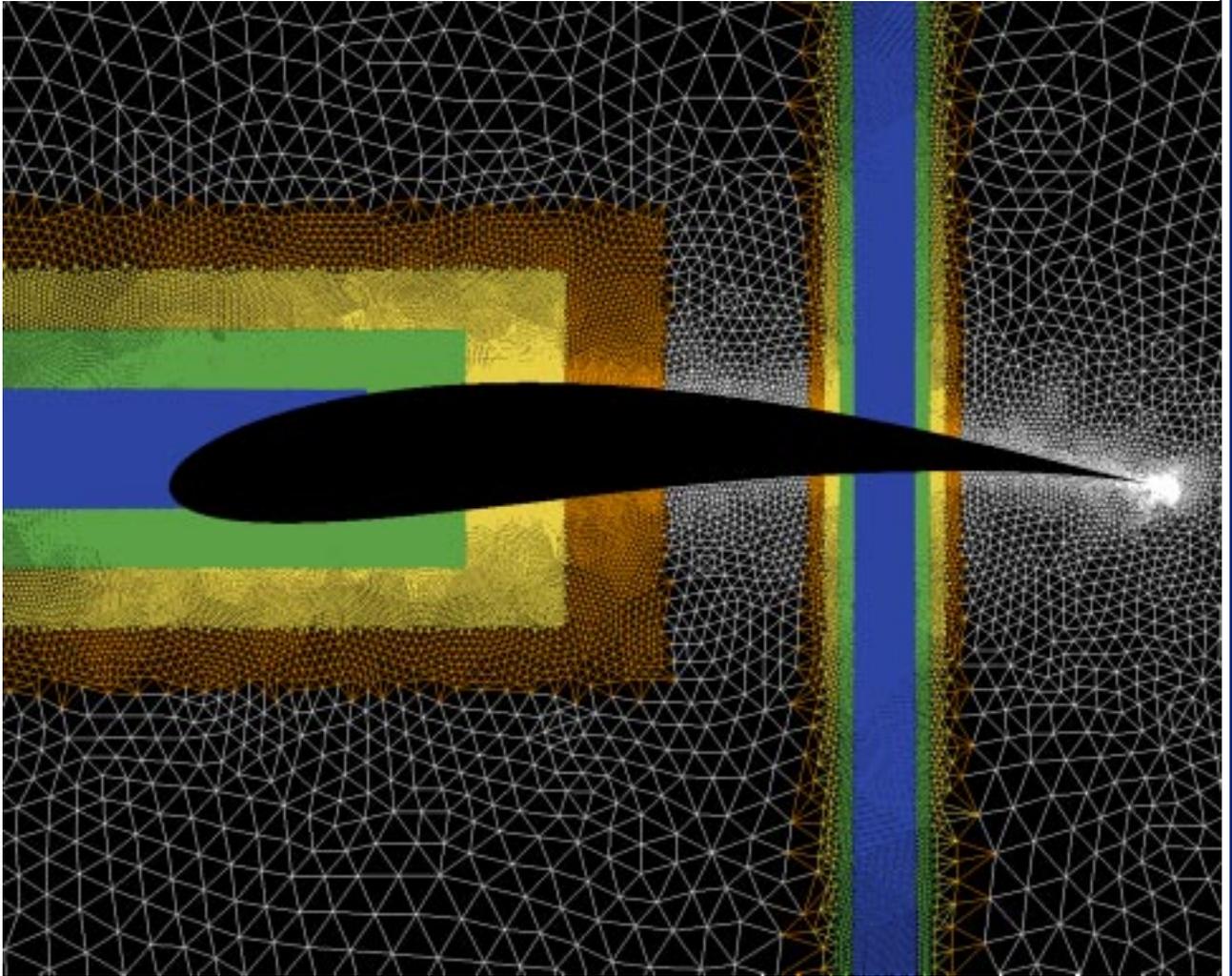


gridpoints



THE NAS SYSTEMS DIVISION QUARTERLY

WINTER 1999

News from NAS

Scheduling tool adapted for the Grid

In September, NAS computer scientists finished upgrading the Portable Batch System (PBS), the division's main supercomputing job queuing system, to make it a cornerstone of the distributed, heterogeneous Information Power Grid network. "There were two main projects," says Bhroam Mann, a researcher in the NAS grid environments group. "The first was to extend PBS to allow time-based advance reservations of PBS-managed resources. The other was to allow PBS to communicate with Globus," the metacomputing toolkit developed at Argonne National Laboratory and the University of Southern California.

In the past, researchers who wanted to use multiple, distributed computing systems simultaneously had to explicitly reserve these resources in advance. The changes to PBS give the system the ability to make these reservations automatically. The new version of PBS also acts as a user interface for Globus, a set of commands used to execute jobs on remote computers (*see story, page 20*). "Globus-aware PBS will allow users to more easily submit their work to the Grid using the familiar PBS interface, without having to learn a new batch queuing system," says Mann. "Scientists want to spend their time doing science, not on new methods for submitting their work."



AMES CENTER FOR BIOINFORMATICS

Virtual clinic headlines NASA SC99 display

The "Virtual Collaborative Clinic" (VCC), a telemedicine application that allows physicians at

multiple sites to interact in real time with shared three-dimensional images of human organs, was chosen as the featured demonstration at the NASA exhibit booth at the SC99 High Performance Networking and Computing Conference in Portland, Oregon. The demonstration, highlighted in publicity materials published by SC99 organizers, is one of more than 30 demos and eight videos to be presented by NASA centers at the November conference. (For a complete list of demonstrations, see page 29.)

Dr. Muriel Ross and colleagues at the NASA Ames Center for Bioinformatics created the VCC, which uses high-bandwidth networks and multicasting techniques to send three-dimensional, high-resolution stereo medical images to physicians at geographically dispersed locations. This way of sharing medical data could one day help physicians to care for people living in remote rural areas, says Ross, or even to monitor the health of astronauts aboard the International Space Station. "The idea is to bring the clinic to the patient rather than the patient to the clinic," she says.

In a much-publicized demonstration in May, Ross and her colleagues used the NAS Facility's Visualization Laboratory to lead the largest VCC session to date. The event linked researchers from the Cleveland Clinic in Cleveland, Ohio, the Navajo Medical Service Center in Shiprock, New Mexico, and two other locations in California. At SC99, researchers from the Bioinformatics Center will report their findings from the May demo and will conduct a live demonstration of the VCC.

NAS hosts first Grid Forum

The NAS Systems Division hosted the first "Grid Forum" workshop in June, drawing an unexpectedly large audience of about 150 researchers. The meeting included researchers from NASA, the National Center for Supercomputing Applications (NCSA), the National Partnership for Advanced Computational Infrastructure (NPACI), the U.S. Department of Energy, and several other institutions. Participants discussed the need to create a close-knit community of researchers working on computational grids such as the NASA Information Power Grid, the NCSA Alliance's National Technology Grid, and NPACI's Metasystems effort.

By the end of the three-day meeting, participants had set up several working groups in areas such as scheduling, security, and account management, and had agreed to establish the Grid Forum as a permanent, informal consortium. One of the body's main functions will be to promote common standards for the various grid-building efforts.

At a subsequent Grid Forum meeting, held October 19-21 in Chicago, participants discussed "rules of engagement" for the standardization process. The working groups also met to identify technical challenges requiring cooperation among grid-builders. "Potentially, the Grid Forum will be a great resource for people to learn about grids and how to interact with them in a systematic manner," says Mary Hultquist, a researcher in the NAS grid environments group.

News from NAS



CORBIS J. WASTE, NASA JPL

Web site helps Mars trip planners

What's harder than getting everyone to agree on where to take the next family vacation? Getting planetary scientists to agree on where to

land the next Mars probe. Now researchers involved in the planned Mars Surveyor 2001 mission can carry on their debate 24 hours a day—and refer to hundreds of three-dimensional images of the Martian surface—using the NASA Mars Landing Site Web site (marsoweb.nas.nasa.gov/landingsites).

The creators of the site, which moved from another Ames division to the NAS Systems Division in June, hope it will give all interested scientists an equal chance to make their cases for their favorite landing spots. Past landing sites for Mars missions such as Pathfinder have been chosen over the course of a few contentious meetings between mission planners and scientists, explains Glenn Deardorff, a researcher in the NAS data analysis group who helped build the site. Geologists, climatologists, astrobiologists, and others would each argue for the spot they considered most scientifically interesting, while mission planners emphasized the sites offering the greatest safety and mission longevity.

Such meetings are still crucial for current and planned missions such as Mars Surveyor 2001 and the Mars Polar Lander. But using the site's so-called "advanced collaboration environment," researchers can examine their colleagues' data, review archives of landing site workshops, learn about engineering constraints, and read and respond to each others' arguments at their leisure.

Adding depth to the site—literally—is a library of three-dimensional models of more than 70 proposed landing sites encoded in Virtual Reality Modeling Language (VRML) format. Researcher Tim Sandstrom of the NAS data analysis group created the models by texture-mapping photographs of each Mars region onto a terrain grid based on data from the Viking spacecraft. The models can be viewed from any angle, accenting the planet's topography in a way that two-dimensional images can't.

Deardorff says the VRMLs have provoked a "very enthusiastic response" among researchers using the site. "They like

having all these resources in perusable form," he says. "And using the VRMLs, they get a more visceral feel for what the terrain is like."

While a final landing site for the Mars Surveyor 2001 probe will soon be selected from a group of five finalists, the Web site will continue to be useful to scientists planning missions in 2003 and 2005, Deardorff says. In fact, the Mars Web site project is merely part of a longer-term research effort on networked environments as forums for innovation in science and engineering. Says Deardorff, "We're using this as a testbed for developing collaborative technologies."

.....



MICHAEL BOSWELL

NAS, SGI collaborate on 512-processor Origin2000

Silicon Graphics Inc. (SGI), a leader in the construction of supercomputers, and the NAS Systems Division, a leading user, have a common interest in perfecting high performance computing systems for the needs of the research community. In 1998, that joint interest became concrete in the form of an official memorandum of understanding, calling for the joint development of a 256-processor SGI Origin2000 supercomputer (*see story, page 17*). Completed in November 1998, the 256-processor machine, named Steger, surpassed benchmarks set by the division's fastest Cray supercomputers by fivefold. Steger, the first "single system image CC-NUMA" machine to reach 256 processors, has since become a commercial product of SGI with a handful sold to some of its major customers. (A single system image computer appears to the user as if it were a single processor with a unified operating system.)

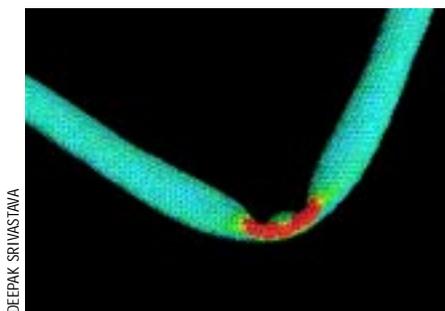
Now NAS and SGI have expanded their partnership by another power of two. In May, NASA Administrator Daniel Goldin and SGI officials signed a new agreement calling for the creation of the world's largest single system image shared-memory computer, a 512-processor Origin2000. SGI delivered the system to the NAS Facility in July. When

running highly parallel software, researchers anticipate, the 512-processor machine will outperform its 256-processor cousin by a factor of three or more.

Successes with Steger made the NAS Facility a logical place to build a 512-processor machine, says Division Chief Bill Feiereisen. NAS engineers' experience with Steger will help them configure and support the 512-processor machine, which will tackle even harder problems and larger datasets.

Division managers named the 512-processor machine Lomax after the late Harvard Lomax, chief of the Ames Computational Fluid Dynamics (CFD) branch from 1970 to 1992. The machine has 196 gigabytes of memory and supports 1.74 terabytes of attached RAID storage (Redundant Arrays of Inexpensive Disks). Jim Taft, a senior consultant in the NAS Systems Division, predicts that by the end of 1999 Lomax will achieve 100 billion floating point operations per second (GFLOPS) when running the production CFD code OVERFLOW, a performance level five times greater than that of Steger. (This improvement will not be wholly attributable to Lomax's greater number of processors, Taft points out. The machine also has more cache memory per processor, and the OVERFLOW code itself has been modified to run faster on 512 processors.) A limited number of benchmark timings for the 512-processor machine will be available by mid-November, Taft says—in time for him to present them at the SC99 High Performance Networking and Computing Conference (*see page 29*). By April 2000, selected users funded by NASA's High Performance Computing and Communications (HPCC) program and the Information Technology (IT) Base R&T program will have access to the machine. The new system will eventually become a major resource for users of the NASA Information Power Grid.

