How Sun's protocols shape networks, after p.80

AdvancedSystems

Client/Server Products for Unix Professionals

July 1994 $3.95

Vector Monster

News

Teaching objects to fetch data

Raidtec breaks $1,000 RAID barrier

Applix Real Time

MapInfo 3.0

Reviews

Cadam goes Unix

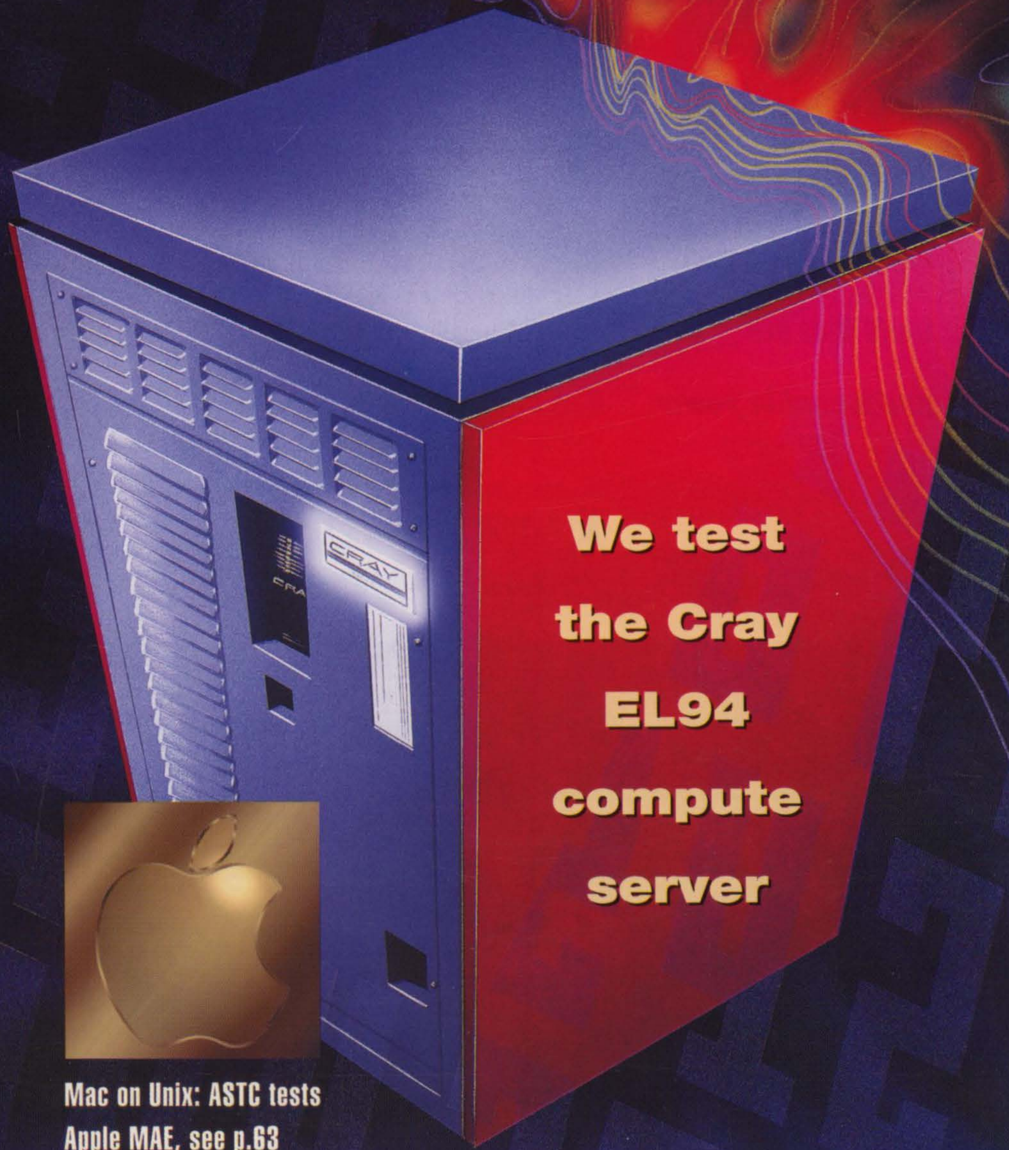
Big backup tools for big backup jobs

Serial ports galore

Technology

Languages that just will not die

Client/server: Relational objects?



We test the Cray EL94 compute server

Mac on Unix: ASTC tests Apple MAE, see p.63

Cray's affordable vect

By David Burnette

While the deskside EL94's hardware offers megaflops galore, it's Cray's industrial-strength code-development tools that are unequaled in power.

