Special Report

1997 Gordon Bell Prize Winners

This year we saw a dramatic improvement in price/performance, even though the latest machines were not used. Instead, the entrants found a way to squeeze more performance out of each node and reduce communication overhead.

Alan H. Karp
Hewlett-Packard Laboratories

Ewing Lusk
Argonne National Laboratory

David H. Bailey
NASA Ames Research Center

The Comprehensive Nuclear Test Ban Treaty, which forbids the testing of nuclear weapons, is responsible for the dramatic performance gains reported in this year's Gordon Bell Prize competition. The connection between the treaty and the Bell Prize is simple to understand: One way to maintain confidence in the nuclear stockpile needed for national security is to use computation to tell you what you would have learned from testing. The recognition that such computations far exceeded the capability of existing computers led US Department of Energy officials to initiate the Accelerated Strategic Computing Initiative (ASCI) program in high-performance computing.

The first result of this program, the ASCI-Red machine (briefly described in the sidebar, "ASCI-Red: Big Iron that Really Shines"), was delivered to Sandia National Laboratories in 1997. Four of this year's entrants reported results on this machine.

This year we also saw a dramatic improvement in price/performance, even though the latest machines were not used. Instead, the entrants found a way to squeeze more performance out of each node of their workstation farm and reduce the communications overhead among nodes. One entry reported speedup results, a category that is no longer recognized. Nevertheless, this entry received full consideration. The winners were announced at Supercomputing '97 in San Jose, California. Details can be found in the conference proceedings.[1]

BELL PRIZE

The Bell Prize recognizes significant achievements in the application of supercomputers to scientific and engineering problems with particular emphasis on

- performance, which recognizes those who solved a real problem in less elapsed time than anyone else, or
- price/performance, which encourages the development of cost-effective supercomputing, or
- compiler-generated speedup, which measures how well compiler writers are doing at easing the programming of parallel processors.

Gordon Bell, a former National Science Foundation division director and now a senior researcher at Microsoft, is sponsoring $2,000 in prizes each year for 10 years to promote practical parallel-processing research.

WINNERS

Nhan Phan-Thien and Ka Yan Lee of the Mechanical and Mechatronic Engineering Department of the University of Sydney in Australia, along with David Tullock of Los Alamos National Laboratory, were awarded $1,000 for their work in modeling suspensions on a workstation farm consisting of 28 DEC Alpha machines. Even though these were older machines running at "only" 266 MHz, the team reported an impressive 10.8 Gflop/s/$M (billion floating point operations per second per million US dollars) and an equally impressive raw performance of 4.7 Gflop/s. Had the newest Alpha processors been used, each node would have run almost twice as fast as those available for these runs.

Our other winner is a bit unusual in that it combines two entries. These entries have two things in common. They both describe solutions of the N-body problem using the so-called *tree-code* method, and Michael S. Warren of Los Alamos National Laboratory and John K. Salmon of the California Institute of Technology participated in both winning teams.

Warren and Salmon used ASCI-Red to study clustering in the early universe by following the motions of 322,000,000 self-gravitating particles. The total simulation ran at 170 Gflop/s. However, at early times, when the particles were more uniformly distributed, the code ran at over 430 Gflop/s. Had they chosen to solve a less interesting problem that did not develop large density fluctuations, they could have reported this number. They also ran a test case similar to one that won an earlier Bell Prize using an N^2 algorithm and measured a computation rate of more than 630 Gflop/s.

Their other submission, done along with Donald J. Becker of the NASA Goddard Space Flight Center, M. Patrick Goda of Los Alamos, Thomas Sterling of Caltech, and Gregoire S. Winckelmans of the Universite Catholique de Louvain in Belgium, presented N-body simulations of self-gravitating particles and of vortices in a fluid. Their work was done on a

0018-9162/98/$10.00 © 1998 IEEE

farm of 16 PC-class machines costing about $50,000 in total, on which they achieved around 1 Gflop/s, some 20 Gflop/s/$M. This price/performance number needs to be lowered somewhat to account for the price difference between complete systems and the "parts-in-a-box" approach used here.

Each of the other entries had something to recommend it. At one stage in the judging, it looked like everyone would be named a finalist. The other entries are briefly described below, after a closer look at the winning entries.