Officials at both NASA and SGI project that very large multi-processor machines will have a significant impact on the way large-scale science and engineering problems are solved. The 512-processor system is already known to carry out calculations far faster than the most powerful computers built by Cray Research. "What used to take a month when scheduled on the NAS Cray systems can now be [done] overnight on the 512-CPU machine," says Taft. Applications such as OVERFLOW and LAURA (the Langley Aerothermodynamic Upwind Relaxation Algorithm, a program for simulating airflow over vehicles in hypersonic flight) will put all 512 processors in the new system to good use, Taft says. Reusable launch vehicles, such as the X-33 experimental space plane, and rescue vehicles for the International Space Station are all being studied using Steger, and will be grist for the new system as well. The 512-processor machine, says Feiereisen, "will allow us to fly a reusable launch vehicle in the computer before cutting a single piece of metal."



Nanotubes of a new bent

Just as an emergency flair will ignite at its center when snapped in half, a bent carbon

nanotube will acquire higher chemical reactivity at the "kink" or folding point, according to simulations recently reported by NAS senior research scientist Deepak Srivastava. The finding, subsequently confirmed in experiments, may mean that nanotube walls can be induced to bond with other molecules, leading to a new generation of devices made from the tiny structures.

Carbon nanotubes have been a tantalizing scientific curiosity ever since their discovery in 1991. The tubes, sheets of carbon atoms rolled into cylinders a few billionths of a meter wide, are some of the most rigid, regular molecular-scale structures in nature. Nanotechnologists have therefore been eyeing them as potential components for miniature motors, transistors, and other devices.

Just as important as the mechanical and electronic uses of nanotubes, however, may be their chemical reactivity. The side walls of nanotubes are usually inert, which has led designers of molecular electronics and nano-composite structures to focus on the tubes' end caps. But nanotubes with reactive spots on their walls could be attached at these points to other nanotubes, or could even be used as chemical detectors, sticking to substances in their environment.

Srivastava decided to investigate the question. In a paper published in the *Journal of Physical Chemistry* in May, Srivastava described both classical and quantum simulations revealing increased local reactivity at sites of twisting or kinking. The greater the twisting or kinking, the greater the region's tendency to bond with gas atoms such as hydrogen. "Controlling the twisting and kinking by mechanical means gives us a unique handle to 'paint' relatively inert sidewalls of carbon nanotubes in many designs and patterns that might be useful," concludes Srivastava. "For example, an entirely new range of nanoscale molecular electronics and sensing applications can be envisioned if we have the ability to 'paint' nanoscale materials in a controlled way."

Putting evolution to work: A new 'genetic algorithm' designs drug molecules autonomously

BY WADE ROUSH

Some of the most effective drugs used today emerged from the laboratories of evolution, not those of the pharmaceutical industry. Morphine, penicillin, digitalis, tamoxifen, and many other compounds derived from plants or fungi take their power from their exquisitely specialized molecular structures, the product of millions of years of random variation and natural selection. It's unlikely that chemists could have conceived of these substances, let alone synthesized them, if they had not been discovered first in nature.

But while nature's ways are powerful, they are also slow. People don't have eons to wait while new medicines evolve. To find new drugs ripe for commercialization, pharmaceutical companies have had to devise far more proactive drug discovery methods, including a recent innovation called "rational drug design." In this approach, researchers determine the three-dimensional structures of known drug molecules, and then build and test molecules with similar structures. They usually search for molecules that fit with the same receptor sites in the body but don't cause as many unwanted side effects as the existing drug. This approach has its own limits, however, since it's not always clear which variations on known molecules are worth testing.

What if evolutionary processes and rational drug design could somehow be

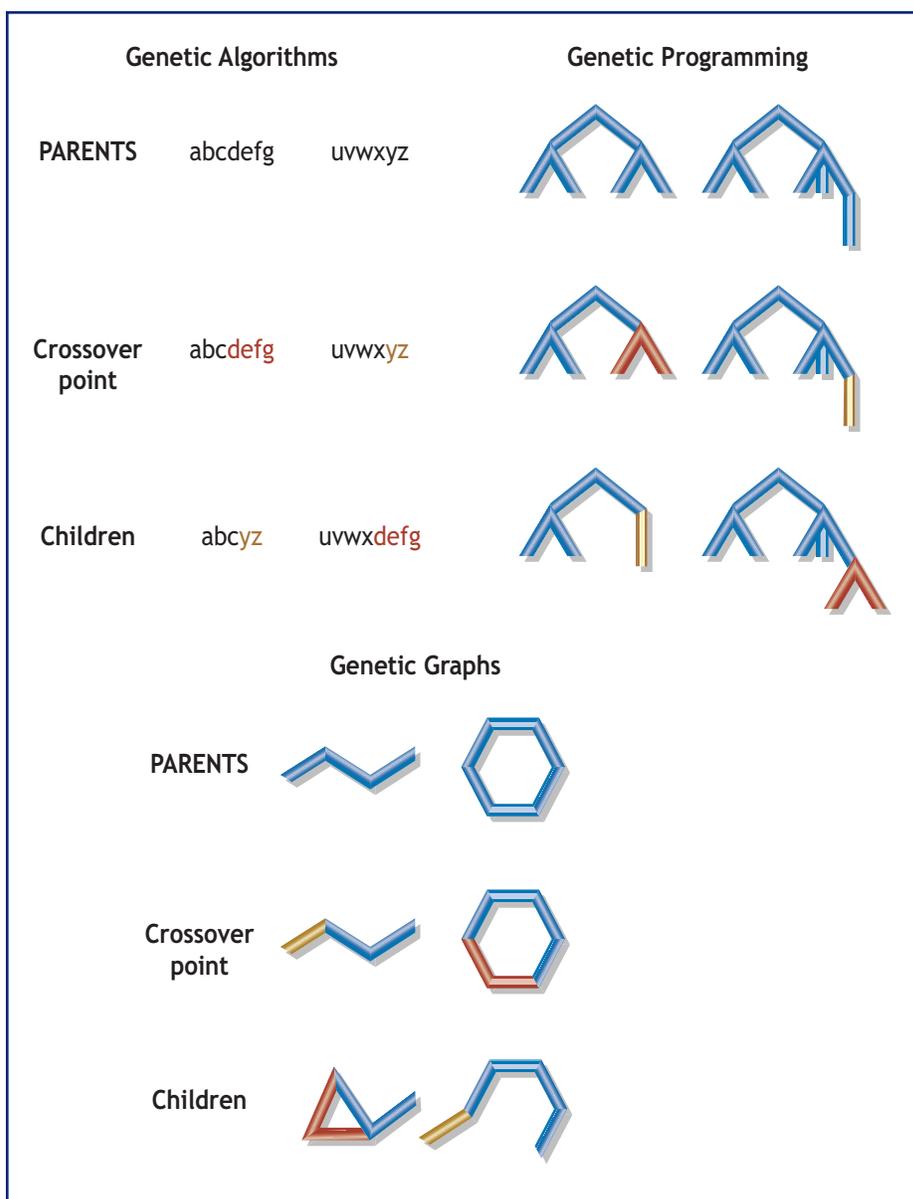


Figure 1. Different types of "crossover" or recombination are used in genetic software to determine which structural information will be passed on to the next generation.

made to work together? Could a procedure similar to natural selection be used to find promising drugs?

Those are two of the questions being asked by Al Globus, a researcher in the NAS Systems Division's science and technology group. In recently completed studies using genetic algorithms, Globus has begun to answer his questions, by "evolving" complex simulated molecules from nothing more than a few simple fragments.

Software measures fitness

The Java-based software Globus developed takes groups of model molecules, represented by simplified stick figures called graphs, and measures their "fitness" in terms of their similarity to a given target molecule. The program randomly breaks apart the fittest molecules in a population, then rejoins them in new combinations. These new graphs take the place of other graphs with lower fitness, and the whole process starts over.

In computer simulations run last fall spanning dozens and sometimes hundreds of generations, Globus's software

spawned several remarkably advanced organic molecules, including morphine and diazepam (a sedative and muscle relaxant). The experiment was not intended to uncover new drugs, Globus emphasizes, but was instead a "proof of concept" demonstrating that genetic software can evolve complex structures autonomously. The researcher using the software provides only a "fitness function," a formula used to weed out undesirable graphs in the population. Globus, giving a lighthearted spin to a complicated process, says "All you have to do is figure out how to write the fitness function, and the program will happily evolve molecules for you."

Molecules aren't the only entities that can be designed this way. Any system whose parts can be represented by graphs, Globus's experiments show, can evolve under the influence of a fitness function. That includes electronic circuits, nanodevices, and other structures. Globus is currently applying the technique to the design of digital electronic circuits, and he reports that the software is already able to generate simple parity circuits.

Going beyond current techniques

Though Globus is one of the first to apply genetic graphs to the design of pharmaceuticals or electronics, genetic software techniques aren't new. Whatever the object being "evolved," the method is roughly the same: Start with a group of randomly generated individuals. Recombine them using "crossover" rules similar to those that govern the exchange of DNA between chromosomes during sexual reproduction. (This process determines which combinations of the parents' genetic information will be passed on to the next generation.) Finally, use a fitness function to determine which recombined individuals will survive into the next generation.

This evolutionary approach to design is remarkably versatile. A new self-configuring airplane wing being designed by researchers in the NAS data analysis group, for example, uses genetic algorithms to test and select the optimal wing shapes for various kinds of flight (see story, page 8). Several years ago, Globus tried to use genetic algorithms to automatically generate descriptions of force fields in molecular dynamics simulations. He eventually gave up on

this idea, but kept thinking about genetic algorithms, wondering whether they might work well in a related field: the

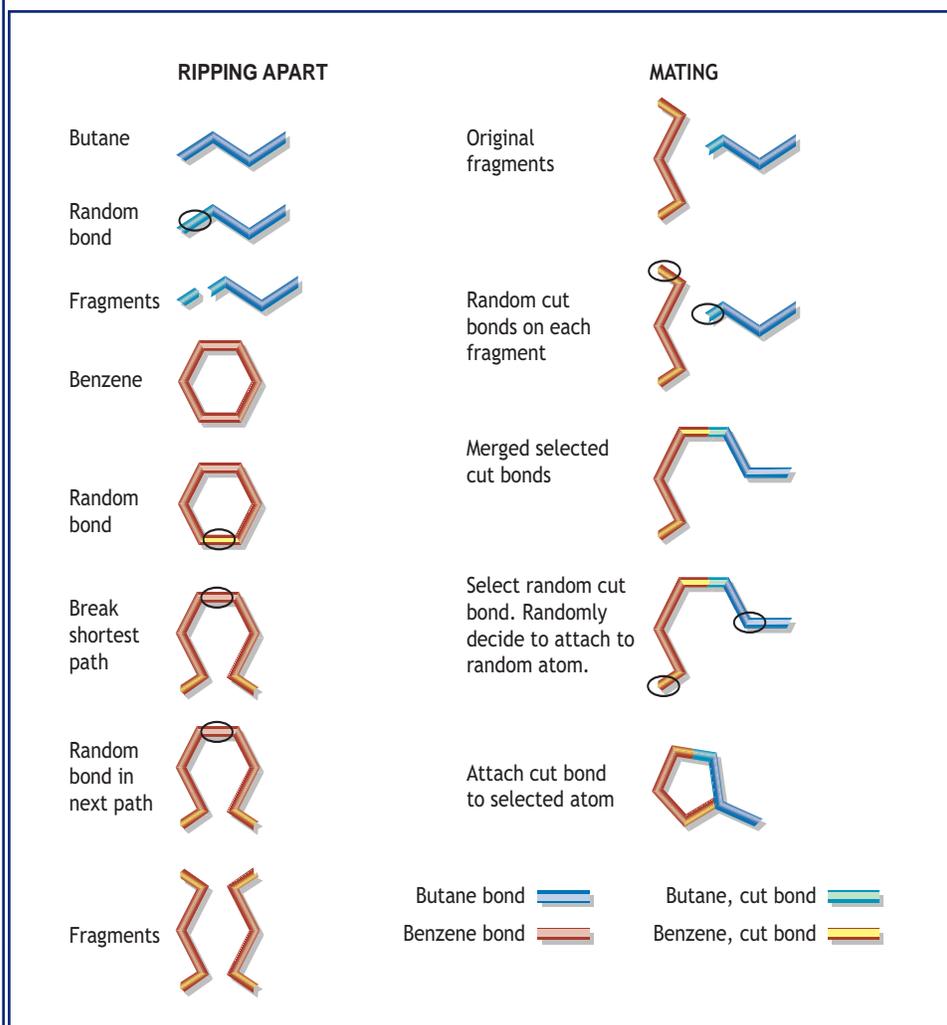


Figure 2. An example illustrating the crossover procedure Globus devised for his drug design studies.