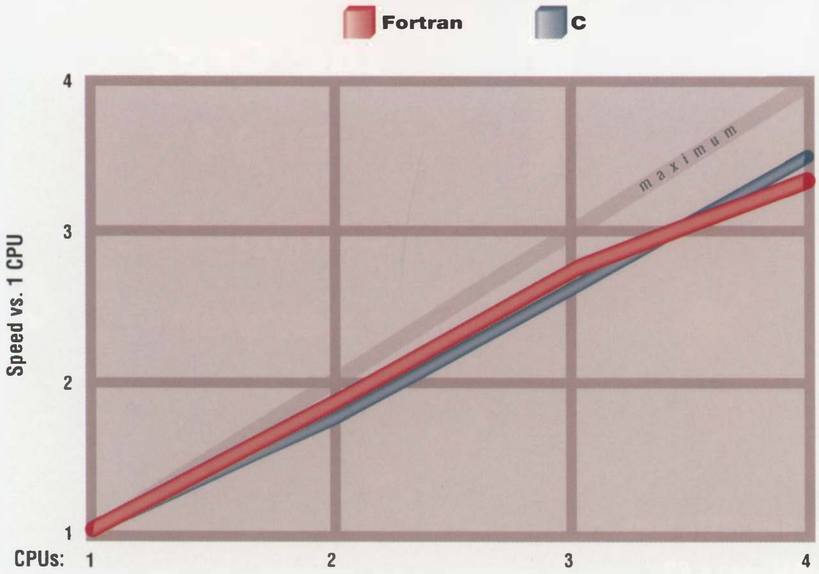
When Cray Research decided to enter the departmental server market, it didn't mess around. Its EL94 deskside supercomputer dwarfs Sun's popular SPARCserver 1000 in pounds, cubic feet, and BTUs. Whereas Sun's puny 1000 contains a duet of screaming fans, Cray's EL94 houses a half-dozen belly-mounted mini-turbines that threaten to elevate loose floor tiles while cooling its seven internal 3-gigabyte disk drives, 512 megabytes of RAM, and four CPUs. The steel-plated computational juggernaut throbbed in a corner of our lab for six weeks as we put it through its paces.

The EL94 is a different class of machine than what we're used to seeing here at the Advanced Systems Test Center. Over the last year we've reviewed a handful of what we considered large Unix servers from the likes of Digital, NCR, SGI, IBM, and Sun. These departmental servers were essentially bloated workstations, tricked out with disks, memory, and the I/O throughput necessary to tackle the enterprise job load. At their heart, though, were RISC processors wailing away at a pitch as high as 200 MHz and swaddled with megabyte caches.

The EL94's clock is a mere 33 MHz and it has no cache. Don't laugh. Its four processors pack a tremendous wallop if fed properly. Sporting four floating-point vector units each, naturally, their preferred diet is vectors and lots of them. The EL94 chewed its way through one problem in 36 minutes, when properly vectorized, while a two-CPU SPARCstation 10 clone required nearly 13 hours. We'll cover more performance data later in the article.

It's interesting to see the evolution of Cray's machines. The Cray-1, introduced in the late '70s, had a peak performance of 160 megaflops, almost a quarter of the EL94's theoretical maximum. The C916, Cray's flagship system, has a theoretical maximum of 15½ gigaflops. The EL series sits comfortably above antiquity on the performance curve and delightfully below the C916 on the price curve.

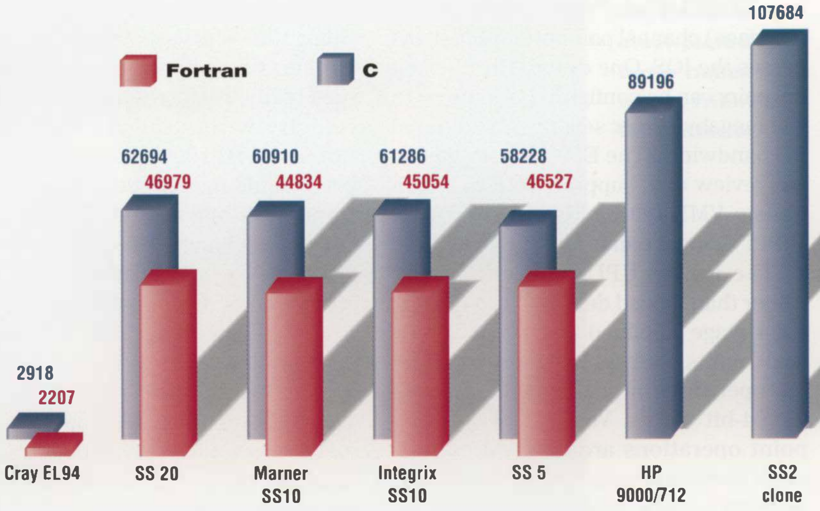
Mersenne speed up results



'Honey, I bought a Cray'