PRICE/PERFORMANCE

It's morning. You pour cereal in your bowl, shake the orange juice, fill your glass, and pour the milk over your cereal. What does this scene have to do with high-performance computing and the Gordon Bell Prize? The answer is another question: Why did you shake the juice but not the milk?

Milk and orange juice both consist of water with some dissolved substances and some solid material. However, the solids in milk form a colloid while those in orange juice are in suspension. The solids in a colloid do not settle; those in a suspension do. Hence, the orange juice must be shaken to mix in the solids that have settled to the bottom of the container.

Suspensions surround us, from the black plume coming from the bus you're trapped behind, to the concrete being poured for your new sidewalk, to the paint you're about to apply to your living room wall, to your salad dressing. In fact, any time you see the instructions "shake well before using," you're dealing with a suspension. Even the blood in your veins is a suspension.

Suspensions of solid particles in a liquid are of importance in a number of industrial processes. For example, some chemical processes are practical only with suspensions because the surrounding liquid both mixes the constituents and carries away excess heat. Designing such a process requires an understanding of such things as the sedimentation rate, thermal properties, and viscosity of the resulting mixture. This last topic is the area studied by this year's price/performance winners, Nhan Phan-Thien, Ka Yan Lee, and David Tullock.

One way to determine the viscosity of a suspension is to drop in a heavy particle and measure the rate at which it falls. (We aren't measuring viscosity here in the sense of fluid dynamics, just resistance to flow in a particular direction. But it's easier just to say "viscosity.") The rate of fall will depend on the real viscosity of the fluid itself, the density of the test particle relative to the fluid, and the density and concentration of the particles in suspension. The rate may also depend on other factors such as the shape of the suspended particles and their interactions with the fluid.

Dropping a marble in a test tube is an easy experiment, but models can provide additional insight. For

ASCI-Red: Big Iron that Really Shines

In 1991, Eugene Brooks of Lawrence Livermore National Laboratory subtitled his project's annual report, "The Attack of the Killer Micros," by which he meant that commodity microprocessors would replace specialized engines for high-performance computing. This year's Bell Prize results might answer the question of whether or not Brooks was right. Unfortunately, the answer depends on what you mean by micro.

If by micro you mean the CPU, the war is over, and the micros have won. Only one entry submitted this year was run on a conventional supercomputer, and its performance lagged the best reported by almost an order of magnitude. If, however, by micro you mean a system, as in a PC or RISC system farm, the micros are losing. The performance of the commodity-system-based farms still lags behind the big iron, epitomized by ASCI-Red, by up to two orders of magnitude. So while the performance of the micros has reached the point where gangs of them can beat up on a lowly vector processor, they still need the specialized infrastructure of the classic supercomputers to really shine. We'll have to wait and see if the micros catch up in this area as well.

The ASCI-Red machine used by four of this year's entrants is the first step in achieving a five-order-of-magnitude increase in compute power over the conventional vector supercomputers. Many feel that increasing sustained compute speeds from the 1 Gflop/s range to over 100 Tflop/s will enable scientists to include enough of the relevant physics in their problems to get reliable results.

The ASCI-Red machine has more than met its target of a peak speed of 1 Tflop/s. It currently has a peak performance of 1.8 Tflop/s and has run at over 1 Tflop/s on the Linpack scalability test. Each node consists of two Intel Pentium Pro processors, each capable of performing 200 Mflop/s. Communication bandwidth between nodes is more than 6,000 times that of a conventional Ethernet. System memory is almost 600 GBytes, and enough I/O performance is available to checkpoint the entire memory in less than five minutes. The 85 cabinets that make up the machine take up some 1,000 square feet.

Is it only a coincidence that the fastest machine in 1946, the Eniac, also took up 1,000 square feet? Of course, the Eniac only performed 5,000 operations per second, but that was good enough to be a thousand times faster than its contemporaries. The 1.8 Tflop/s ASCI-Red only outdoes its competition by a factor of 10 or so.

Hardware is nice, but software makes it usable. As much as possible, the software provided with the system makes the machine look like a Unix supercomputer. High-level languages, interactive debuggers, performance analysis tools, and a checkpoint/restart capability are all provided.

Suspensions of solid particles in a liquid are important in many industrial processes. Designing such processes requires an understanding of such things as the sedimentation rate, thermal porperties, and viscosity of the resulting mixture.

example, it isn't easy to measure the motion of a test particle through wet concrete. We'd also like to know why the suspended particles move the way they do. Is their motion dominated by the forces generated by the test particle or do the suspended particles affect each other? How do the forces they experience depend on particle density, particle shape, and fluid viscosity? Answers to such questions can help engineers design suspensions that better suit their needs.