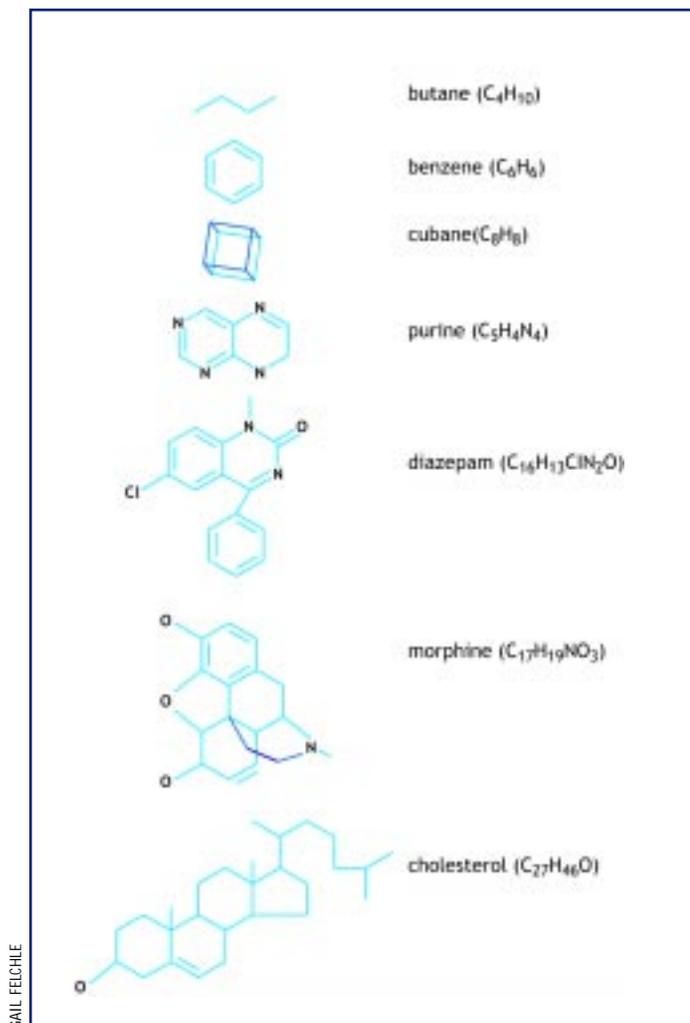


Figure 3. Globus's genetic software "discovered" benzene, second from top, after a median of 39 generations. Cubane was discovered in 46 generations, purine in 245, morphine in 208, diazepam in 256, and cholesterol in 1,765.

molecular design of drugs and nanoscale electronics.

For the technique to work, Globus first had to find a way to represent individual molecules, the units on which natural or artificial selection act. In most past genetic algorithm studies, Globus relates, the objects in a population have been depicted either as strings of alphanumeric symbols or as trees representing the mathematical instructions needed to solve a particular problem (see Figure 1, page 4).

Simulating recombination is easy using such a model. During the crossover phase, randomly selected subtrees are simply exchanged between individuals, creating new instructions.

But graphs can symbolize more than just mathematical instructions, Globus realized. If the vertices in a tree were represented as carbon atoms rather than variables or operators, for example, genetic algorithms could be used to evolve complex organic molecules. "People had evolved trees, but other than one company, nobody had ever evolved graphs," says Globus. "It seemed like a logical thing to do."

During crossover, Globus observed, graphs could be split apart between any pair of vertices and then randomly re-joined, just as the subbranches of trees are exchanged in regular genetic programming. But for these graphs to represent real molecules, a couple of kinks needed to be worked out first. For one, graphs that simply exchange subbranches can never form rings, which are important parts of most organic molecules such as benzene, amino acids, and many drugs. Graphs with rings must therefore be added to the population manually. Second, fragmenting a ring in preparation for crossover is also problematic, since the usual procedure of breaking a single random bond doesn't suffice to cut the molecule apart. "The whole point of using graphs is that you can have rings," says Globus. Unfortunately, "It's not at all obvious" how to write and implement a program to break rings apart in a random, unbiased way, he explains.

Survival of the fittest

To make graph evolution work, Globus had to invent an entirely new crossover operator as well as a new fitness function. In his approach (see Figure 2, previous page), a random pair of molecules is chosen from the population, and a fitness function is used to measure their similarity to the target molecule. (This involves counting the number and varieties of bonds in each molecule and comparing the results to those of the target molecule.) The fitter molecule is called "father," and the less fit individual is discarded. The same process is repeated to find a "mother" molecule.

Next, a pair of atoms in the father molecule is chosen at random. One of the bonds linking this pair is chosen and cut apart. If the molecule has a chain or tree structure, one cut will be sufficient to break the molecule in two; if it includes a ring, bonds continue to be cut at random until every path connecting the original pair of atoms has been severed. The same steps are repeated for the mother.

Then one fragment from each parent is combined, producing a "son." If the father or mother fragments are derived from ring molecules and have more than one broken bond, these bonds are also joined at random, until all broken bonds on at least one fragment have been mated, tying up the fragments' loose ends. The unused fragments of mother and father are also joined together and called "daughter." Two pairs of individuals are then chosen at random from the population. In one pair, the less fit individual is replaced by the son; in the other pair, by the daughter. A single generation is completed when the whole population has been replaced.

Globus knew that if he had designed his procedure correctly, each generation should be fitter, on average, than the preceding one. Given enough time, the fittest molecules in the population should gradually come to resemble or even duplicate the target molecule.

Flight of the condor

Buying the necessary time, however, became Globus's next big challenge. Just as natural selection shows itself only after the accumulated effects on individuals have altered a whole species, genetic algorithms must usually simulate many individuals over many generations before producing promising results. Globus found this was especially true for his software, which behaved very unpredictably. Sometimes the software generated molecules similar or identical to the target molecule after only 40 or 50 generations; other times it went nowhere even after hundreds of generations.

Globus determined that to generate statistically significant results, he would need to run the code 20 to 30 times for each target molecule. The program would stop as soon as the target molecule appeared in the population, but up to a thousand generations might be required to reach this goal. "For small molecules, the programs are over in minutes or hours," Globus says. "But the larger molecules take many hours or even days."

'All you have to do is figure out how to write the fitness function, and the program will happily evolve molecules for you.'

If he used only his own SGI workstation to evolve the graphs, the project would take

years. That led to an "outside-the-box" insight. Globus realized that an abundant source of computing power was all around him—the more than 400 workstations owned by the NAS Systems Division, many of which are idle at night and on weekends. Globus received permission from NAS officials and the go-ahead from computer scientists at the University of Wisconsin to install the Wisconsin "cycle-scavenging" system, called Condor, on a number of workstations. Rather than running the genetic algorithm 30 times on the same machine, he could now run it once on 30 separate machines.

Globus started the experiments using three relatively simple compounds as the target molecules: benzene (C_6H_6), cubane (C_8H_8), and purine ($C_5H_4N_4$). With a starting population of 200 random graphs, the median number of generations required to find benzene was 39.5. With a starting population of 100, the program took a median of 46.5 generations to find cubane and a median of 245 generations to find purine.

When he gave the software tougher molecules as the targets, it still succeeded, but just barely. On one run, the program discovered morphine ($C_{17}H_{19}NO_3$) after 208 generations, but many other runs never approached this target. Diazepam ($C_{16}H_{13}ClN_2O$) emerged after 256 generations in one run, and a molecule similar to cholesterol was formed after 1765 generations.

Globus documented his results in a paper in November

1998 and presented it at the Sixth Foresight Conference on Molecular Nanotechnology in Santa Clara, Calif. Soon after the presentation, however, he discovered a serious bug in his program. It was neglecting to tie up the loose ends during crossover—that is, it was skipping the step in which mother and father fragments with multiple broken bonds are joined at several different points. "If you don't do this second part, it's very difficult for ring structures to evolve," Globus explains.

Once he fixed the problem, the program's success rate rose dramatically. In newer simulations, nearly all of Globus's runs have found their target molecules. "As [evidence] that my program can find a drug molecule, that's an ideal result," he says. He hopes that pharmaceutical companies will pick up on his study and adapt the technique to design novel molecules that are stronger and safer than current drugs. "The real point will not be to find the exact target molecule, but to find molecules that are pretty similar," he says.

Applying evolutionary techniques to circuit design

Meanwhile, Globus has moved on to the evolutionary design of advanced electronic circuits, which can also be represented using graphs. "As far as the program is concerned, a circuit is just a bunch of nodes [such as resistors, capacitors, and transistors] with wires between them," he says. The big difference is that wires in circuits, unlike molecular bonds, have directionality: electrical current can only flow from a negative anode to a positive cathode. This means that during crossover, the program must mate the negative ends of some cut wires with the positive ends of others.

It's "tricky" to arrange this, Globus admits. So far, he has coaxed his software to evolve simple circuits such as parity and delay circuits. It has yet to evolve a serial add circuit, one of the most fundamental circuits in electronic engineering. The eventual reward of such studies, however, could be the blueprints for new kinds of computers. Future nanoscale computing devices, for example, may require circuits with highly novel geometries. It may be easier to evolve and test millions of such structures in simulations than to design even one circuit from first principles.

With the NAS Condor pool now 149 workstations strong, Globus should have plenty of computing power to continue his circuit design studies. NAS senior scientist T.R. Govindan is optimistic that Globus's program will come across useful designs. "Genetic algorithms have the potential to discover molecular and electronic circuit configurations that are unconventional, yet effective," Govindan says. "Studies like Al's are helping advance molecular pharmacology toward finding new, more effective drugs, and I anticipate that there would be a similar impact on electronic circuit design and layout."

Designing a 'smart wing' for the Mars airplane

BY DAVID KENWRIGHT

One of the most familiar and awe-inspiring scenes at the oceanside is that of a seagull soaring, plummeting, and rising again over the waves. The gull switches between these flight modes by pivoting, curling, twisting, or flattening its wings, techniques that Orville and Wilbur Wright are known to have observed before they constructed their 1903 Flyer. The brothers equipped the Flyer with an ingenious system of cables that allowed the pilot to twist the wings in opposite directions, banking the craft for turns and maintaining lateral balance. The discovery of this "wing warping" principle was one of the most important reasons the Wrights achieved controlled flight of a manned, powered aircraft ahead of their competitors.

It seems appropriate, then, that a similar mechanism may be used for a robotic aircraft destined to fly over the dunes and canyons of Mars in the centennial year of the Wright brothers' first flight over the sands of Kitty Hawk. Last February, NASA Administrator Dan Goldin announced that the space agency will fly an airplane on Mars in 2003, taking aerial photographs of the spectacular Valles Marineris, a huge valley system that outstrips Earth's Grand Canyon by a factor of ten. A team of three researchers in the NAS Systems Division is creating a flexible, shape-shifting "biomimetic" wing for possible use on the Mars plane.