We knew the EL94 arrived when we received a call from our building's lobby saying that a 600-pound crate was sitting on the loading dock. We gave an elevator a run for its money and freighted it up four flights to the ASTC. Alas, its bulk wouldn't fit through our door so we uncrated it in the hall, to the amazement of our neighbors as they witnessed the unveiling of what might appear to some as a gray-and-red milking machine (the unit's Wisconsin heritage may have something to do with its appearance). The steel banding sprang away, the top came off, and we got our first look at the baby Cray, swathed in a polyethylene amniotic sack. Like fathers everywhere, we looked beyond its squat oafishness and saw only raw potential and supercomputing legendary — its nameplate proudly proclaimed the EL94 to be a product of Cray Research Inc., known for building the most powerful computers in the world.

or monster



Mersenne prime benchmark results

Completion times in seconds

The EL94 rolled weightily off its pallet, down a ramp, and into our lab with a Cray engineer in tow. He set immediately to slapping it on its rear and making sure all the parts were in the right place. It wasn't long before we heard the cry of six fans and seven disk drives surge to life, fed by 220 volts of single-phase power. Exhaling a steady breath of warm air through louvers in its front and back panels, and with a Wyse console terminal perched atop a flat expanse, the EL94 ran a series of internal self-checks and booted up Unicos 7.0, Cray's version of System V-derived Unix with BSD extensions (Unicos 8.0 will be out when you read this). It beckoned as a node on our lab network a few minutes later.

Since the ASCII console terminal offered little in the way of a GUI, we preferred accessing the EL94 from our workstations. All the familiar goodies were there: *xterm*, *rlogin*, *rcp*, and others. We were pleasantly surprised to find how quickly we felt at home in the fields of Cray.

This is not to say Unicos is identical to vanilla System V with BSD extensions. When adding new users we were greeted with our first indication that the system came from a foreign land: accounting groups. Cray's machines possess an abundance of utilities to track CPU usage, limit the amount of CPU time consumed by user processes, bill users according to the amount of CPU time used, and dole out system privileges to select groups of users. Accounting groups are a means to classify users based on their allowable workloads. These sorts of mainframe disciplines reminded us of the machine's ancestry.

Pencils up

We freely admit supercomputers are as new to us as they probably are to most RISC jockeys. Cray provided us with several benchmarks developed specifically to test the capabilities of parallel vector machines, which are a different sort of beast entirely from the usual lot of Unix systems that parade through the ASTC. Before we launch into a discussion of the Cray's performance results, please pardon us while we don our pedagogue's cap. The anatomy of an EL94 CPU deserves some description, for it explains why it can do so much with only a 33-MHz clock.

The system cabinet houses the main CPUs, disk drives, memory, and the I/O subsystem. Each pair of processors resides on a single module, from which there are four 266-megabyte-per-second ports into memory. The first two are for loads from memory to the CPU, the third is for stores to memory from the CPUs, and the fourth is for I/O. The four-processor EL94 thus has a total of eight 266-megabyte-per-second ports from memory, of which six (for a total bandwidth of 1,596 megabytes per second) go to the CPUs and two (for a total bandwidth of 532 megabytes per second) are dedicated to I/O.

The EL94's memory is built from 70-nanosecond DRAM chips, and thus isn't the lickety-split sort that goes into a workstation's L2 cache. One of our memory bandwidth tests yielded a value of 1.2 gigabytes per second on the Cray, compared to 17 megabytes per second on a SPARCstation 10 clone.

I/O is handled by a 68030-based VME computer, dubbed the I/O Subsystem (IOS), that taps into a 266-megabyte-per-second memory port of each CPU pair and handles all communication between memory and VME-based peripheral devices.

The EL94's cabinet can accommodate only a single IOS, so its VME throughput is limited to 40 megabytes per second. For higher band-

continues

continued

width requirements there is a memory-based HiPPI (High-Performance Parallel Interface) channel pair option, which bypasses the IOS. One or two HiPPI-channel pairs can be configured, adding up to 400 megabytes per second of additional I/O bandwidth. The EL98, big brother to our review unit, supports 16 IOS, which boosts VME I/O bandwidth, and four HiPPI channel pairs.

The EL94's CPUs are "brainiacs" rather than "speed demons" and employ a five-stage pipelined architecture. Four scalar units perform arithmetic and logical operations on scalars, which are single 64-bit words. Vector and floating-point operations are handled by four other execution units. Vector operations work on vectors, which are groups of 64-bit words. Though vectors are more frequently thought of as mathematical constructs, in the EL94 they are simply sets of numbers on which the CPUs perform their operations. A characteristic of vector operations is that one CPU instruction operates on many operands, unlike scalar operations, where one instruction works with at most two operands. Supporting the processing units are 72 scalar registers, each 64 bits wide, and 8 vector registers, each containing 64 64-bit words. All of this creates a computational orchestra that performs up to four vector operations simultaneously, whether addition, multiplication, or a combination of the two.

