1996 Gordon Bell Prize Winners

New hardware and an increase in the scale of an existing machine enabled contestants to nearly double last year's results in the performance and the price/performance categories.

Alan H. Karp
Hewlett-Packard Laboratories

Al Geist
Oak Ridge National Laboratory

David Bailey
NASA Ames Research Center

All three teams honored in this year's Gordon Bell Prize competition benefited from new or scaled-up hardware introduced since work for last year's competition was completed. While the gap in raw performance between the price/performance and performance winners narrowed, the ratio was still more than 300.

All 1996 finalists in the annual competition, which recognizes significant achievements in the application of supercomputers to scientific and engineering problems, were in the categories of

- 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.

One entry was received in the third category, compiler-generated speedup, which measures how well compiler writers are doing at easing the programming of parallel processors. As happened last year, one finalist performing work in a category that no longer exists, special-purpose machines, nevertheless received an award for achieving impressive results.

The winners were announced November 21 at Supercomputing 96 in Pittsburgh. Details can be found in the conference proceedings available from ACM Press, New York. This is the ninth year of the prize, which *Computer* administers. 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.

RESULTS

Adolfy Hoisie of the Cornell Theory Center and Stefan Goedecker and Jurg Hutter of the Max Planck Institute for Solid State Research were awarded $1,000 in the price/performance category for running an electronic structures calculation on a Silicon Graphics multiprocessor at 6.3 Gf/s/$M (billion floating-point operations per second per million dollars). This figure would have been even higher had they used less efficient but more highly parallel algorithms that run at a higher rate but take longer to produce the result.

The Grape-4 special-purpose machine that won the performance prize last year received an honorable mention this year in the performance category for a simulation of galaxy formation. Performance of 1/3 Tf/s (trillion floating-point operations per second) could not be ignored. Toshiyuki Fukushige and Junichiro Makino of the University of Tokyo will split $500 for their entry.

Toshiyuki Iwamiya, Masahiro Yoshida, Yuichi Matsuo, Masahiro Fukuda, and Takashi Nakamura of Japan's National Aerospace Laboratory solved a very challenging fluid dynamics problem on the 166-processor Numerical Wind Tunnel. Their original submission reported a computation rate of almost 93 Gf/s on 140 processors, which two months later they improved to 111 Gf/s on 160 processors. While this rate is lower than that achieved last year on the same machine with 140 processors, the problem solved this year is much harder to parallelize effectively. They will share an award of $500.

Three other entrants described impressive applications of parallel processors. Hong Ding and Robert Ferraro of the Jet Propulsion Laboratory applied parallel processing to an interesting problem in climatology, assimilating all the data received from such diverse sources as satellites and weather stations.

Jim Greer of Trinity College in Dublin and Dave Mullally of the Convex Division of Hewlett-Packard

0018-9162/97/$10.00 © 1997 IEEE

1997 Gordon Bell Prizes

The Gordon Bell Prizes recognize achievements in large-scale scientific computing. Entries for the next prize are due May 1, 1997, and finalists will be announced by June 30, 1997. Pending approval by the Supercomputing 97 program committee, finalists will be invited to present their work at a special session of that meeting in November 1997. Winners and honorable mentions will be announced following the presentations.

The 1997 prizes will be given for work in the following categories:

Performance

The entrant must convince the judges that the submitted program runs faster than 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, such as a Cray Y-MP. If neither of these measurements can be made, the submitter should document the performance claims as well as possible.

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 better than that of any other entry. 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 the greatest 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.

General conditions

- 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 Z-80s 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 Gf/s but solves a problem in a day that previously took a year might win over an entry that runs at 300 Gf/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, 1997.

took a new approach to solving the problem of finding the configuration of electrons in molecules. Instead of attempting to solve the Schrödinger equation by numerical means, they searched the space of possible configurations using Monte Carlo methods (random numbers and trial and error).

Makoto Suzuki of the National Institute for Environmental Studies in Japan and Hikaru Samukawa and Osamu Gohda of IBM's Tokyo Research Laboratory decided to automatically parallelize a program written in High Performance Fortran to analyze satellite data. An interesting aspect of this work was their choice of IBM's commercial compiler rather than a research prototype, as used by previous entrants.