The wing, which the team will build and test next year, will mimic living systems in three ways. First, it will continually sense conditions in its environment, process this input electronically, and adjust its own outer shape to achieve an appropriate flight profile. Second, these adjustments will be made by "synthetic muscles" made of active materials, tiny servo mechanisms, and other actuators hidden inside the wing. Third, the wing will be

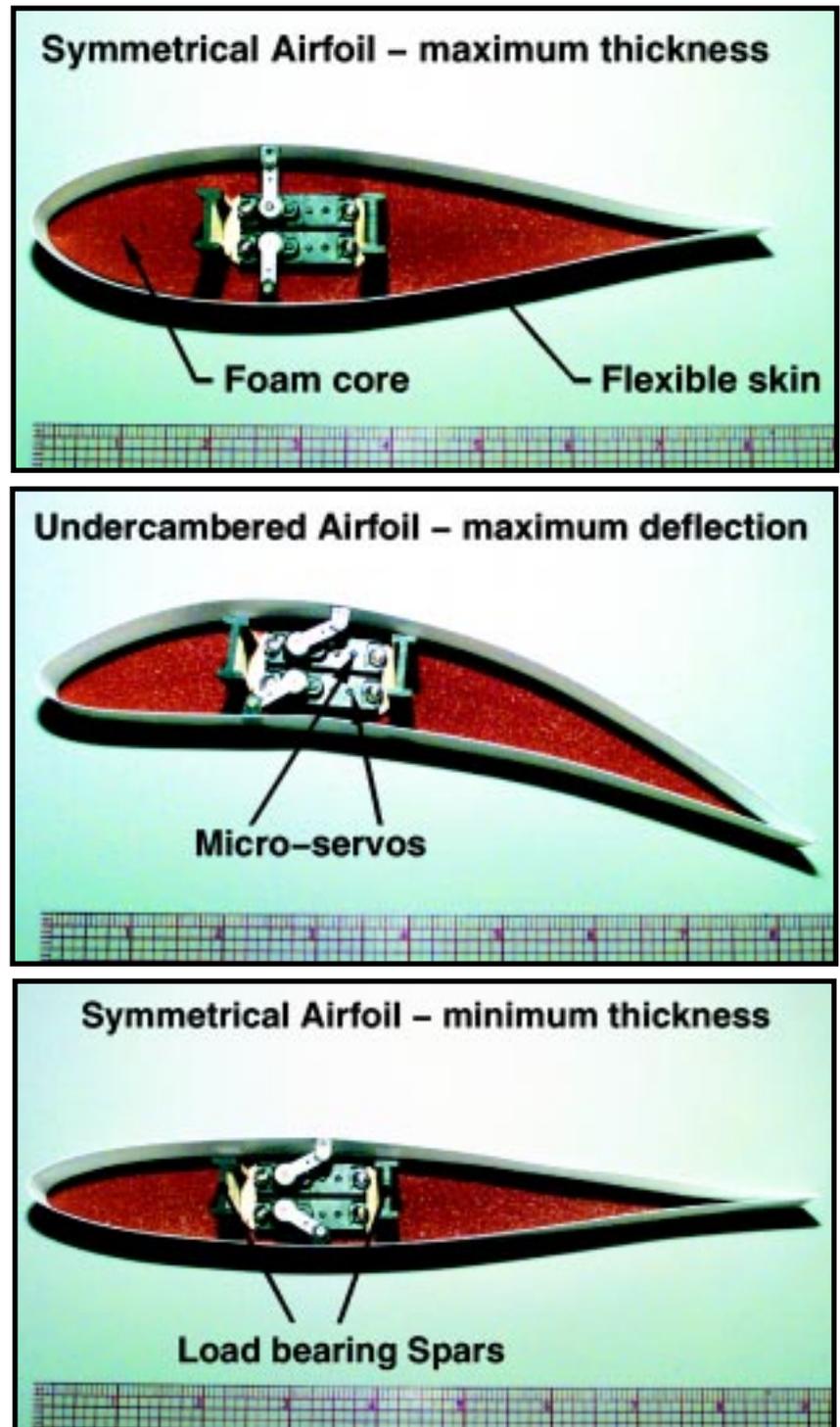


Figure 1. Inside this model "smart" wing is a simple reshaping mechanism using micro-servo actuators. The servos apply forces to the flexible skin and can reshape the airfoil in 1/10th of a second. The prototype wing may contain hundreds of micro servos or thousands of synthetic muscles.

mission it must perform. Why change the shape of a wing in flight? To gather the data researchers need, unmanned aerial vehicles (UAVs) may have to gain altitude using "ridge lift" from mountains or rising thermal currents, traverse long distances with minimal loss in altitude or energy, loiter for extended periods for landing site analysis, or provide a stable platform to photograph geological structures. Each of these modes requires a different airfoil shape or wing cross section for optimal performance.

Bridging biology and engineering

One way pilots control the lift generated by an aircraft's wings is to extend or retract the flaps on the leading edge, changing the wing's surface area. Unfortunately, this method depends on complex hydraulic and control systems with many moving parts, a liability for an aircraft flying millions of miles from the nearest maintenance hangar. Birds' wings offer a far more promising model for an adaptable, self-configured aircraft wing. In fact, scientists and engineers in the emerging field of "biomimetics" are looking to nature as the inspiration for many new structures, materials, and machines. By bridging the fields of biology and engineering, biomimetics researchers can develop smart structures that interact with their environments and change their design to achieve superior performance.

In June, NASA researchers David Kenwright, Chris Henze, and Shishir Pandya received an award through the Ames Research Center Director's Discretionary Fund to design, build, test, and fly a biomimetic wing. The group's early plans and models include several elements new to aerospace design, such as a large array of actuators that will be embedded in the wing and driven by one or more microcontrollers connected to a computer. The actuators will consist of both electromechanical devices (micro servos) and active materials (synthetic muscles).

A variety of synthetic muscle actuators have been developed in the past decade, including shape memory alloys, piezoelectrics, and electroactive polymers. The synthetic muscles contract or bend when energized by an electric current and can be manufactured in wires as thin as a human hair. Synthetic muscles are smaller and more reliable than micro servos because they have no moving parts other than the material itself. They are thousands of times smaller than the smallest available micro servo and may be attached or built into the exterior skin. They have the potential for more precise and complex motions than are possible with micro servos. At present, however, micro servos can be positioned more precisely than synthetic muscles and can be more easily controlled with small and lightweight on-board microcontrollers. Computational studies will be required to determine how to best arrange synthetic muscle fibers in a wing.

The main function of the micro servos will be to apply forces to the outer skin of the wing near its high point (*see Figure 1,*

Continued on next page

equipped to "evolve" optimal wing configurations on its own. During wind tunnel tests, genetic algorithms will be used to generate and test thousands of alternate wing shapes, helping the wing to zero in on the fittest shapes for each mode of flight.

The project is designed to demonstrate the feasibility and advantages of "smart" components for autonomous aircraft such as the Mars plane. The work may also yield new insights into the uses of biological models in aerospace design and other areas of engineering.

Mars aviation: the challenges

A key goal of Mars exploration is to search for the best places to collect samples for return to Earth—places where geology indicates the possible past existence of water or layered sediments. Another goal is to provide high-resolution images of rock structures that are very high or span thousands of miles. Orbital probes such as the Mars Surveyor are already gathering these kinds of data, which help researchers decide where to land vehicles such as NASA's planned 2001 Mars probe (*see article, page 2*). However, an airplane flying a few thousand feet above the Martian surface will be able to capture significantly more detail than even the lowest-flying orbiter.

The Mars Airplane project, led by mission planners at NASA Langley Research Center in Virginia, is in the early stages of development. Engineers already know, however, that designing an aircraft intended to fly in the atmosphere of another planet will present many unique challenges. The gas composition, air density, and gravitational fields, for example, are significantly different from those found here on Earth. On Mars, carbon dioxide is the atmosphere's most abundant gas. The air density at the surface is equivalent to that found at 100,000 ft. above the Earth, and gravity is about one third that at sea-level on Earth. Wind tunnel or flight tests on Earth cannot exactly recreate all these conditions, making it difficult to design an optimal wing shape.

The proposed smart wing will reconfigure and optimize its shape during flight to suit the atmospheric conditions or the

Continued from previous page

thousands of times smaller than the smallest available micro servo and may be attached or built into the exterior skin.

They have the potential for more precise and complex motions than are possible with micro servos. At present, however, micro servos can be positioned more precisely than synthetic muscles and can be more easily controlled with small and light-weight onboard microcontrollers. Computational studies will be required to determine how to best arrange synthetic muscle fibers in a wing.

The main function of the micro servos will be to apply forces to the outer skin of the wing near its high point (*see Figure 1, page 8*). Differential movements of the servo arms change the camber or curvature of the wing,

while coordinated movements increase or decrease the thickness. In the finished biomimetic wing, this mechanism would be repeated at tens or hundreds of locations across the wing. Unlike a conventional wing, the airfoil shape at each location could be significantly different from its neighbor's shape. In optimal configurations, however, the change in shape along the wing is likely to be gradual.

The materials used to construct the biomimetic wing also possess some interesting properties not encountered in present-day aircraft. The exterior skin will flex in response to movements by the internal actuators. Because the external skin must reshape in flight, a rigid internal structure is required to support the bulk aerodynamic forces and bending moments. Internal load bearing spars (analogous to bones in birds) are required to support the skin and provide stable mounts for the array of actuators. The spars will be fabricated from carbon fiber composites based on a construction technique used in high-performance aerobatics aircraft and gliders. The biomimetic wing has no external hinges, seams, or flaps like a conventional wing, so it is aerodynamically "clean" and suited to missions in dusty environments, such as the Mars atmosphere.

Winds of change

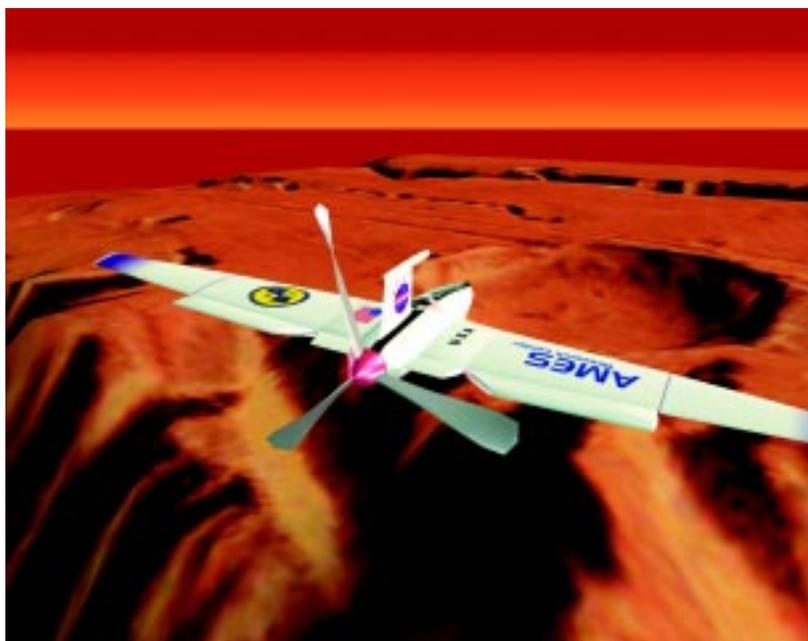
In addition to shape shifting, the biomimetic wing will exploit one more aspect of living organisms: evolutionary design. In wind tunnel tests, an optimal wing configuration will be "evolved" to suit a particular task or conditions using genetic software techniques. For example, a wing shape may be evolved to maximize the lift-to-drag ratio for the purpose

of traversing long distances with minimal loss of altitude or airspeed. The biomimetic wing will reshape itself in just a tenth of a second, allowing thousands of wing configurations to be tested every hour in the wind tunnel.

Speed and selectivity are essential, since a wing containing, say, 100 servos, each capable of assuming 1024 different positions, would have a vast number of possible configurations. A filtering process is needed to efficiently produce wing configurations that are particularly adapted to certain environmental conditions. Genetic algorithms have proved to excel in such roles (*see story, page 4*).

Figure 2 (*next page*) illustrates in more detail how the biomimetic wing will evolve in the wind tunnel. First, a wing configuration is selected by the genetic algorithm. To improve the chances of finding an optimal solution, the initial configurations will be set using results from numerical simulations. The configuration is then relayed through a serial interface to a micro controller embedded in the wing which uses modulated pulses to reposition the servos. Aerodynamic load measurements are then recorded by the force/moment balance and sent back to the PC where they are rated against a fitness function.