The problem with modeling suspensions is the complexity of the computation. We can get the information we need if we can find the rigid body motions of the suspended particles and the marble as it falls through the fluid. The complications come from the geometry of the configuration. The problem isn't the container—that has a regular shape—or the test particles—we can choose them to be spherical. The culprits are the suspended particles: there are lots of them, and they can have arbitrary shapes.

Conventional modeling approaches don't work. Directly simulating the motion of the particles requires that we know the force one particle exerts on another via the intervening fluid. These force laws are only available for spheres and ellipsoids. Partial differential equation methods are difficult because each suspended particle has a boundary with the fluid. Hence, standard methods like the finite-element method are computationally intractable because they require a very large number of elements spread through the volume of the system just to resolve the boundary.

The boundary element method is another way to solve partial differential equations. Here we discretize just the boundary, which leaves us with a two-dimensional problem. We can find all the forces on each particle by evaluating an integral that depends on the value on the boundary. Since we don't know the value on the boundary, we end up solving an integral equation. This equation is easier to solve than the linear equations from the finite element method.

There is a problem for the suspension, though. The most direct application of the method results in a system of equations that is inherently unstable. Good results can still be obtained if sufficient care is used, but the problem was avoided in this entry by the use of a particular formulation of the problem that gives an integral equation that can be solved iteratively. An added bonus is that the algorithm divides up naturally over the processors of a parallel system. The method involves two key steps.

First, we know that the fluid can flow along the surface of the particle and that there is no fluid moving inside it. Mathematically, the fluid velocity has to jump across the boundary. Normally such a discontinuity would make it impossible to get a solution to the problem, but mathematicians have a way around the problem. Think of a thin plastic sheet with a positive charge on one side and a negative charge on the other. An electron approaching this sheet would be attracted to one side and repelled from the other; the force is discontinuous across the sheet. Mathematicians have developed techniques that deal with such double layers, and these methods can be applied to the surface of the particles in the suspension by treating the force on the surface as a double layer.

The second key part of the numerical method has to do with the rigid body motions of the particles. The idea is easiest to see if the particles are spheres. In this case, the solution doesn't change if a particle rotates through some angle. In other words, the problem doesn't have a unique solution. Fortunately, these rigid body motions can be removed from the problem analytically. A key insight is that this process can simultaneously transform the iteration into one that is guaranteed to converge.

The iterative method is simple and surprisingly effective: A master processor keeps a list of particles that need to be updated for the current iteration. Each worker processor is given one or more suspended particles to work on and the cumulative contribution of all the other particles. The worker then computes a new value for its particles and sends the information back to the master. The master updates the cumulative value and sends it to all the slaves.

Getting the updates of the cumulative data out quickly has a tremendous advantage. A run where updates were delayed until the start of the next iteration took over 900 iterations to converge. With immediate updates, convergence took only 37 iterations. The method is also very scalable, with the parallelizable work in one model run accounting for more than 95 percent of the total. The fact that this fraction stays constant indicates that the overhead due to running in parallel doesn't change as more processors are added.

The results of the submitted calculation show some interesting results. The viscosity determined by the sedimentation rate of the test particle mostly falls within the expected range as the volume fraction of suspended particles is varied from 8 percent to 40 percent. Another measure of viscosity—the resistance to rotation—follows a different curve, one close to the lower bound of empirical estimates based on the behavior of two spheres as they squeeze a fluid and rotate relative to each other. Further calculations will be needed to interpret these results.

The problem submitted modeled 1,000 particles, each discretized into 24 finite elements, for a total of 98,000 unknowns. The calculation converged in six iterations, taking almost three trillion floating-point operations and finishing in a little over 10 minutes on a farm of 28 DEC AlphaStation 500 machines with

Calling All Bell Prize Participants

Alan Karp

I have been administering the Bell Prize since its inception 10 prizes and 11 years ago. (We skipped a year when we moved the awards ceremony from Compcon in February to Supercomputing in November.) Now it's time to let someone else have all the fun, so I'm stepping aside. I'm leaving the Prize administration in good hands for the next 10 years.