New hardware introduced since last year's competition enabled an improvement of 1.8 times in both performance and price/performance. In fact, all three teams honored this year benefited from new hardware. The 26 processors added to the Numerical Wind Tunnel increased its peak performance to 280 Gf/s. An even more impressive expansion of the Grape-4 increased it to 1,269 special-purpose processors and gave it a peak performance of nearly 2/3 Tf/s. The price/performance winner used a new shared memory machine from Silicon Graphics.

The raw performance gap separating the price/performance and performance winners narrowed this year, but the ratio was still over 300. Even using the lower of the two figures reported for the general-purpose Numerical Wind Tunnel, we find that this machine ran more than 100 times faster than the price/performance winner. In other words, a jet engine

simulation that takes 8 hours on the 166-processor Numerical Wind Tunnel would take a month on the price/performance winner. (The eight-day calculation performed on the Grape-4 would take nearly eight years on the Silicon Graphics machine!) Of course, you have to be able to dedicate a multimillion-dollar machine to a single calculation for a day to do it.

Here is a closer look at the problems tackled by this year's competitors.

PRICE/PERFORMANCE

Crystals are regular structures of atoms. Typically, a basic cell made up of a few atoms is repeated enough times to produce a macroscopic structure. For example, sodium and chloride atoms form a cubic lattice that is repeated to build up a salt grain. While salt is important for our biological survival, silicon would appear to be equally important for our economic survival. The chips in all but the most advanced computers start as crystals of pure silicon.

To quote the main character in the movie *Tin Cup*, "Perfection is unattainable." He was talking about golf, but his words apply even in the atomic world. Crystals are not perfect. Even if there are no contaminants, some atoms will be missing from the lattice. Understanding such defects is critical if we are to continue to reduce the size of the circuit elements in our computers.

The silicon crystals used in computers are formed by slowly cooling a melt consisting of almost pure silicon. Because the crystal forms slowly, each atom has time to find its most stable location, one that balances all the forces it is subject to and minimizes its energy. Take one atom out of the crystal and its neighbors' forces are no longer in balance. They must move to new positions, but now *their* neighbors experience a different force, and so on. Two atoms can even form a chemical bond not found in a perfect crystal. As we move farther from the missing atom, its effect on the overall structure lessens, until we eventually reach a region with a structure nearly identical to that of a perfect crystal.

How large are the distortions caused by missing atoms? How likely are these distorted regions to overlap? How far must we go to find an undisturbed lattice? These are some of the questions this year's price/performance winners, Adolfy Hoisie, Stefan Goedecker, and Jurg Hutter, hope to answer. This inquiry led them to simulate a crystal of 64 silicon atoms with one atom missing from the lattice.

While experimental studies and semi-empirical calculations have been used to study such crystals, an approach based on the quantum mechanics of the electrons is also important. Such calculations provide input parameters needed by the semi-empirical methods and can provide insight into details of the crystal structure.

Solving an equation that describes the quantum world can be so computation intensive that researchers must manage a trade-off between computational complexity and accuracy.

Unfortunately, when there are more than just a few electrons, solving the equation that describes the quantum world involves too much computation. Hence, approximations are used. As with all approximate methods, a trade-off is made between the solution's computational complexity and its accuracy (and/or detail). In this year's entry, Hoisie, Goedecker, and Hutter made an approximation that needs more computation but that more accurately represents the physics of the problem than do simpler methods.

The solution scheme starts with the local density approximation of density functional theory. This simplification states that the motion of an electron in the real material is approximated by that of an electron moving in a medium that represents the average of all the other atomic nuclei and electrons. The difficulty is accounting for the effect of this electron on the distribution of all the others. In other words, how do we make the solution self-consistent?

The method used in this entry introduces the self-interaction corrections directly. While potentially more accurate than other schemes, it introduces a significant complication. When the force on an electron's motion is approximated by the average distribution of the electrons and the nuclei, the force experienced by the electron doesn't depend on its location. In this case, the problem can be shown to reduce to that of finding the eigenvalues of a particular matrix. When the self-interactions are included, however, the forces experienced by the electron depend on what orbit it is following around the nucleus.

The calculation involves several different algorithms, each needing careful tuning to obtain good performance on the complete problem. The winning entry made several significant improvements to conventional schemes. The problem was partitioned in such a way as to minimize data sharing, an important consideration even on shared memory machines like the Silicon Graphics system used. The background electric field was computed using fast Fourier transforms (FFTs), and a new way—better suited to the processors' RISC architecture—was devised for writing the terms in the FFTs.