By measuring the fitness of different wing configurations, and storing the configurations and resultant fitness in



Researchers at NASA Ames Research Center developed this proposed design for a propeller-powered Mars Airplane.

memory, it is possible to successively build up a population of wings with some distribution of fitness. Especially fit individuals can then be combined, or hybridized, to generate variable offspring for further testing. The fitness functions employed will vary depending on the type of environment to which the wing must adapt. For pre-flight testing and

By bridging the fields of biology and engineering, biomimetics researchers can develop smart structures that interact with their environments and change their design to achieve superior performance.

development, multiple strategies will be explored. The idea is to evolve as varied a repertoire of flight characteristics as possible, so that the aircraft can adapt quickly and efficiently to both expected and unforeseen environmental challenges.

In actual flight tests, the force/moment balance will be replaced by on-board sensors that measure position, velocity, and acceleration. The wing will become a flying laboratory, recording and evaluating its own performance. The smart wing will offer not only greater autonomy and reliability, but could potentially teach researchers Martian aerodynamics without having to build a wind tunnel on Mars.

Evolution in the Wind Tunnel

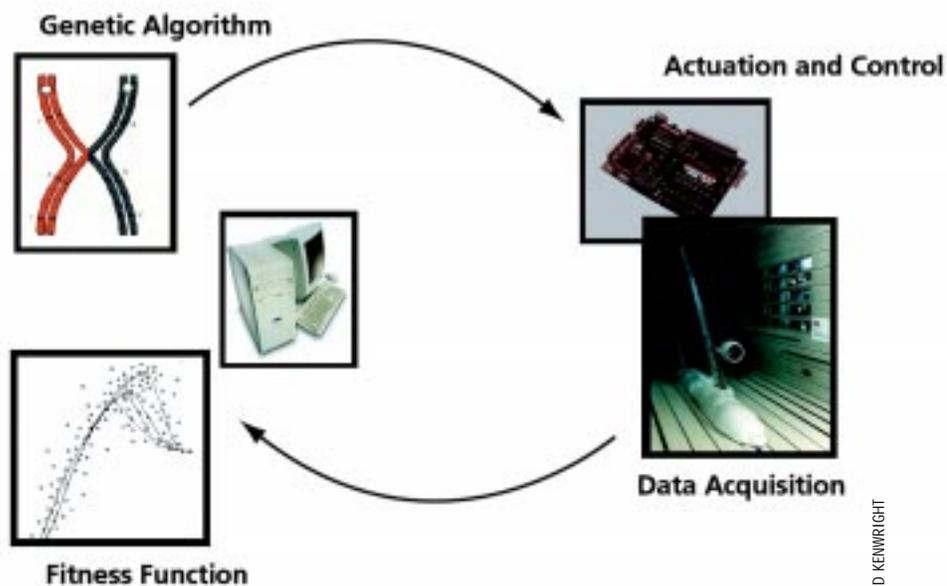


Figure 2. A machine learning program, such as a genetic algorithm, will provide the “brains” to reshape the biomimetic wing. The genetic algorithm selects the fittest designs based on aerodynamic force and moment measurements from either a wind tunnel balance or on-board sensors. The algorithm instructs a microcontroller (embedded in the wing) to reposition the actuators and reshape the wing. This experimental set-up will make it possible to test thousands of wing configurations during a one hour wind-tunnel test.

Building on earlier results

The biomimetic wing project builds on past research at NASA and the Department of Defense. The Defense Advanced Research Projects Agency (DARPA) has funded research into adaptive and flexible wing technology for high-performance military aircraft. The aerodynamic benefits demonstrated by these studies included increased lift-to-drag ratio, improved maneuverability, and delayed flow separation. NASA’s Jet Propulsion Laboratory in Pasadena, Calif., has developed lightweight synthetic muscles using electroactive polymers that will be Mars-tested as wiper blades for scientific instruments on future rovers. Last year, NASA began a six-year “Aircraft Morphing Program” intended to integrate smart technologies into high-payoff aircraft applications.

Before progressing to wind tunnel and flight tests, scheduled to begin in October 2000, the biomimetic wing’s performance will be simulated using new computational fluid dynamics software developed by Ravi Samtaney at Caltech. The software detects large eddies formed around a deforming three-dimensional wing. Data resulting from these studies will be explored using the Virtual Wind Tunnel developed at NAS (*see story, page 24*). The genetic algorithms and actuator control software are also being developed in-house by the NAS data analysis group. With help from many experts such as these inside and outside of NASA, a Mars

plane with seagull-like wings may soon soar over the plains of Mars.

David Kenwright is a member of the NAS data analysis group and a senior research scientist for MRI Technology Solutions.

For a free subscription to *Gridpoints* please return the enclosed postage-paid subscription card.

Radical computer architecture speeds & simplifies dynamic simulations

BY RUPAK BISWAS

The ability of computers to solve hitherto intractable problems and simulate complex processes using mathematical models makes them an indispensable part of modern science and engineering. Aerospace simulations have been a particularly important application, since they are several times cheaper than wind tunnel experiments and field trials, and can be completed much faster. However, because they lack absolute accuracy, simulations have yet to completely replace expensive and time-consuming physical tests.

Aerospace simulations require solving a set of non-linear partial differential equations over a finite region around a simulated object, such as an airfoil. Structured grids that divide the region into many small quadrilaterals are the most natural way to break up or “discretize” a computational domain. Structured grids are characterized by a uniform connectivity pattern, that is, all internal grid points have a fixed number of neighbors. However, complex domains must often be divided into multiple structured grids to be completely discretized, requiring a great deal of human intervention. Unstructured meshes, by contrast, can be gen-

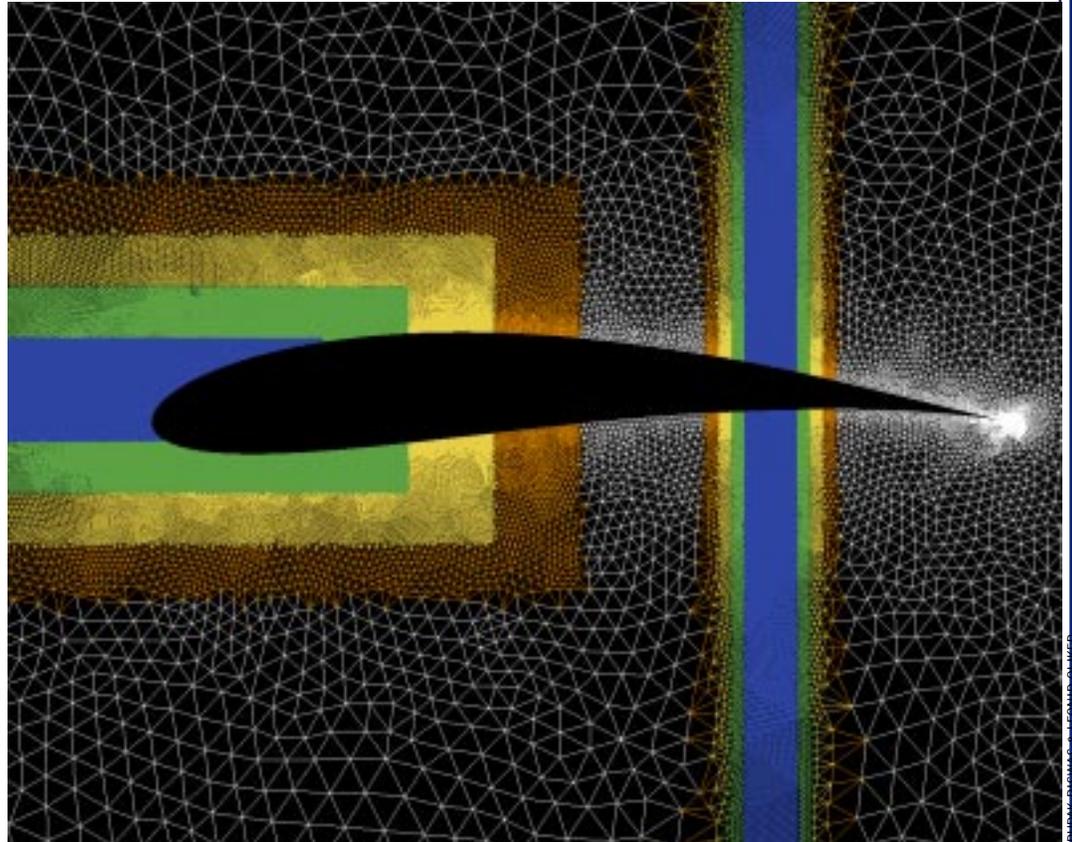
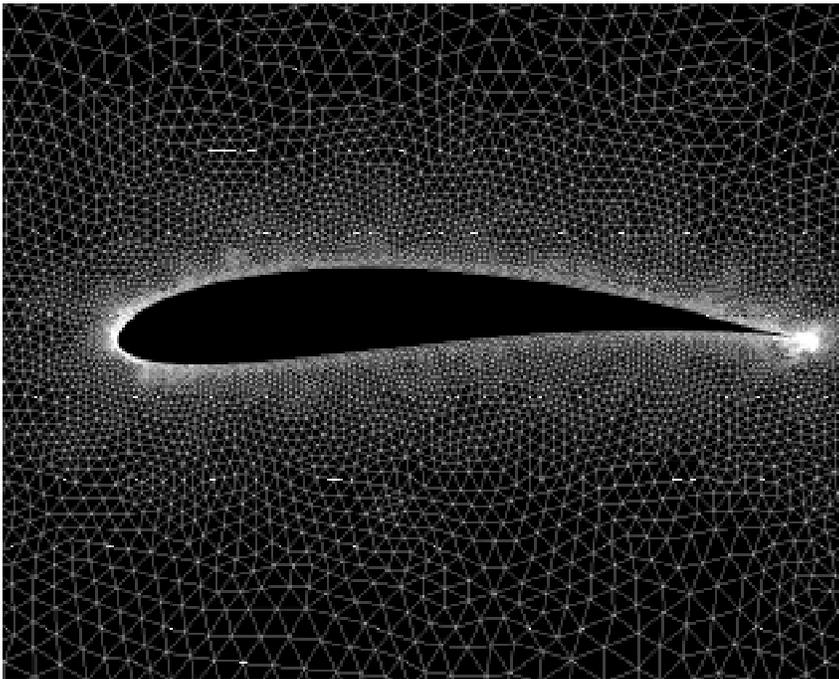


Figure 1

erated automatically for applications with complex geometries or dynamically moving boundaries (but at the cost of higher storage requirements to explicitly store the connectivity information for every point in the mesh).

Using a standard fixed mesh, it may be time-consuming or even impossible for a simulation to resolve fine-scale features. Efficiency can be significantly improved by inserting new grid points in regions that require more resolution, and removing points from regions where less resolution is acceptable. Unstructured grids, by their very nature, facilitate this kind of local, dynamic mesh adaptation. This adaptability allows unstructured meshes to be used to efficiently solve problems with evolving physical structures, such as shock waves, contact discontinuities, vortices, and shear layers.



RUPAK BISWAS & LEONID OLIKER

Figure 2

To compute air speed, direction, pressure, and other parameters around a simulated airfoil, the space is first divided into thousands of triangles forming an "unstructured mesh." Figure 1, left: The mesh after the density of triangles has been increased in specific regions to better simulate fine-scale phenomena such as turbulence. Figure 2, above: The mesh before refinement.

On uniprocessor machines, numerical solutions of such complex, real-life problems can easily require several hours to days, a fact driving the development of increasingly powerful parallel (multi-processor) supercomputers. The unstructured, dynamic nature of many systems worth simulating, however, makes their efficient parallel implementation

a daunting task. This is primarily due to the load imbalance created by the dynamically changing nonuniform grids and the irregular data access patterns. These cause significant communication at runtime, leaving many processors idle and adversely affecting the total execution time.

Nonetheless, many researchers believe that adaptive unstructured-grid techniques will play an important role in future high-performance supercomputing. To minimize the "overhead" associated with a numerical simulation, mesh adaptation and dynamic load balancing must therefore be accomplished rapidly and efficiently.