It's good to look back after a decade, especially one that's seen such dramatic changes. Our first performance winner achieved 450 Mflops/s. This year four entries achieved more than 500 times that figure, and they didn't even win a prize!

I'd like to collect the thoughts of all past participants. Please send me any reminiscences you have. I'd like to hear those secret stories about racing the clock to get that last run, convincing system administrators to give you the time on the big machines, and last-minute glitches in your price/performance entry. I have some of my own, like the time the announcement of the finalists was delayed because of Iraq's invasion of Kuwait. Did your participation affect your work at the time? Did it have any affect on your future work? Was the influence positive or negative? I'm particularly interested in the thoughts of those who did not win a prize.

266-MHz processors linked with a Fast (100 Mb/s) Ethernet. This setup cost $435,000 when new, but the machines are no longer being sold. Even at this price, the computation rate of 4.7 Gflop/s translates into 10.8 Gflop/s/$M, more than 70 percent better than last year's winner. Using the depreciated value of these machines would raise this figure to over 15 Gflop/s/$M. The raw performance is equally impressive, nearly five times higher than any other price/performance winner.

COMBINED ENTRY

This year the judges set a precedent by choosing to award a single prize to two separate entries. Of course, the entries have something in common. Michael Warren of Los Alamos and John Salmon of Caltech participated in both, and both entries used the same computer program to implement the basic numerical method. This program, a new version of the code that was used in an earlier Bell Prize winning entry, is used to follow the motion of particles that interact by long range forces such as gravity.

Normally when particles interact over large distances, there is little to do but compute the force each particle exerts on every other particle. The fact that the computer time scales as the square of the number of particles limits the size of the simulation to 1,000,000 particles on even the fastest computers available today.

The more efficient algorithm, called a tree-code because of the data structure it uses to represent the problem, relies on the fact that the force of two very close objects on a distant one is almost the same as that of a single particle exerting the combined force of both. This fact lets us treat a large number of distant particles as if they were one. Now the computer time increases only as the number of particles times its logarithm, NlogN scaling. Of course, if a large number of particles clump together, the forces between pairs of them must be computed individually. In a parallel system such clumping leads to load imbalance because the processors computing the forces on particles in such clumps must do more work than those working on more isolated particles.

Even so, the computer time saving is substantial. Instead of having to evaluate a trillion forces to follow the evolution of 1,000,000 particles, a tree-code only needs around a billion force evaluations. The winning entry was able to follow the motion of over 322,000,000 particles. An unintended result of this work was an illustration of how science really works, in contrast to the way it is often taught in school.

Science is a process, not a set of facts. You propose an explanation for a phenomenon and a way of testing your idea. If you pass the test, your idea gains credence; if it fails, it's back to the old drawing board. Right? Well, not exactly, as the result of this first part of this combined entry shows.

The phenomenon in question is the total mass in the universe. There is strong evidence that there is five to 10 times more mass than we see in luminous matter like stars, gas clouds, and the like. For example, galaxies in clusters are moving so fast that the members of the cluster would fly apart if there wasn't a lot more mass than we see. Astronomers also believe that the expansion of the universe is consistent with considerably more mass than is visible.

One of the hypotheses of the nature of the missing mass is that it is made up of particles moving at modest velocities, the so-called cold dark matter. The test is the formation of galaxies in the early universe. We can't run a set of experiments, but we can ask if a computer model's statistical properties match those we observe.

The input to the models is the density variation following the Big Bang. Even a very small density peak will attract more mass leading to a bigger peak and even more mass. Eventually, we get a galaxy. Large density variations will produce galaxies faster than smaller ones, and the galaxies will have higher velocities relative to each other. Until recently, the observations did not provide a good estimate of these fluctuations, but the Cosmic Background Explorer satellite is now providing more accurate information. Here's where the trouble for cold dark matter comes

Talk about a tough year to pick finalists! Four entries reported performance exceeding 200 Gflop/s/$M.

in. The new constraints on the density fluctuations lead to models in which the distribution of velocities of galaxies in the model is considerably higher than what was observed. So much for this theory.

Not so fast, say Warren, Salmon, and their collaborators. The earlier simulations only followed some tens of thousands of particles. They couldn't simultaneously model a large region of space and the formation of clusters with large numbers of particles. The simulation submitted followed over 322,000,000 particles in a cube almost a billion light years on a side. This simulation is currently being analyzed. It is hoped that it will confirm and refine earlier 17,000,000 particle simulations showing that the cold dark matter hypothesis is consistent with some observations once thought to rule it out.