To approximate the electrons' effects on each other, this team found a new representation based on rational polynomials rather than transcendental functions, such as logarithms, that are commonly used. Because evaluating this function doesn't take long, they avoided storing a large table in memory and simply recomputed the values as needed.

They also simulated the influence of the electrons close to the atomic nuclei in a novel way that emphasizes the interaction's local nature and makes the calculation simpler. Furthermore, they sped up evaluation of the interaction by approximating Gaussian and error function evaluations in the repre-

sentation with 50-term polynomials.

The team made several choices that hurt their measured gigaflop rate but reduced their time to achieve a solution. For example, they noted that the early stages of the FFT needed to compute the potential energy involves lots of zeros. By skipping these operations, they were able to reduce the runtime of this phase by about one third, even though the computation rate went down by roughly 10 Mf/s. Even recomputing the approximation for the electrons close to the nucleus cost them gigaflops. The function evaluations ran at only 60 Mf/s, but the memory saved allowed them to solve a larger problem.

The sample calculation ran for 7 minutes on a Silicon Graphics Power Challenge powered by six MIPS R8000 processors. Using a lower bound on the number of floating-point operations gives a rate of 0.9 Gf/s and a price/performance of over 6 Gf/s/$M. The model's output clearly shows the distortions in this part of the lattice as well as a new chemical bond, thus adding to our knowledge of silicon crystals.

PERFORMANCE

We all know how a jet engine works. Fuel is burned in a chamber, hot gases shoot out the back, and because every action has an equal and opposite reaction, the plane goes forward. While this simple process describes the space shuttle rockets, a modern jet engine is considerably more complex. The main problem with an otherwise simple jet is getting enough oxygen mixed with the fuel to achieve sufficient combustion. Unlike astronauts, most of us aren't willing to sit on a tank of liquid oxygen while waiting our turn to take off.

A jet engine has five main components, all intended to increase gas pressure so that the output flows as fast as possible. As the plane moves forward, the inlet scoops up air and slows it, raising its pressure. This slowed air next encounters the compressor, a device comprising alternating rows of rotating and stationary propeller blades. The rotating blades (rotors) add kinetic energy to the gas; the static blades (stators) convert this kinetic energy into even higher gas pressure. After passing through as many as 10 sets of rotors and stators, each set consisting of between 50 and 200 blades, the gas enters a combustion chamber where fuel is burned. Part of the energy of the hot gas turns a turbine that powers the compressor. The rest of the energy is directed as a gas flow that exits through the nozzle and pushes the plane forward.

Jet engine designers face many choices. The engine's power and efficiency are determined largely by the shape of the rotors and stators, the number at each stage, and the number of stages. Until now, detailed simulations have considered only airflow that comes straight into the engine. This assumption simplifies the calculation, since the flow can be assumed to be axisymmetric. While the calculation is still three-dimensional, the computation needs to consider only two neighboring blades at each stage and follow them only for the time it takes for one to rotate to the position of the other.

Detailed simulations have considered only airflow coming straight into a jet engine, but what happens when the airflow enters the inlet at an angle?

The prize-winning entry from the Numerical Wind Tunnel group is particularly concerned with the safety implications of the first few stages of compression when the airflow enters the inlet at an angle. Such a flow occurs when the plane takes off and has its body axis pointing above its flight path, when turbulent air enters the inlet, and during extreme flight maneuvers. These situations require a study of all the blades in several rotor-stator stages.

Many things can go wrong when the airflow is no longer symmetric about the axis of the engine. For example, when the flow enters at an angle, some of the rotors will experience the full force of the air and some will be partially shielded from the flow. The result can be a rotor spinning like the tub of a washing machine with an unbalanced load. Turbulent air entering the engine can lead to a rotating stall when the pressure on both sides of the compressor is the same. In the worst case, the engine can lose all its thrust.

The Numerical Wind Tunnel team decided to parallelize the problem by assigning two adjacent blades to a processor. This decomposition of the problem minimizes the amount of communication needed between processors, but it raises an interesting problem. The metal parts of the engine define the boundaries of the flow. If a static grid is used, the boundary will sweep through the grid as the rotors move. If a grid that moves with the rotors is used, it will sweep past the stators. Either choice will lead to a complex program.