Recently, three competing parallel architectures have emerged, each with its own set of programming paradigms. The author and his colleague Leonid Oliker, a computer scientist at Lawrence Berkeley National Laboratory, set out to compare the performance of these architectures—distributed memory systems, distributed shared-memory systems, and multithreading systems—on a standard unstructured mesh adaptation problem. Surprisingly, multithreading, the least well-known of the three architectures, offered the best combination of speed, efficiency, and ease of use.

Contrasting Approaches

On distributed-memory systems, each processor has exclusive access to its own local memory. To access data in another processor's memory, a copy of the desired data must be explicitly sent across the network using a message-passing protocol such as Message Passing Interface (MPI) or Parallel Virtual Machine (PVM). To run an application on such machines, the programmer must decide how the data should be distributed among the local memories, communicated between processors during the course of the computation, and reshuffled when necessary. This allows the user to design efficient programs, but at the cost of increased code complexity and programming work.

In distributed shared-memory architectures, each processor has a local memory but also has direct access to all the memory in the system. On the SGI Origin2000 computer, for example, each processor uses a local cache to fetch and store data. Cache coherence is managed by hardware, but since the shared memory is physically distributed, the memory access times are nonuniform. Such computers are therefore classified as cache-coherent nonuniform memory access (CC-NUMA) machines. Parallel programs are relatively simpler to implement on such systems since each processor has a global view of the entire memory. Parallelism can be achieved easily by inserting directives into the code to distribute steps in a program among the processors. However, the portability of code between machines may be diminished, and sophisticated cache management techniques may be necessary to enhance parallel performance.

Multithreading is a radically new concept in parallel computing. The hardware of traditional monothreaded computers can process only a single thread (a short sequence of instructions)

Parallel computing at a glance

distributed memory: each processor "owns" its local memory, and data is shared between processors via message-passing protocols

shared memory: all processors have access to all memory locations. Easy to program, but processors must be prevented from modifying the same memory location simultaneously

multithreading: while a blocked thread (a short sequence of instructions) waits for data from memory, a processor can switch to a ready thread, maximizing efficiency

Continued on next page

Continued from previous page

at a time. As a result, such computers remain idle while waiting for information to arrive from memory. Multithreaded processors, on the other hand, can tolerate memory latency and utilize substantially more of their computing power by processing several threads of computation at once. For example, each processor in the Tera MTA supercomputer, manufactured by Tera Computer Company, has hardware to support up to 128 threads.

This multithreaded architecture is especially well-suited for irregular and dynamic applications such as unstructured-mesh simulations. Unlike CC-NUMA machines, multithreaded computers have large uniform shared memory and no data cache, and are completely insensitive to data placement. Parallel programmability is therefore simplified since the user has a global view of the memory and need not be concerned with the data layout.

Surprisingly, multithreading, the least well-known of the three architectures, offered the best combination of speed, efficiency, and ease of use.

Programming pluses and minuses

Using distributed-memory machines and/or the message-passing paradigm demand the greatest programming effort for

almost all but the most easily parallelizable applications. Data has to be explicitly spread across processors, and special data structures have to be created and consistently maintained for objects lying on partition boundaries. For a computational fluid dynamics application, this implies a high bookkeeping overhead for all the shared vertices and edges of the mesh. In addition, simulations using adaptive methods can be slowed by load imbalances among processors because of the changing nature of the grids. In our current study, we used the PLUM global load balancer (Parallel Load Balancing for Adaptive Unstructured Meshes, also developed by Oliker and Biswas) to balance the workload, minimize the runtime interprocessor communication, redistribute the mesh after each adaptation, and maintain consistent information about the mesh across all the processors. Thus, unstructured mesh adaptation in a message-passing environment incurs several additional costs.

The shared-memory programming paradigm is generally much simpler than message-passing because all memory in the system is equally accessible to the software, even if the access time is nonuniform. Unfortunately, the user must ensure that multiple processors do not simultaneously modify the same memory location. The simplest prevention strategy is to use low-level locks that allow only one processor to modify critical values at a time. However, for the dynamic unstructured application used by the author and his collaborator to test the three architectures, it was more effi-



TERA COMPUTER COMPANY

The Tera MTA supercomputer, made by Tera Computer Company

cient to decompose the triangles of the computational mesh into independent sets such that members did not share edges or vertices. All processors then work together on all the triangles in the same set.

Dynamic load balancing on shared-memory machines, meanwhile, is easily achieved through compiler directives.

Two overheads are associated with this second strategy: the need to insert newly-generated triangles due to refinement into sets, and a global synchronization between the processing of individual sets. But because the memory is globally accessible, no explicit data redistribution is required.

All three architectures experience latency, a wait while data is transferred to and from memory. But while shared-memory architectures use data caches to hide latency, a multithreaded processor switches to a ready thread as a blocked thread waits for its memory reference to complete. The larger the number of concurrent computational threads, the greater the performance. A flat, uniform shared-memory without cache thus removes many of the performance drawbacks associated with CC-NUMA machines for irregular applications.

On the Tera MTA, all memory addresses are randomized by the hardware so that it is impossible to control data placement. Also, an MTA processor switches with zero overhead among active threads at every clock period even if a thread is not blocked. Synchronization among threads is provided by the memory itself without operating system intervention. Synchronization may stall one of the threads, but not the processor on which the threads are running. Once a code has been written in the multithreaded model, no additional work is required to run it on multiple processors, since there is no difference between uni- and multiprocessor parallelism. Low-level locks are retained in the multithreaded implementation of the mesh adaptation code to ensure that adjacent triangles are not updated simultaneously. But no partitioning, data redistribution, independent sets, or explicit load balancing are required, greatly simplifying the programming effort.

Architectures Under the Lens

A standard computational mesh simulating flow over an airfoil was used for benchmarking experiments comparing the three parallel architectures. The coarse initial mesh consists of 14,605 vertices and 28,404 triangles (*see Figure 2, page 13*). Mesh refinement is usually required around the leading edge of the airfoil. At transonic Mach numbers, shock waves form on both the upper and lower surfaces of

the airfoil which then propagate to the far field. This scenario was approximated by manually specifying the level of adaptation in these regions. This allowed the performance of the unstructured mesh adaptation and load balancing algorithms to be measured without first running the full simulation to find areas of interest.

In the test runs, the initial mesh was refined a total of five times to generate a mesh consisting of 488,574 vertices and 1,291,834 triangles (see Figure 1, page 12). Table I, below, summarizes the performance of the parallel, dynamically adapting, unstructured mesh application on various platforms using different programming paradigms.

It is important to reiterate that the irregular, adaptive nature of such algorithms makes it particularly difficult to achieve good performance. For reference purposes, the timing for the original serial mesh adaptation code is reported. It consists of approximately 1,300 lines of C, and requires 6.4 seconds to execute the benchmark simulation on a 250 MHz MIPS R10000 processor.

The message-passing version demanded the greatest programming effort, almost doubling the size of the original serial code. Significant additional memory was also needed, mainly for the send and receive buffers for the bulk communication during the redistribution phase. Despite these drawbacks, the message-passing code showed reasonable scalability and can be easily ported to any multiprocessor system supporting MPI. The best time on the 640-node CRAY T3E owned by the National Energy Research Supercomputing Center at Lawrence Berkeley National Laboratory was 3.0 seconds when running the simulation on 160 processors. The adaptation application required only 0.61 seconds, but the partitioning and data redistribution times were 1.7 seconds and 0.69 seconds, respectively. With more processors, the refinement and redistribution times continued to decrease, but this was more than offset by the increase in the partitioning time.

The best time on an SGI Origin2000 at the NAS Systems Division of NASA Ames Research Center was 5.4 seconds when running the benchmark on 64 processors. Again, partitioning required 2.3 seconds, and the runtime trend indicated that using more processors would neutralize the decrease in the refinement and redistribution times. The shared-memory directive-based version required far less programming effort than the MPI implementation and

had a low memory overhead. This code can be ported to any system supporting a global address space. Unfortunately, the fine-grained nature of these computations resulted in poor cache reuse and significant overhead due to false sharing. The best time of 39.6 seconds was obtained with 8 processors. Mesh refinement required 17.0 seconds, while the remaining time was spent in decomposing the triangles of the mesh into independent sets to prevent two or more processors from writing into the same memory location. This forced adjacent triangles to be in different sets, dramatically increasing the cache miss rate.

The Tera MTA at the San Diego Supercomputing Center handled the

Programming Paradigm	System	Best Time (# of Processors)	Code Increase	Memory Increase	Scalability	Portability
Serial	R10000	6.4 (P=1)				
MPI	Cray T3E	3.0 (P=160)	100%	70%	Medium	High
MPI	SGI O2K	5.4 (P=64)	100%	70%	Medium	High
Compiler Directives	SGI O2K	39.6 (P=8)	10%	5%	None	Medium
Multithreading Directives	Tera MTA	0.35 (P=8)	2%	7%	High	Low

Table I: Performance by Paradigm. Each architecture was tested using an adaptive mesh refinement algorithm designed to increase the density of triangles in specified regions of a two-dimensional unstructured mesh. Note that the timings for the various parallel versions, which are the best times for any number of processors tested, include the overhead of dynamic load balancing (this is absent in the serial version). It is also important to note that different parallel versions use different dynamic load balancing strategies. Details are given in the text of this article and in a technical paper to be presented at the SC99 conference in November. The best wall-clock time of 39.6 seconds for the directive-based version on the Origin2000 (O2K) was unexpectedly slow. Several reasons contribute to this poor parallel performance. Since triangles are processed a set at a time (triangles in a set do not share edges or vertices), the cache miss rate increased dramatically from the serial case where all triangles are processed in the natural order, in which subsequent triangles are usually close together in memory. Another consequence of the poor data locality is the significant overhead that is incurred when an accessed memory location is not on the list of most recently used pages. This is known as a "TLB miss" and causes the operating system to call a relatively expensive routine that finds the physical address in a memory table. Poor parallel performance also stems from the structure of the code which, for programming simplicity, assumes a flat shared-memory model and does not consider data locality or cache effects. When triangles of a particular set are being processed, each processor refines distinct triangles that have non-overlapping edges and vertices. However, since data structures are not explicitly reordered, cache lines contain mesh objects that may be required by several processors simultaneously. This "false sharing" problem is exacerbated when new mesh objects are created during refinement. Each time a new word is written to cache, all copies of that cache line residing on other nodes are invalidated. The hardware is therefore overloaded attempting to maintain cache coherency in this environment. The author, who can be contacted at rbiswas@nas.nasa.gov, welcomes suggestions on improving the code's performance in the CC-NUMA environment and is happy to share the code with colleagues.

Continued on Page 26

Chess competition puts 256-processor Origin2000 to the test

BY HOLLY AMUNDSON

No medals went to a team of MIT computer scientists taking part in the Ninth World Computer-Chess Competition in June, but their fourth-place finish did come with a reward. The group was able to test-drive its computer chess program on Steger, the 256-processor SGI Origin2000 supercomputer at the NAS Systems Division. At the time of the competition, the Steger system was the world's largest shared-memory, single-system-image, parallel supercomputer, lending the team not only considerable

computing power but also an opportunity to study the performance of their

computer-chess program, Cilkchess, in a massively parallel environment.

Running their computer chess application on the 256-processor machine was an important learning experience, team members say.

"You learn what the real problems are," says project lead Charles Leiserson of MIT's Laboratory for Computer Science (LCS).

Despite a few minor glitches during the competition, the MIT team racked up more points than 26 other competitors, falling short of first place by only half a point. Ac-

cording to Leiserson, the team gained "invaluable" experience with memory handling, latency hiding, and other technical factors affecting the speed of parallel computations. "The hands-on experience allowed us to identify issues with the memory allocation manager in the SGI system, leading us down new avenues of research," Leiserson says. Engineers and system administrators at the NAS Facility also found the competition enlightening. "Cilkchess really stressed the 256-processor system," helping to expose previously unrecognized problems, says Jens Petersohn, a NAS systems engineer who worked with the MIT team.