Several performance figures were quoted. The overall simulation ran for nearly 10 hours on 2,048 nodes (4,096) processors of ASCI-Red at a rate of 170 Gflop/s, an impressive figure considering the load-balancing problems late in the run when some regions had 1,000,000 times more particles per unit volume than others. At early times, before there was much clumping, the code ran at over 430 Gflop/s on 3,400 nodes. By contrast, a 1,000,000 particle simulation using the N^2 algorithm ran at 630 Gflop/s on 3,400 nodes. This calculation took about one minute of wall clock time per time step. In contrast, the calculation submitted followed 300 times more particles but only took two minutes per time step. Clearly, if the physics allows it, the tree-code should be used.

Warren, Salmon, Donald Becker of NASA Goddard, M. Patrick Goda of Los Alamos, Thomas Sterling of Caltech, and Gregoire Winckelmans of University Catholique de Louvain in Belgium submitted the second half of this winning entry. It shows how to use the same tree-code to model both self-gravitating particles and fluid flows.

We can't model individual molecules. A billion molecules would fill a cube only 1/1,000th of a millimeter on a side. Besides, molecules interact with short range forces. Instead, we do what appears to be a surfer's dream—we take the curl of the fluid flow. This mathematical transformation of the differential equation describing the flow changes the quantity of interest from the velocity of the flow at any point to something called *vorticity*, a measure of the "swirliness" of the flow.

Discretizing the vorticity equation gives us a representation of the flow as a collection of little whirlpools, *vortices* in fluid mechanics jargon. Each vortex can be treated as an individual particle interacting with the other vortices by a long-range force. Although this force is more complicated than gravity, the same tree-code can be used to follow the motions of the vortices.

There is one complication that doesn't arise in the gravitational problem that follows the motion of physically distinct objects. The vortex particles are representing a continuous fluid. As they move, some of them clump together, leaving regions where the particles are far apart. To prevent this clumping from degrading the accuracy of the solution, additional particles are added into the emptier areas every once in a while. In the simulation submitted, the model started with 57,000 vortices and ended up with over 360,000.

The computers used in this entry hark back to the early days of stereo. Back then companies like Heath would sell you a box of parts that you could assemble into a stereo. All you had to do was plug the right part into the right socket and run all the wires to the right place. That's what was done by the Los Alamos and Caltech groups. They each bought a box of parts, things like an Intel Pentium Pro 200-MHz CPU, an Intel motherboard, disks, memory modules, various controller cards, right down to the cases. Even the software, Linux and GNU tools, is more like a kit than a commercial product. Each site constructed a cluster of 16 machines assembled from these parts at a cost of about $50,000 per cluster.

The Los Alamos machine was used to follow the motion of nearly 10,000,000 self-gravitating particles as they formed galaxies in a region of space nearly a third of a billion light years on a side. The complete simulation ran at 880 Mflops/s, but at earlier times that didn't have much clumping the performance reached 1.2 Gflop/s.

The Caltech machine was used to follow the merger of two smoke rings, a problem that has long been used to study the fundamental processes of turbulence. One puzzle such models may help solve is why the two smoke rings merge more rapidly than analytic models predict. Flows such as this are particularly challenging for fluid dynamics codes because the swirling motions in each smoke ring result in very complicated grids. Running at nearly 950 Mflops/s lets investigators use enough particles to get realistic simulations.

It's difficult to assign a price/performance figure to these pile-of-parts machines. If we simply use the cost of the parts, we get 20 Gflop/s/$M, but this isn't fair to those who buy commercial machines. A custom PC shop will charge two hours of labor for the hardware assembly and two hours of labor for installing the operating system and testing the machine. At $75/hour (We're in the wrong line of work!), we're left with better than 18 Gflops/s/$M, an impressive figure for such challenging problems.

OTHER ENTRANTS

Talk about a tough year to pick finalists! Four of the entries reported performance exceeding 200 Gflop/s, and two reported price/performance of over

1998 Gordon Bell Prizes

Entries for the 1998 Gordon Bell Prize are due May 1, 1998, and finalists will be announced by June 30, 1998. Pending approval by the Supercomputing 98 program committee, finalists will be invited to present their work at a special session of that meeting in November 1998. Winners and honorable mentions will be announced following the presentations.

