of a High-level Message-Passing Interface*, Technical Report MCS-P342-1193, Argonne National Laboratory (1993).
9. I. Foster, R. Olson, and S. Tuecke, "Productive Parallel Programming: The PCN Approach," *Scientific Programming* **1**, No. 1, 51–66 (Fall 1992).
10. I. Foster and S. Tuecke, *Parallel Programming with PCN*, Technical Report ANL-91/32, Rev. 2, Argonne National Laboratory (1991).
11. W. D. Gropp and B. F. Smith, *Users Manual for KSP: Data-Structure-Neutral Codes Implementing Krylov Space Methods*, Technical Report ANL-93/30, Argonne National Laboratory (August 1993).
12. M. T. Jones and P. E. Plassmann, "An Efficient Parallel Iterative Solver for Large Sparse Linear Systems," *Proceedings of the International Mathematics Association Workshop on Sparse Matrix Computations: Graph Theory Issues & Algorithms*, Minneapolis, MN, University of Minnesota (1991).
13. R. Butler and E. Lusk, "Monitors, Messages, and Clusters: The p4 Parallel Programming System," *Journal of Parallel Computing*. To appear.
14. R. Butler and E. Lusk, "Monitors, Messages, and Clusters: The p4 Parallel Programming System," *Parallel Computing* **20**, 547–564 (1994).
15. J. Felsenstein, *Phylip Manual Version 3.3*, University of California, Berkeley, CA 94720 (1990).
16. G. L. Olsen, C. R. Woese, and R. Overbeek, "The Winds of (Evolutionary) Change: Breathing New Life into Microbiology," *Journal of Bacteriology* **176**, No. 1, 1–6 (January 1994).
17. D. C. Liu and J. Nocedal, "On the Limited Memory BFGS Method for Large-scale Optimization," *Mathematical Programming* **45B**, 503–528 (1989).
18. W. Gropp, *Early Experiences with the IBM SP1 and the High-Performance Switch*, Technical Report ANL-93/41, Argonne National Laboratory (1993).
19. D. Lifka, M. Henderson, and K. Rayl, *Users Guide to the ANL SP Scheduling System*, Technical Report ANL/MCS-TM-201, Argonne National Laboratory (1994).

Accepted for publication December 19, 1994.

William D. Gropp *Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, Illinois 60439-4844 (electronic mail: gropp@mcs.anl.gov).* Dr. Gropp is a computer scientist in the Mathematics and Computer Science Division at Argonne National Laboratory and Deputy Scientific Director of Argonne's High-Performance Computing Research Facility. After receiving his Ph.D. in computer science from Stanford University in 1982, he held the positions of assistant (1982–1988) and associate (1988–1990) professor in the Computer Science Department of Yale University. In 1990, he joined the Numerical Analysis group at Argonne. His research interests are in adaptive methods for PDEs, software for scientific computing, and parallel computing. His current projects include an implementation of the Message-Passing Interface standard, domain decomposition techniques for solving PDEs, and research into programming models for parallel architectures.

Ewing Lusk *Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne,*

Illinois 60439-4844 (electronic mail: lusk@mcs.anl.gov).
Dr. Lusk is a senior computer scientist in the Mathematics and Computer Science Division at Argonne National Laboratory and Scientific Director of Argonne's Advanced Computing Research Facility. After receiving his Ph.D. in mathematics at the University of Maryland in 1970, he served first in the Mathematics Department and later in the Computer Science Department at Northern Illinois University before joining the Automated Reasoning group at Argonne in 1982. His research interests are in automated theorem proving, logic programming, and parallel computing. His current projects include an implementation of the Message-Passing Interface standard and research into programming models for parallel architectures.

Reprint Order No. G321-5567.