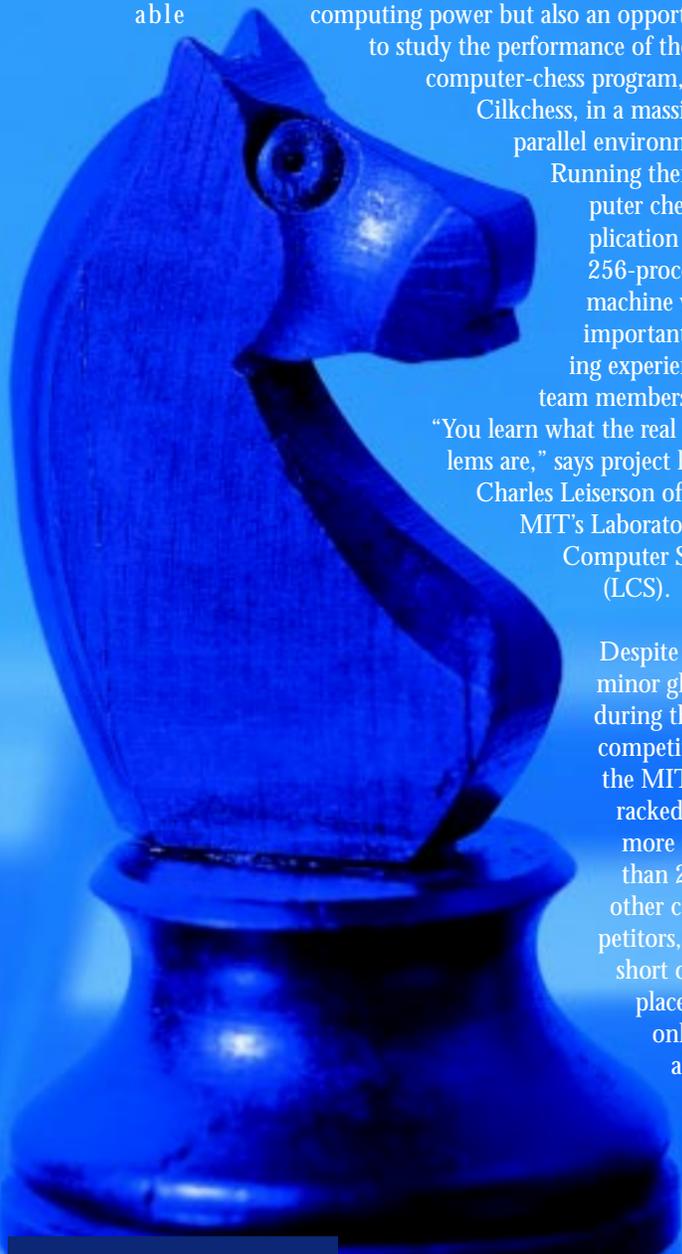


Cilk makes the grade

Chess is more than a game to Leiserson and his colleagues at MIT. To them, it's also an instrument for demonstrating the performance power of Cilk, the computer language used to compose Cilkchess.

Members of the Supercomputing Technologies group at LCS created Cilk, an extension of the common programming language C, to address challenges in the parallel programming of dynamic, irregular applications. As Leiserson explains, it is often difficult to predetermine how much work will fall to each processor in parallel computations (such as those required in a chess game). As a result, some processors in a traditional parallel programming environment become overworked, creating bottlenecks that slow down the entire program. Cilk is specially equipped to handle unpredictable computational structures that don't fit the regular patterns found in traditional parallel programs. To make efficient use of multiple processors, applications written in the Cilk language call on two techniques: multithreading and work stealing. Multithreaded programs divide themselves into multiple "threads," or short sequences of instructions, to solve a single problem (*see article, page 12*). Available processors "steal" these threads automatically, without explicit instruction by the programmer. The result is a more balanced spreading of computations to unused processors.

Cilk is being used by researchers across the United States for applications such as graphics rendering, artificial evolution studies, and simulations of protein folding and galaxy formation. Professor Bonnie Berger of LCS and the MIT



mathematics department, for example, has used the Cilk programming language to build a simulator for biological virus shells.

Don Dailey, a system administrator for the Supercomputing Technologies Division at LCS, was asked to join the Cilkchess team at MIT after winning the 1993 International Computer Chess Championships in Indianapolis with his program Star Socrates, an early version of Cilkchess. "I have loved chess ever since I was a little kid," Dailey says. "The development of Cilkchess started as a hobby." Very quickly, however, Cilkchess became a vehicle for showcasing Cilk-5, the most recent version of the language. "The Cilk language makes it very easy to write a parallel program like Cilkchess," adds Dailey.

Work-stealing algorithms, says Dailey, turn out to be a good way to accommodate the varying workload generated during a chess game, where moves are unpredictable and cannot be assigned in advanced to certain processors. Early in a game, it is often unclear which team has the better position. Sometimes, the positions are unbalanced, with the two sides showing very different strengths and weaknesses, and with frequent confrontations between high- and low-ranking pieces. These differences often lead to interesting play, but in such cases predicting the outcome of the game is nearly impossible. "Cilkchess excels when the position is unbalanced," Dailey boasts.



Bigger is better

In 1996 the Cilkchess team placed first in the Dutch Open Computer Chess Championship, running the program on a 12-processor Sun UltraSPARC Enterprise 5000 server. The team finished second in the same competition in 1997 and 1998,

using a 4-processor DEC Alpha Server 4100 in 1997 and a 64-processor SGI Origin2000 at Boston University's Center for Computational Science in 1998.

For the 1999 World Chess Competition in Germany, Leiserson invited a number of possible sponsors to work with the MIT computer team. The organization offering the largest machine, Leiserson proposed, would be the group's choice for the competition. SGI and the NAS Systems Division won the bidding with the offer of Steger, NASA's largest multiprocessor, shared-memory computer. SGI, which claimed 5 percent of the computing time on Steger as part of its 1998 agreement with the Division to build the machine, donated all of its time to the MIT group.

Assembled at the NAS Facility in November 1998, Steger is twice the size of its largest predecessors, the two 128-processor Origin2000 computers at the division. NASA and SGI joined forces on the project in order to explore the behavior of very large parallel systems. In memory size and speed, the Steger system ranks high above the CRAY C90 and J90

supercomputers at NAS; it contains 64 gigabytes of memory and 1.33 terabytes of disk space, and can reach speeds in excess of 20 GFLOPS (billions of floating point operations per second) on real applications (four to five times faster than von Neumann, the 16-CPU CRAY C90 at NAS). Currently, over 700 regular users share time slots on Steger, including the Boeing Company and a team of engineers at NASA Langley Research Center who are designing a new reusable launch vehicle.

For applications such as Cilkchess that create unpredictable, rapidly varying workloads, Petersohn points out, the 256-processor system has a significant edge over serial computers or computers with only a few processors. According to Leiserson, Cilkchess was so well adapted for a massively parallel system that the program required no major changes to run on Steger, merely small, "standard adjustments" to initialization scripts.



Cilkchess vs. the world

Skillful programming can take a computer chess team only so far, Leiserson notes. Success also depends on the "deep search capacity" of the program. During a game of chess, each move generates a new set of possible countermoves and counter-countermoves, a phenomenon termed a "game tree." With additional moves, the number of possibilities increases at an exponential rate. The depth to which the program can look down different branches of the game tree during play defines the deep search capacity of that particular machine. The deeper the search capacity, the stronger the program.

After their second-place showing in 1998 at the city of Leiden in the Netherlands, the MIT team was eager to travel to Germany to participate in the 1999 world championship. Even before leaving, however, the group faced a few challenges. During practice runs on Steger, the Cilkchess team encountered issues related to the Origin2000's operating system. Steger allocates memory in such a way that certain processors can become overloaded with initialization tasks. The uneven data initialization slowed the team's initial progress, but NAS engineers helped the MIT group work out the problem.

The competition itself was computer-versus-computer, but the event retained the flavor of a traditional chess tournament. Participants tracked their programs' moves with physical boards and regulation chess clocks. A majority of the participants ran their programs on microprocessors, enabling them to bring their own hardware and software to Germany. Several teams, including Cilkchess, used plain terminals linked to their remote computing resources. Chess fans were able to follow the event in real time on web sites created by LCS and the organizers of the world chess

Continued on next page

Continued from previous page
championship.

Playing against a program called Junior in the third round of the competition, Cilkchess ran into some bad luck. The team's rook became trapped as a result of the dreaded "horizon effect," the inability of a chess program or its computer platform to search far enough into the future to predict the moves of its opponent. "A machine does not have the common sense that a human has. A person could easily see something like this [the rook's predicament]," explains Leiserson. Despite any minor glitches in Steger's execution, however, Leiserson maintains a positive regard for the SGI system. "Steger behaved very well, and both NASA and SGI

engineers provided wonderful technical support," he says.

Game trees

Key to a winning chess program is 'deep searching,' the ability to explore many alternative branches in a tree of possibility and weigh each move and countermove.

The horizon effect
During the third round of the competition, Cilkchess fell prey to the 'horizon effect,' an occasional inability to see beyond the next few moves. Its competitor, Junior, moved in for the kill.



doing

Working with Steger gave the Cilkchess team specific insight into memory allocation, data locality, latency hiding, and other important aspects of parallel computing. Memory allocation, says Leiserson, is one of the biggest questions to tackle when running applications on non-uniform, shared-memory multi-processor computers such as Steger. Where can data be stored and how should it be assigned to that storage space? "This experience with

In the end, first place in the competition went to Shredder, a microprocessor-based program created by German programmer Stefan Meyer-Kahler. Teams from the United States and from Germany placed second and third, respectively. The MIT team's fourth-place showing was a bit disappointing, but by no means devastating, according to Leiserson. Win or lose, collaborations similar to the one with the NAS division are "the best way for [our] group to improve upon Cilk and Cilkchess," he says.

Learning by



Cilkchess creator Don Dailey (left) and opponent at the 1999 World Computer Chess Competition in Paderborn, Germany

memory allocation has led to some nice research and some new algorithms that we wouldn't have been motivated to study without having confronted the problem in the field," says Leiserson.

An important characteristic of a chess program is its ability to quickly search and retrieve data. Programmers typically try to reduce latency or communication delays by directing that data be stored in memory locations physically close to the processors using the data. But Steger's caching hardware provided the team with a new way to speed up searches. A cache retrieves large blocks of data, called cache lines, in nearly the same time it would take to retrieve a single piece of data. Cache space is created rather than directly allocated, and filled only when it is physically needed by the system. Connection time between memory and processors is therefore reduced, an effect known as latency hiding. "Running such a dislocalized program as Cilkchess confirmed Steger's ability to effectively hide memory latency," says Petersohn.

But this wasn't the only useful feedback received by NAS engineers. The MIT team also uncovered a pre-existing bug that was causing a "hot-lock problem," in Petersohn's words. This phenomenon occurs when many processors are waiting to acquire exclusive access to the same data. While one processor accesses the data, all of the other waiting processors execute the same instructions over and over again. "This problem was causing a substantial slow-down in the system," Petersohn says. In light of this discovery, NAS engineers installed an updated version of the operating system, resulting in a significant performance gain.

Currently, NAS researchers are collaborating with SGI to develop an even larger Origin2000 system including 512 separate processors (*see story, page 2*). Dailey says he would jump at the chance to use the 512-processor machine in next year's computer chess competition. "More processors mean higher performance," he enthuses. Petersohn cautions,

however, that adding more processors to the picture could actually aggravate the problem of memory allocation.

Either way, Dailey is preparing Cilkchess for its next match, implementing a new function he calls “temporal difference learning.” In this approach, each piece and location on the chessboard is assigned a number or weight of importance. The computer plays a series of games with itself, adding up the weights to become familiar with good plays in any given situation. Asked whether his team plans to enter the improved Cilkchess in the next World Computer-Chess Competition, Leiserson is enthusiastic: “We’re always ready for a battle!”

Holly Amundson is an intern in the NAS publications and media group. She majors in materials science and engineering at California Polytechnic State University in San Luis Obispo.



Globus: An Infrastructure for the Information Power Grid

BY PETER ADAMS

In October, thirty years to the month after the birth of the Internet, computer scientists reached another milestone—one that could prove as significant as the Internet's debut. Researchers at four NASA research centers, the University of Southern California's Information Sciences Institute (ISI), the Energy Department's Argonne National Laboratory in Illinois, and several other institutions inaugurated a fully functional prototype of a new "super-Internet" for high performance computing. The Information Power Grid (IPG), as the network is known, joins supercomputers and storage devices owned by the participating organizations into a single, seamless computing environment.