What's a Cray good for?

The EL94's word length is 64 bits — a C integer variable is 64 bits long, double-precision quantities are 128 bits. The machine we reviewed had 64 megawords, or 512 megabytes, and as we said before, no cache. With a large word length comes greater numerical precision and a much larger range of numbers accessible by integer and floating-point variables. A fast memory subsystem allows programs to work effectively on large datasets and access this data without worrying about running into performance roadblocks by stepping out of cache or waiting for a page of memory to be swapped in from disk. The kinds of problems that rely on these features are most often found in engineering and scientific disciplines, rather than in commercial applications.

A typical Cray application is a relatively small program working on an enormous dataset. Consider simulating airflow over an aircraft's wing. After dividing the volume of air around the airfoil into tiny cubes, physical quantities such as the temperature, pressure, and velocity within these cubes can be stored in 3-D arrays. The simulation program would then process this data over increments of time and calculate how the values change, effectively modeling how the wing would perform in the air. It is these sorts of simulations that permit aircraft designs to be tested without having to build physical models and test them in wind tunnels.

Cray's software-development environment is not the namby-pamby collection of *cc*, *make*, *lint*, and *dbx* that forms the standard programmer's workbench. It is a set of powerful tools that analyze, debug, and help parallelize programs. When you buy a Cray, you expect high-performance computing. Cray's software suite allows programmers to achieve performance gains not possible on high-end Unix workstations.

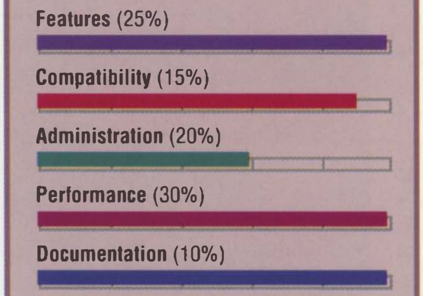
Because it shares the same software environment as Cray's larger systems, engineers can develop programs on the EL94 and deploy them on bigger iron. Alternatively, rather than share a slice of a C90, for example, a department might prefer to purchase its own EL94 that makes up for its lower performance by being a dedicated machine; utilizing 100 percent of a 400-megaflops machine is (roughly) equivalent to a 10-percent slice of a 4-gigaflops machine.

ASTC

Advanced Systems Test Strip

EL94
Cray Research Inc.
655-A Lone Oak Dr.
Eagan, MN 55121
800-289-2729 or 612-683-3800

Pricing $300,000 as tested.
Summary If you have array-intensive calculations, you need a Cray. The EL94's superb system-software suite, vector performance, and upward compatibility with Cray's bigger iron makes owning megaflops easy.



Overall rating 9.0

A prime example

Vectorizing a program can offer tremendous speed improvements on the EL94. Parallelizing the program further boosts performance. A problem that fell out of our attempts to port a peculiar C program to the Cray served as an excellent test bed for doing both.

A favorite exercise to test new supercomputers is to have them find the next largest prime number. David Slowinski of Cray Research found the current largest on a 16-CPU C90. It is $2^{859433}-1$, otherwise known as the 33rd Mersenne prime. This number cannot be expressed as a 64-bit integer, which makes it impossible to calculate using normal means (you can't simply do *pow (2,859433)-1*). However, algorithms exist that can perform the necessary computations while preserving precision. One such method, compliments of Cray Research, demonstrates the power of parallelization and vectorization.

We won't go into the exact nature of the algorithm used to calculate the 33rd Mersenne prime (interested readers are urged to send mail to *david.burnette@advanced.com* for a copy of the

How test strips work: *Categories are judged compared to other products in their class. We judge different products on different categories as needed.* ***Features*** *evaluates capacity, expandability, reliability, and availability features.* ***Compatibility*** *includes compliance with relevant standards, ease of porting from other Unix systems.* ***Administration*** *gives credit for tools that aid system administration.* ***Performance*** *summarizes tests of various comparative performance metrics.* ***Documentation*** *looks at the quality and completeness of paper and on-line documentation. (Other products reviewed in this issue are rated on a similar system, with categories adjusted to the nature and use of each product class.) Weightings are based on reader surveys and expert knowledge. Total of extensions is divided by 50 and truncated to one decimal place to yield an overall rating on a scale of one to ten. Adjust the weightings to customize the Test Strip to your own needs.*

programs), but its structure merits some attention. The important computations take place within a main loop, which is easily diced up and spread over the EL94's four CPUs (i.e., parallelized). Within the main loop are three smaller loops that step through the elements of an array and perform some simple arithmetic operations on them. On a typical RISC machine, the processor would have to treat each array element individually. On the EL94, however, these three inner loops are easily vectorized, meaning that groups of the array elements are processed at once, making for a tremendous increase in performance.