The 1998 prizes will be given for work in the following categories:

- *Performance.* The entrant must convince the judges that the submitted program runs with superior performance compared with any other comparable engineering or scientific application. Suitable evidence will be the megaflop rate based on actual operation counts or the solution of the same problem with properly tuned code on a machine of known performance.
- *Price/Performance.* The entrant must show that the performance of the application divided by the list price of the smallest system needed to achieve the reported performance is superior. Performance measurements will be evaluated as for the performance prize. Only the cost of the CPUs, memory, and any peripherals critical to the application need be included in the price. For example, if the job can be run on diskless computation servers, the cost of disks, keyboards, and displays need not be included.
- *Compiler Parallelization.* The combination of compiler and application that generates superior speedup will be the winner. Speedup will be measured by dividing the wall clock time of the parallel run by that of a good serial implementation of the same job. These may be the same program if the entrant can convince the judges that the serial code is a good choice for a uniprocessor. Compiler directives and new languages are permitted. However, anyone submitting an entry in other than a standard, sequential language will have to convince the judges that the parallelism was detected by the compiler, not by the programmer.

There are four general conditions for eligibility:

- The submitted program must have utility; it must solve a problem that is considered a routine production run, such as making daily weather predictions or solving an important engineering or scientific problem. It should not be a contrived or experimental problem that is intended just to show high speedup.
- Entrants in the price/performance category must demonstrate that the machine they used has real utility. (No fair picking up a few used processors for $1 each.) Only list prices of components should be used. If the machine is not on the market, the entry is probably not eligible, although the judges will consider any reasonable estimate of the price.
- One criterion the judges will use for all categories is how much the entry advances the state of the art of some field. For example, an entry that runs at 250 Gflop/s but solves a problem in a day that previously took a year might win over an entry that runs at 300 Gflop/s solving a more mundane problem. Entrants who believe their submission meets this criterion are advised to document their claims carefully.
- In all cases the burden of proof is on the contestants. The judges will make an honest effort to compare the results of different programs solving different problems running on different machines, but they will depend primarily on the submitted material.

Contestants should send a three- or four-page executive summary to Angela Burgess, IEEE Computer Society, 10662 Los Vaqueros Cir., Los Alamitos, CA 90720-1314 before May 1, 1998.

10 Gflop/s/$M. Each of the other entries had something to recommend it. At one stage in the judging, it looked like everyone would be named a finalist.

The highest performance was reported by Mark Sears of Sandia, Greg Henry of Intel, and Ken Stanley of the University of California at Berkeley. They ran MP-Quest, a production electronic structures program that accounts for 50 percent of the jobs on the Sandia Paragon. On 2,809 nodes of ASCI-Red their computation ran at a rate of 256 Gflop/s. The computational kernel ran even faster, at 360 Gflop/s. The significant achievement is that they can now model systems as large as 3,000 atoms in about a day as opposed to the week to months required on other machines.

Next best was the 246 Gflop/s reported by T. Anzaki of Hitachi, H. Nakamura of Tokyo University, and S. Aoki, T. Boku, K. Kanaya, Y. Yamashita, and T. Yoshie of Tsukuba University in Japan., along with H. Nakamura of the University of Tokyo. Their goal is to understand the fundamental nature of atomic particles by modeling the way quarks interact by exchanging gluons. (It's analogous to the way electrons interact by exchanging photons.) Their runs were made on the CP-PACS (Computational Physics by Parallel Array Computer System). This machine has 2,048 HP PA-RISC processors modified to support vector operations. The processors are connected by a network that allows one node to access the memory of another in less than 3 microseconds and exchange data at over 300 MB/s.

Close behind was the entry by Stephen Attaway, Keven Brown, David Gardner, Bruce Hendricson, Steve Plimpton, and Courenay Vaughan of Sandia and Ted Barragy of Intel. They also submitted runs of a production code, this one called Pronto, on the ASCI-Red using 3,600 nodes. Pronto is a finite-element code widely used to simulate the way material deforms, particularly in a collision. The code ran at 225 Gflop/s on a standard calculation. The performance of 76 Gflop/s when there is a collision is also impressive when you realize that roughly two-thirds of the time is spent searching for the contact points, a calculation that doesn't need floating-point operations. The performance would have been considerably higher, as high as 140 Gflop/s, had this search been included in the operation count.