The solution chosen uses two grids that slip past each other. The passages between the blades, extending forward and back to cover the area between rows of rotors and stators, is divided into two blocks. One block rotates with the blades, the other remains fixed. Each block is discretized into a large number of grid points. As the simulation proceeds, the blocks move relative to each other, and the flow from one block into the other is computed by linear interpolation between grid points along the interface. The main computational part of the code now looks very much like an axisymmetric calculation of a single rotor stage, which simplifies the code development effort.

The problem submitted by the winners models a single rotor stage consisting of 160 blades. Each processor is responsible for the solution at 50,000 or more grid points. Hence, each node must be quite powerful and have a large memory. The Numerical Wind Tunnel meets this requirement. Currently it comprises 166 vector processors, each having 256 Mbytes of memory and capable of a peak computing

rate of 1.7 Gf/s. The processors are connected by a high-bandwidth crossbar switch.

The calculation submitted ran at almost 93 Gf/s on 140 processors. Within two months, the winners had increased the rate to 111 Gf/s on 160 processors. A simulation of one complete rotation of the rotors generates 46 Gbytes of data in fewer than 8 hours. Visualizing this much information is a problem. Even using a Cray Y-MP to do particle trace and streamline calculations with several Silicon Graphics workstations for visualization isn't enough to keep up. Work has begun to address this problem.

SPECIAL-PURPOSE MACHINES

Ever since its eye surgery, the Hubble Space Telescope has dazzled us with beautiful pictures. Turbulent clouds of glowing gas, dark blobs containing newly born stars, and detailed views of dying stars have all appeared in the mainstream press. The high resolution of the Hubble has also captured less visually compelling information about other celestial objects that is nevertheless equally important to astronomers.

Toshiyuki Fukushige and Junichiro Makino of the University of Tokyo, this year's honorees for work with special-purpose machines, are particularly interested in new data on the central regions of galaxies. Astronomers now believe that each galaxy is surrounded by a halo of so-called dark matter, material not in the form of stars. Although we can't see dark matter, we can observe the effect its gravitational pull has on stars.

The Hubble data seems to show a cusp in the density of this material that varies inversely with the distance from the center of the galaxy. Astronomers would like to know if such a density distribution is an inherent part of galaxy formation, if it develops as the galaxy ages, or if its presence indicates a massive black hole in the center of the galaxy. Of course, it is also possible that the observations are being misinterpreted.

Direct numerical simulation is one way to determine which of these possibilities is most reasonable; however, great care is needed. The calculation must be quite accurate, since numerical errors can act like diffusion, which will smooth the density. The calculation will also have to simulate a large number of objects. The problem is that a lot of the particles accumulate in the high-density region. We need to make sure the simulation has enough particles farther from the galaxy's center, where it is easier to determine the shape of the distribution.

Fukushige and Makino addressed these concerns in two ways. First, their simulation uses a large number of particles, more than 780,000. They also decided to use direct calculation of the force of each particle of dark matter on each other particle rather than one of the hierarchical methods that approximate the effects of distant objects. This choice means that there is no question about how these approximations affect accuracy. Also, the hierarchical methods lose their performance advantage when the density of objects is highly peaked.

To better understand Hubble Space Telescope data on the central regions of galaxies, researchers speeded up an *n*-body calculation by producing a specialized processor.

Another problem that affects the calculation's accuracy is close encounters between the particles. Because the gravitational force increases inversely with the square of the separation of two objects, near misses result in highly curved trajectories, which in turn require very small time steps. All gravitational *n*-body calculations assume that the force law changes when particles are closer than a certain distance. The smaller this distance, the more accurate the calculation but the smaller the required time step during a close encounter.

Two factors made it possible for the team's calculation to use a much smaller separation for this modification of the force law and a larger number of particles than other studies. First, the individual time-step algorithm was used. Each particle keeps its own clock and knows how large an acceleration it experiences, which determines how large a time step can be used. The position and velocity of the particle that will end up with the earliest time on its clock are updated on the basis of forces from all the other particles at their predicted positions at the new time. In this way, effort isn't wasted frequently updating the positions of slowly moving objects.

The second and most significant difference between the *n*-body calculation submitted by Fukushige and Makino and all other such calculations is the hardware used. The time to compute the forces of each particle on all the others increases as the square of the number of particles. Once we are dealing with more than a few thousand particles, this force calculation completely dominates the computer time needed. The University of Tokyo team's solution to speeding up the calculations was to produce a specialized processor that efficiently pipelines the force calculation.