Special directives exists within Cray's Fortran and C implementations that instruct the machine to parallelize certain regions of a program. Similarly, other directives, when placed before loops, instruct the machine to vectorize the code. To avoid incompatibilities when porting code to other architectures, Cray ingeniously cloaks the Fortran directives as comment lines, and uses *#pragma* statements in C. Our source code moved to workstations with only a few modifications, and these had to do with calls to timing routines, not the parallel and vector directives.

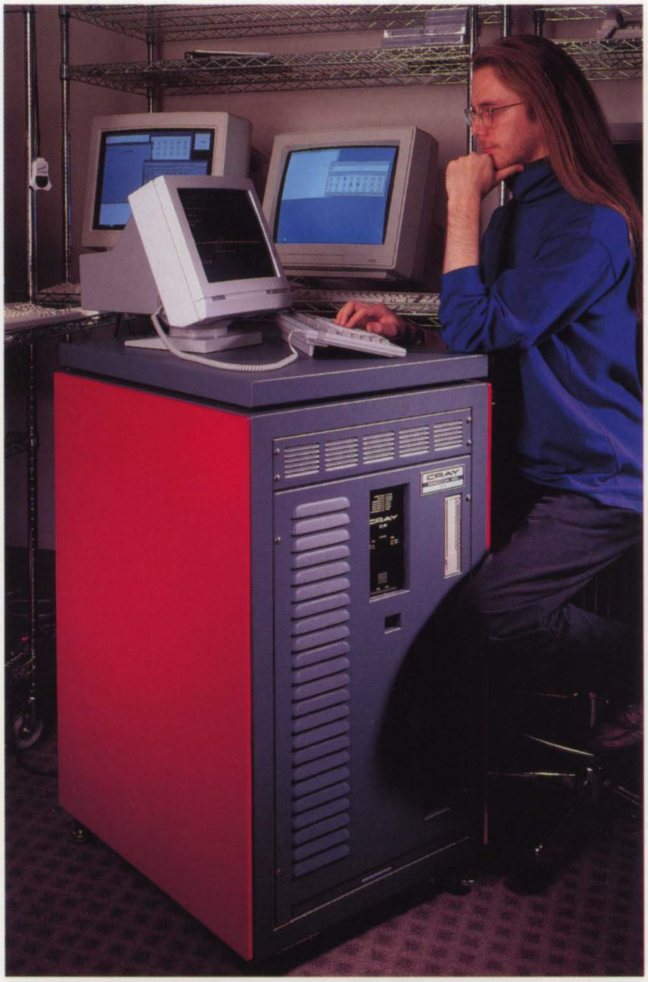
See chart **Mersenne prime benchmark results** for a comparison of the Cray with other machines running Fortran and C versions of the Mersenne program.

It takes more than hardware

Cray may be popularly known for its liquid-cooled lightning-fast hardware humming away in the basements of government agencies reading our mail (just kidding), but we soon discovered the software side of the supercomputing equation is a significant factor in achieving high performance.

It's all fine and dandy to have a vector machine idling beside your desk, but if you can't harness its full capabilities you may find yourself with an expensive space heater rather than a state-of-the-art supercomputer. Cray has a suite of system utilities that analyze Fortran and C programs, providing data to help you tailor them to best fit the EL94's architecture. This set of tools comes bundled with the development environment and is a powerful aid to getting the most from the machine.

continues



RIO COSTANTINI

Manufactured in Chippewa Falls, WI, the EL94's steel-plated exterior belies its warm, friendly Unix interior.

How we scored the EL94

Features. Cray's software suite is impressive, easy to use, and valuable. The EL94's self-diagnostic features, enormous disk capacity, and CPU and I/O expandability make for a very scalable computing resource.

Compatibility. EL94 binaries can run on Cray's big YMP and C90 machines and source code flows freely between the architectures. Its programming environment is standard Unix for the most part.

Administration. The care and feeding of an EL94 is no more difficult than for any other large Unix server. Cray's support infrastructure is elaborate and professional.

Performance. What can we say? Provided the machine is used for what it's good at — vectorizable programs working on large datasets — the EL94 is an impressive workhorse. It is not a commercial enterprise server, however, so don't expect the likes of FrameMaker to operate at lightning speeds.

Documentation & support. Cray's documentation is both abundant and various. On-line man pages, the docview on-line documentation interface, and several yards of gray three-ring binders provide a wealth of information. Though voluminous, the paper docs are indexed well and feature "roadmaps" that guide novitiates to desired sections. Service fees are based on system configuration and start at $1,000 a month, which provides access to a toll-free service center and on-site maintenance. The EL94 comes with a 90-day warranty.

continued

Though their interfaces are somewhat crude compared to GUIs on the RISC-workstation side of the fence, Cray's tools make up in power and effectiveness what they lack in luster.