The fourth entry on ASCI-Red was submitted by Karen Devine, Scott A. Hutchinson, Andrew Alinger, John Shadid, Ray S. Tuminaro of Sandia and Gary L.

Commenting on the positive response he has received over the years, Gordon has graciously agreed to extend the Prize for another 10 years

Hennigan of New Mexico State University. They submitted results on yet another production code, MPSalsa, which simulates chemically reacting flows. Their models were used to optimize the characteristics of a reactor that does chemical vapor deposition of gallium arsenide, a substance now being used in some high-performance chips. Their largest problem ran on 3,600 nodes at almost 212 Gflop/s.

Other work was also quite interesting. Ishfaq Ahmad, Shahriar M. Akramullah, and Ming L. Liou of The Hong Kong University of Science and Technology reminded us that parallel processing is important for things other than floating-point calculations. They were able to encode a video stream in real time on 330 nodes of an Intel Paragon, and one-third real time on a cluster of 20 Sun Ultra-1 machines.

Parallel programs are notoriously hard to write, a problem being addressed by David O'Hallaron, Jonathan Shewchuk, Jacobo Bielak, Omar Ghattas, Loukas Kallivokas, Heshing Bao, and Jifeng Xu of Carnegie Mellon University. They described how a toolset they developed helped them parallelize a program used to model ground motion due to earthquakes. Their tool chain that takes a description of the problem domain and a description of the solution method written in C extended with array syntax and a library of primitive operations on arrays. This input is used to generate a grid, partition it for parallel execution, distribute it over the nodes of the parallel machine, and produce a file used for the actual parallel runs. All of this is done without human intervention. They achieved near-perfect speedup on a model of an aftershock of the Northridge earthquake.

Daniel M. Pressel of the US Army Research Laboratory ported a production fluid dynamics program from a Cray vector processor to a number of shared memory machines that use RISC processors. He reported a speedup of almost 25 on a 30-processor Silicon Graphics 2000. Although parallelizing this program was straightforward (a vector code is inherently parallel), tuning the program was a challenge. The main difficulty most people report is that the memory latency on the RISC machines requires careful coding to make good use of the cache memory. Pressel achieved parallel efficiency of more than 80 percent on as many as 30 processors.

Performance of another production code was reported by S.J. Thomas and M. Desgagne of Environnment Canada and A.V. Malevsky of the Centre de Recherche en Calcul Applique, both in the province of Quebec, Canada. Their Mesoscale Compressible Community model is widely used by Canadian researchers to study the atmosphere over a limited geographic area. Canadian operational forecasts are based on a model with a 35-km resolution and take about 40 minutes on an NEC SX-4. This group has been able to do a 48-hour weather prediction at 10-km resolution in only 48 minutes using a 600-processor Cray T3E and NEC SX-4 with 32 processors. They did not say if it rained when their model said it would.

This is the 10th year of the Gordon Bell Prize. The Gordon Bell Prize was originally scheduled to run for 10 years, which would make the 1997 awards the last. Commenting on the positive response he has received over the years, Gordon has graciously agreed to extend the Prize for another 10 years. ❖

Reference

1. *Proc. Supercomputing*, ACM Press, New York, 1997.

THE JUDGES

Alan H. Karp, who chaired the judging committee, is a senior member of the technical staff at Hewlett-Packard Laboratories in Palo Alto, California. Contact him at Hewlett-Packard, 1501 Page Mill Rd., Palo Alto, CA 94304; karp@hpl.hp.com.

Ewing (Rusty) Lusk is a senior computer scientist in the Mathematics and Computer Science Division at Argonne National Laboratory. Contact him at Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439; lusk@mcs.anl.gov.

David H. Bailey, with the Numerical Aerodynamic Simulation Program at NASA Ames Research Center, is one of the authors of the widely cited NAS Parallel Benchmarks. Contact him at NASA Ames Research Center, MS T27A-1, Moffett Field, CA 94035-1000; dbailey@nas.nasa.gov.

E-mail contacts for the prize-winning teams:

Price/performance winner—modeling of suspensions on a farm of DEC—Alpha machines: Nhan Phan-Thien, nhan@mech.eng.usyd.edu.au

Performance winner—N-body simulations on ASCI-Red: Michael Warren, msw@lanl.gov