Each Grape-4 processor can compute 49 floating-point operations every 100 nanoseconds (three machine cycles). With 1,269 processors, the machine has a peak rate of 660 Gf/s. Despite all the overhead of sending the positions and velocities from the front-end processor to the special-purpose processors and all the forces back to the front end, as well as the fact that everything but the force calculation is done on a DEC AXP 3900 (a fast workstation, but still just a workstation), the computation sustained half the peak rate for almost eight days.

The calculation followed 780,000 particles for 3 billion years, with an average time step of under half a million years. This detailed study showed that the

density rises more steeply than inversely with distance from the center of the galaxy but less steeply than inversely with the square of the distance. Thus, further work on the distribution of dark matter will be needed to resolve the differences between this simulation and the interpretation of the Hubble data.

OTHER ENTRANTS

Two of the remaining three entries attacked the problem of assimilating the large amount of satellite data being collected. The last entry proposed a novel scheme for solving electron configuration problems.

Weather models produce their output on a regular grid and should have their initial and boundary conditions specified on the same grid. Unfortunately, observations come from ground stations, weather balloons, and satellites, which rarely produce data at the points where models need it. The situation is complicated by the errors inherent in the measurements.

The Data Analysis Office of NASA's Goddard Space Flight Center has defined the Physical-Space Statistical Analysis System to collect diverse observations at 6-hour intervals and reconcile them with the regular grid used by the weather prediction system. Hong Ding and Robert Ferraro parallelized this system for the Intel Paragon. Using 512 processors, they solved a problem involving 80,000 observations in under 3 minutes. Their computation is so much faster than needed to keep up with the data collection rate that more accurate modeling can be included in the analysis.

Makoto Suzuki, Hikaru Samukawa, and Osamu Gohda parallelized the analysis of data from a Japanese satellite that monitors the Earth's atmosphere. As the satellite enters or leaves the Earth's shadow every 50 minutes, it collects the absorption spectrum of the atmosphere as a function of height. The analysis consists of guessing a vertical distribution of atmospheric components, computing the absorption through each of the 180,000 molecular lines at each of 76 heights, smoothing the result to the spectral resolution of the satellite instruments, correcting the assumed density of each gas, and repeating the whole process 10 or 20 times until agreement is reached between the calculation and the observation. Using a High Performance Fortran program on a 16-processor IBM SP/2 produced a solution in 15 minutes, more than fast enough to keep up with data collection.

Computational chemists want to solve for the electron distribution in molecular systems. The standard scheme is based on expanding the solution as a sum of known functions with unknown coefficients. Unfortunately, an impractically large number of terms are needed to get an accurate solution. Jim Greer and Dave Mullally took a different approach. They noted that most of the terms in the expansion make a very small contribution to the solution and thus could be dropped. Their approach uses a Monte Carlo search technique to build up the set of functions, keeping only those that make substantive contributions to the solution. The problem is highly parallelizable, since the Monte Carlo sets can be computed on separate processors. They solved for the electron distribution in a water molecule in under 7 hours on an eight-processor Convex SPP-1000. Comparable calculations using other methods required tens of hours on faster computers. ❖

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.

Al Geist is a senior research scientist at Oak Ridge National Laboratory and leader of the computer science group there.

David Bailey, with the Numerical Aerodynamic Simulation Program at NASA Ames Research Center, is one of the authors of the widely cited NAS Parallel Benchmarks.

Karp can be contacted at Hewlett-Packard, 1501 Page Mill Rd., Palo Alto, CA 94304; karp@hpl.hp.com. Geist is at Oak Ridge National Laboratory, PO Box 2008, Oak Ridge, TN 37831-6367; geist@msr.epm.ornl.gov. Bailey's address is NASA Ames Research Center, Mail Stop T27A-1, Moffett Field, CA 94035-1000; dbailey@nas.nasa.gov.

E-mail contacts for the prize-winning teams:

Price/performance winner—study of silicon crystal structure: Stefan Goedecker, goedeck@prr.mpi-stuttgart.mpg.de

Performance winner—fluid dynamics on the Numerical Wind Tunnel: Masahiro Fukuda, fukuda@nal.go.jp

Performance winner—galaxy formation study on the Grape-4 special-purpose machine: Toshiyuki Fukushige, fukushig@chianti.c.u-tokyo.ac.jp