No development suite would be complete without a source-code debugger. Though not a tool to parallelize or vectorize your code, *cdbx* is Cray's version of *dbx* with an X Window interface. *dbx* vets will find using *cdbx* easy and painless. We pointed and clicked our way through loops, breakpoints, and pointer references. It came in especially handy when we ported the Mersenne program from Fortran to C; 20-megabyte core dumps at 45 megaflops can get ugly.

Once a program graduates from *cdbx*, it's ready for *profview*, Cray's X-based utility to analyze *prof* output. Anyone familiar with ordinary Unix *prof* will find *profview* easy to use. It displays pie charts showing the percentage of time spent executing a program's various routines. Other features provide *prof*-esque tabular information, sorted in the usual ways: by number of calls, CPU consumption, etc. *profview* relies on data generated by *prof*, which in turn gets its data from a file created from a program's successful execution.

The *atexpert* utility, short for "autotasking expert," attempts to examine Fortran or C source code and predict its speed up when run on multiple processors. (Cray calls parallelization *tasking*, and it considers automatic parallelization *autotasking*.) As with *profview*, using *atexpert* requires some preparation. Cray's Fortran and C compilers have special options that cause an executable to create a datafile, which *atexpert* uses to make its predictions. After one successful run of the target program, *atexpert* reads the datafile and shows the degree to which certain routines will benefit from parallelization, as well as how the entire program will fare over multiple CPUs.

We used *atexpert* on several programs. It was a good indicator of which codes would benefit from multiple processors and which weren't worth the effort to parallelize. The Mersenne program scored high marks with *atexpert*. Compare its prediction of speed up with what we actually achieved and you'll see a close match: *atexpert* predicted a speed up of 3.4; we got 3.3. *atexpert*'s graph predicts performance improvements for a default of eight CPUs; we had only four on which to test its prognostications. In addition to predicting a program's parallel performance, *atexpert* offers observations on what can be done to derive greater speed gains from multiple CPUs.

Another autotasking tool, *atscope*, analyzes Fortran source and helps find loops that can be parallelized but the compiler might not detect automatically. With appropriate mouse clicks, *atscope* will embed the necessary tasking directives in a source file, which can be a great time saver for large programs.

Two tools provide timing and execution statistics for programs. *flowview* is an X-based utility that shows the amount of time spent in each routine along with a dynamic calling tree that describes the calling relationships between all routines in a program. It also reports the number of calls, the average time per call, and indicates which routines are good candidates for inlining. A special trace library must be linked into a program that, when successfully executed, produces a datafile that *flowview* uses to get its statistics.

jumpview provides similar information to *flowview* but works differently by combining simulation and runtime statistics. It uses the same trace library, but you must execute your programs with the *jt* command, which loads a special kernel that analyzes the program's instructions and alters them to generate counts of the number of times each code block was executed. *jumpview* is more accurate than *flowview*, providing timings down to one clock period, and can report megaflops ratings for your programs.

While *flowview* and *jumpview* focus on the execution of a single program, *procview* takes a wide-angle approach and reports memory and I/O activity for a process or group of processes. *procview* gets its data from the *procstat* command, which requires no special libraries; programs run as is. With this

Seven disk drives dominate the forward section of the EL94 and provide 21 gigabytes of storage. The central panel contains system reset switches and status indicators. Power cables go to the on/off breaker in the lower right.

RIO COSTANTINI

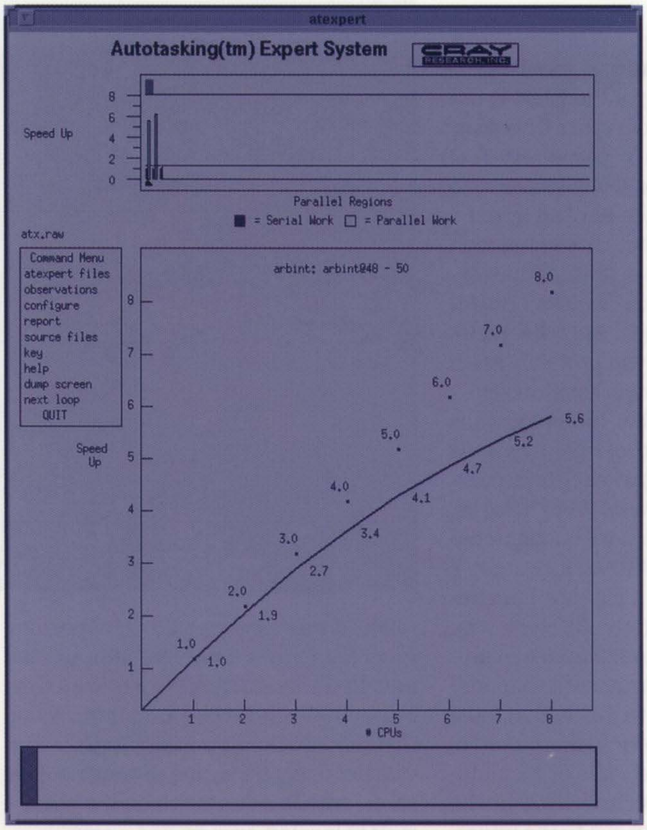
tool, you can see which files were performing the most I/O, what the transfer rates were, and whether there was any wait-time due to contention for disk devices or I/O channels.

The rubber meets the road

With all this said and done, it's time to get down to brass tacks and address the performance of the Cray EL94. We strapped ourselves into the Cray's bucket seats and went on a vectorized road race through the rarefied air of high-performance computing. At Cray's suggestion, we used standard supercomputer benchmarks to gauge the performance of the EL94. In addition to the Mersenne program discussed above, we ran the stride3, stream, Linpack1000, and NAS kernels tests. The results are summarized in the table **Supercomputer benchmark results**.

The stride3 test is a Fortran program written by Mark Seager at Lawrence Livermore National Laboratory. It attempts to measure the effects of cache and memory speed on a machine's megaflops (millions of floating-point operations per second) rating. The program accesses an array with successively greater *strides* or increments. See chart **stride3 benchmark results** for a comparison of the Cray with a two-CPU SPARCstation 10 clone.

For the most part, the Cray maintains an even 40 megaflops of performance. There are bands at lower levels, however, resulting from multiple accesses to the same bank of memory. The EL94's memory is divided into 32 banks. The bank busy time, or the amount of time that a bank is unavailable after it has been accessed, is five clock periods. To minimize bank conflicts, data is distributed among the 32 memory banks in a round-robin manner. Accessing data sequentially doesn't cause any bank conflicts but accessing memory by strides of 32 will cause the same memory bank to be used repeatedly and cause lower performance. This affect is also seen at strides of 16 and 8, although not as strongly.



Cray's *atexpert* analyzes programs to determine how they would benefit from parallelization. Its projections of speed up matched closely with reality in our tests. Though Cray hardware is legendary we were equally impressed with its software.

Fortunately, it is easy to avoid memory conflicts by using a simple programming trick: set the leading dimension of multidimensional arrays to an odd number. For example, in C, instead of doing *int data[64][64]* do *int data[65][64]*. The advantage of this scheme is that a program can access all of the machine's memory and not see a significant drop in memory bandwidth. This is in contrast to cache architectures, for which any long stride results in seriously degraded performance. Simulations that require datasets on the order of 100 million elements and access this data in irregular ways will easily violate caches. The Cray's strength in this regard is that all data references are satisfied from its high-speed memory. Main memory acts essentially as one large 512-megabyte cache.

The stream test is a Fortran program written by John McCalpin at the University of Delaware. It also tests memory bandwidth and provides an aggregate megabytes-per-second value for operations on a two-million element array. The Cray achieves a maximum rate of about 1.2 gigabytes per second to main memory from four CPUs. Our trusty SPARCstation 10 clone achieved a maximum value of 17 megabytes per second.

Linpack is the old diehard of floating-point benchmarks. The original version used a minuscule array of 100 by 100 elements. Linpack1000 does a little better by using a 1,000 by 1,000 array, which probably will not fit in most systems' caches. The EL94 comes closest to reaching its theoretical peak performance of 532 megaflops, achieving a high of 311 megaflops on four CPUs.

The NAS kernels were developed by the Numerical Aeronautical Simulation program at NASA Ames. Several versions of these benchmarks ranging from level 0 to level 1,000 are officially recognized. The level 0 permits no changes to the source code, level 20 allows 20, and so on. We ran the level 20 version and achieved a value of 40 megaflops. As with SPECmarks, this value is a geometric mean of several individual tests. For the record, the EL94's performance ranged from 112 to 21 megaflops on these subtests.

While our intention is not to suggest a Cray EL94 and a SPARCstation 10 are equivalent or that any purchase decision will come down these two systems, we offer this comparison to show where the Cray lies on the performance curve and how on some tests, the workstation doesn't fare too badly. Whereas the SPARCstation 10 is a good general-purpose desktop computer, the Cray is targeted at a narrower range of applications on which it performs quite well.

continues

continued

Supercomputer benchmark results

	NUMBER OF CPUS				
	1	2	3	4	
NASkern (no changes)	24	21	20	19	Megaflops
NASkern (20 changes)	41	38	35	33	
Linpack1000	107	182	254	311	
Stream (assignment)	424	843	1113	1282	MB/second
Stream (scaling)	425	837	1106	1281	
Stream (summing)	450	1008	1147	1229	
Stream (SAXPYing)	516	1018	1136	1228	

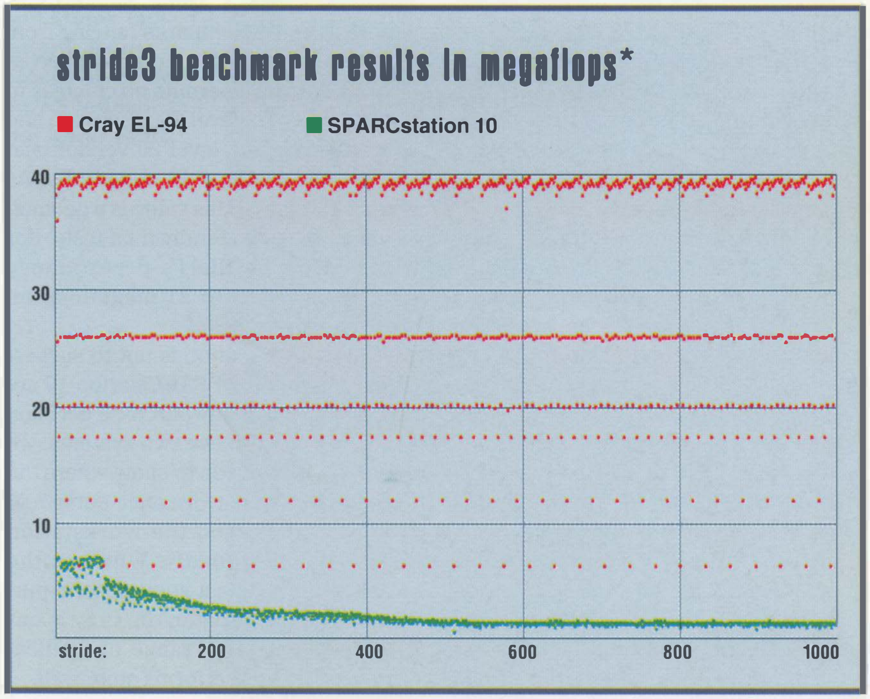
Houston, we have a problem

The care and feeding of the EL94 is not too much different from other Unix iron. Tread-worn System V gurus will feel right at home. BSD longhairs will have to adjust to a few include-file and system-command changes, but these shouldn't pose too great a threat to their mastery of the arcane. Unicos' access control lists, accounting groups, and other mainframe accouterment will probably seem the most alien to the workstation set.

We ported a number of applications over to the Cray, ranging from *rayshade* (a public-domain raytracer) to a password cracker to several benchmarks. The greatest difficulties we encountered were porting from BSD to a System V environment. Also, Cray implements semaphores in a slightly different way, which may require modifications to programs that use the C intrinsic semaphore calls. Cray's counter to this is that most scientific and engineering programmers prefer the higher-level calls of its multitasking library, rather dwelling in the depths of C. Casting an eye to the future, Unicos will incorporate POSIX Pthreads during its 8.0 and 9.0 releases.

During the EL94's stay in our lab, it experienced one hardware fault: one of its two SCSI controller boards went belly up. Though an unfortunate occurrence, it provided us with on opportunity to try Cray's technical support hot line. Every interaction we had with Cray personnel reinforced our impression that they were put on this Earth to service our every need, and the engineer we spoke with in Atlanta, where the hot line staff is located, was no exception.

The technician walked us through an extensive series of diagnostics and self-checks, which allowed us to determine the nature of the fault before a service engineer came to our site. These test were performed by the I/O subsystem, which is a separate computer in its own right with its own SCSI disk and filesystem chock full of investigative utilities. An engineer arrived the next day, swapped in a new board, and booted the machine. Its familiar drone filled our lab and we were back in business. Diagnosing the problem from afar is a boon to service personnel, who can tend to failures with foreknowledge rather than walking blind into a system catastrophe.


stride3 beachmark results in megaflops*

Alone in a hot room

What's a Cray EL94 good for? As a general purpose Unix server intended to run databases, spreadsheets, and word processing packages, look to other, cheaper machines. If your applications are numerically intensive with large datasets that easily break caches on more conventional machines, the EL94 may be your best bet. We found using Cray's parallelizing and vectorizing directives is not difficult, and its suite of software tools makes tackling large codes easier. To get the most from an EL machine, you'll have to wade into these strange waters.

Before the EL94 arrived in our lab, we wouldn't have recognized a parallelizeable program from Microsoft's Rube Goldberg NT. By the time the EL94 finally waddled out the door, however, we realized parallel vector processing is not an obstacle, but a bridge to greater levels of performance. The toll is small, the reward great. ■

David Burnette (david.burnette@advanced.com) *is a technical editor at* Advanced Systems.

Reprinted with permission from the July 1994 edition of **Advanced Systems** magazine.