Researchers are already exploring the IPG's capabilities and limitations. The testbed allows remote operation of computers and scientific instruments, matching of local jobs with distant computing resources, and allotment of computations too demanding for a single supercomputing center to handle. "This is just the beginning of our progress toward simple, powerful distributed computing," says Bill Thigpen, chief of the NAS Engineering Branch.

These first steps were made possible by Globus*, a set of software utilities that allows supercomputers and other de-

* *There is no relationship between the Globus metacomputing toolkit and NAS researcher Al Globus, featured in the article on page 4.*

vices thousands of miles apart to communicate and cooperate. The philosophy behind Globus strongly echoes the architecture of the early Internet, giving the IPG project the heady air of history repeating itself. Defense researchers created the Internet (then called the Arpanet) by interposing phone lines and devices called "Interface Message Processors" (IMPs) between existing computers. (When the second IMP was installed at Stanford Research Institute on October 1, 1969, it turned a single node at the University of California, Los Angeles, into part of a true network. The first packets of information were exchanged between UCLA and Stanford four weeks later, on October 29.) Similarly, Globus mediates between the existing computers and job-scheduling systems at IPG partner sites. With a few extra keystrokes, a researcher can submit a job at one site and specify that it be computed at any other site, as if the IPG were a single giant computer.

Many details must still be refined, but NAS researchers say Globus and the IPG form such a close fit that they should have no trouble meeting the IPG effort's ambitious two-year and five-year goals. "The functional testbed is an important milestone for our two-year goal of a prototype production environment," says Bill Johnston, project manager for the IPG.

Managing heterogeneity

The Globus Project got its start at the SC '95 conference in San Diego, where Ian Foster of Argonne National Laboratory and Carl Kesselman of the University of Southern California's Information Sciences Institute demonstrated

More than 40 super-computing centers have installed Globus on their systems, creating a testbed that spans three continents (left). NASA Ames Research Center is a charter member of the group, known as the Globus Ubiquitous Supercomputing Testbed Organization (GUSTO). The NASA Information Power Grid, which includes Ames as well as Goddard Space Flight Center, Glenn Research Center, and Langley Research Center complements and extends the GUSTO network.

that linked high-performance computers could work together well on a single job. These two remain the project's co-principal investigators, and in five years Globus has grown to link more than 50 sites across the country in the Globus Ubiquitous Supercomputing Testbed Organization (GUSTO).

In the construction of an environment like GUSTO or the IPG, the bewildering variety of hardware and software used by the nation's

supercomputing centers makes an intermediary such as Globus indispensable. The situation is similar to a session of the United Nations General Assembly, which can include hundreds of people speaking dozens of languages that are mutually unintelligible. In order for the Assembly to function, its members must somehow be able to understand each other and, in turn, be able to make themselves understood.

The high performance computing community exists in much the same state of heterogeneity, only here the diversity resides in the system requirements of resources as varied as supercomputers, storage systems, wind tunnels, telescopes, and medical imaging devices. Users at one node on the IPG—which currently links NASA facilities at Ames, Langley, and Glenn Research Centers with several academic institutions—may know nothing about operating the systems available at the other nodes.

To solve its language barrier, the UN employs a skilled group of translators who can simultaneously listen to a speech in one language and repeat it in a second tongue. Instead of arbitrarily choosing one language for its meetings, forcing each and every member to learn to speak it fluently, the UN relies on this middle layer of translators to bear the workload and keep the communication process between speaker and listener as seamless and unobtrusive as possible.

'This is just the beginning of our progress toward simple, powerful, distributed computing... We're learning what the future will be as we progress.'

A comparable intermediate layer, with a similar emphasis on transparency and utility, plays a key role in the ongoing development of the Grid. In Grid terminology, this is "middleware," or, as IPG architect Bill Nitzberg of the NAS Systems Division puts it, "a common layer of resources that various users of different applications and interfaces can access." This common layer then acts on behalf

of the user, communicating with a heterogeneous set of resources that may be down the hall or thousands of miles away. For the IPG, middleware starts with Globus.

Linking the sites

In practice, Globus consists not of one monolithic software package, but of several customizable tools from which sites can select in order to manage heterogeneity. This adaptability, in combination with Globus's widespread use on the GUSTO testbed, made it a good choice as the infrastructure of the IPG's middle layer, says Nitzberg. "When NASA set about defining the IPG project, Globus clearly became one of the core components," he says. The NAS Systems Division now provides the Globus project with research grants totaling about \$1.25 million per year.

The advantage of using middleware is that regardless of the growth or mutability of the grid, local users remain capable of running jobs. Nitzberg explains, "Globus lets people work in their own world, whether that world is astrophysics or aerospace design." Even disparate disciplines use some of the same resources, and Globus allows the users to gain access to more or better machines while ignoring the additional complexity. "Basically, we don't want our users to have to learn anything new," says Nitzberg.

To ensure this ease of use, much of the Globus-related work at NAS and the other NASA sites involves taking standard tools and building in more specific IPG needs. Among the things Nitzberg and other IPG architects are incorporating from the Globus software toolkit are a security infrastructure considered superior to that provided by the UNIX or NT operating systems; an interface for executing actions remotely; a capability for accessing data and manipulating files remotely; an infrastructure for fault monitoring; and an information service describing the state of grid resources. These tools serve as a framework upon which further middleware, such as systems for job queuing and problem routing, will be placed, says Mary Hultquist, a member of the NAS grid environments group.

The University of Southern California's ISI and the NCSA Alliance have used Globus for production computing work for some time. With the creation of the IPG, many more investigators outside the field of distributed computing will regularly employ Globus. "In a GUSTO research environment, you only have one or two people at a site who use a Globus tool, and they tend to know it quite well," explains Nitzberg. "Here, we'll have new users reading the documen-

Continued on next page

Continued from previous page

tation and, never having seen Globus before, trying to make it work.” To speed up learning, publication specialists at NAS have written a “Quick Start” guide, the first in a planned series of official Globus documents.

Accommodating users

The commissioning of the IPG testbed marks the completion of a major short-term goal of the effort. Thigpen calls it a “a proof of concept” that “gives us a group of systems as a springboard for further work.” But the establishment of a technological infrastructure is just half the job. The other half involves successfully merging the needs of several major sites accustomed to acting independently.

‘In a research

Some of the inter-site issues that must be addressed by any metacomputing system are relatively straightforward, such as identifying the different computing and personnel profiles of each site. NAS, for instance, “is the smallest site in terms of computing power [committed to IPG], but is the largest contributor of people-hours to the IPG project,” says Hultquist. Subjects such as troubleshooting mechanisms are more complex. “If users have trouble with resources they are accessing, they need to be able to pick up the phone and talk to someone who can fix it,” says Nitzberg, “This isn’t a problem at NAS, where the computers are yards away, but how do you ensure the problem is routed, traced, and reported correctly when the resource is 3000 miles away?”

Discussions of inter-site relations can descend to an incredible level of detail, such as the exact wording of an account request form, says Hultquist, who is deeply involved in this process. While these may seem like simple issues, she says, dependable middleware and a common system of administration are vital for the smooth working of the IPG. “I try to look at issues from a user’s perspective,” she explains, “but sometimes the only way to test these decisions is to watch the testbed and see what happens. The key is to provide grid functionality without adversely affecting users, and to design policies that make becoming a part of the Grid easier.”

Long-term goals

The future of both Globus and the IPG looks bright as the Grid paradigm gains ground throughout the field of high performance computing. Thigpen cautions, however, that a large-scale, persistent, high-speed supercomputing network remains a distant goal. Building the IPG is a process of “moving toward what we believe is the future, and learning what that future *really* will be as we progress,” he says.

Because supercomputing sites already exploit their machines to the greatest extent possible, simply linking them to a grid is not going to generate free computing cycles. Instead, the work being done now will move the field closer to the elusive goal of full utilization. The truly revolutionary benefits will come, says Thigpen, when it is simple to run a computing job regardless of geography. “Right now we’re very con-

scious of what machine we’re running jobs on and where that machine is located,” he says. “But the future of the Grid will take us to an environment where we can [specify] what we need and have the job run anywhere.”

Meanwhile, users at each of the IPG sites are pursuing their own scientific problems, and are at the same time providing invaluable information to the IPG’s builders. “By writing Grid-enabled code, they fault-test our work and let us know what we need to fix,” says Hultquist. “At the same time they are preparing themselves for the future of computing.”

Peter Adams is a senior majoring in history at Williams College in Massachusetts. He has worked as a summer intern for the NAS publications and media group for the last three years.

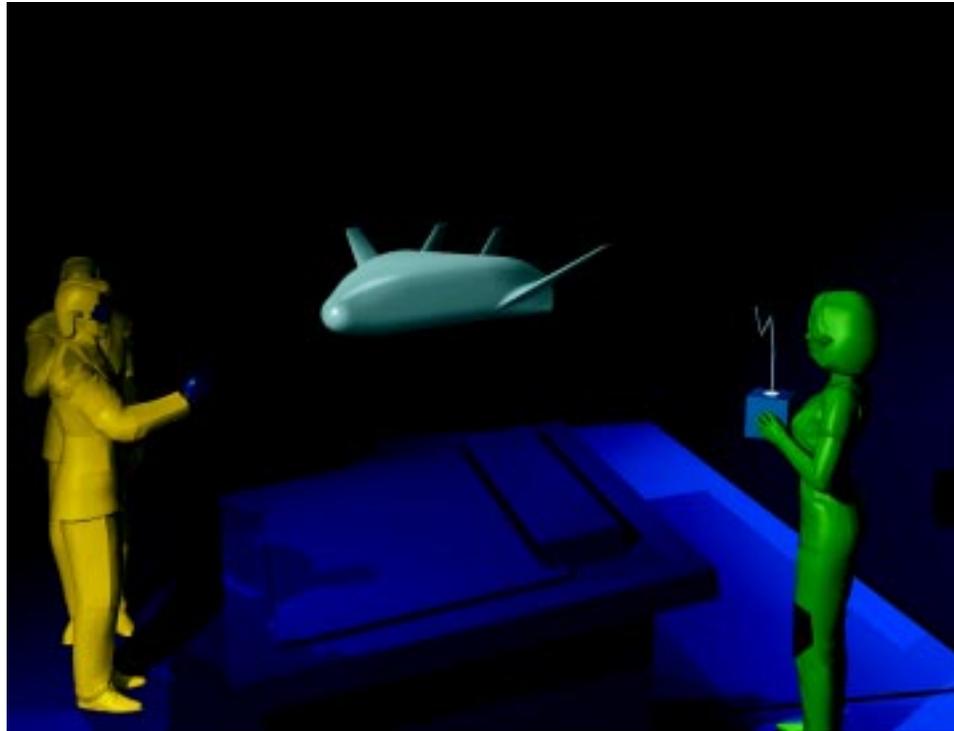
Distance no barrier in distributed, collaborative Virtual Wind Tunnel

BY HOLLY AMUNDSON

When NASA established the Numerical Aerodynamic Simulation program in the early 1980s, it charged the organization with accurately predicting and reproducing the aerodynamic performance of a whole aircraft using computers.* To make these predictions useful, the program was also directed to study new ways of visualizing the huge datasets resulting from these simulations, perhaps including virtual reality systems that would allow an aerospace engineer to “enter” and explore three-dimensional representations of this data. By the mid-1990s, the Virtual Wind Tunnel (VWT) project in the NAS data analysis group had largely fulfilled this goal. One goal for NAS that had *not* been spelled out, however, was that these virtual environments should accommodate whole crowds of people, or that researchers should be able to participate in collaborate virtual work sessions from great distances. Those goals are new, formulated over the last several years by David Whitney, designer of the VWT’s initial distributed infrastructure, NAS data analysis group lead Steve Bryson, and colleague Bryan Green.

The VWT software creates a fully immersive virtual environment where researchers can interact with data spread over complex three-dimensional fields. Inside the collaborative workspace, this data can be portrayed in many different forms and can be examined from any point of view. “Virtual prototyping” software using some of the same principles as

* In 1996, the program was renamed the Numerical Aerospace Simulation Systems Division.



NAS SYSTEMS DIVISION

Researchers collaboratively study a three-dimensional simulation of a reusable launch vehicle in flight. Scenes such as this may become common as the aerospace industry adopts the distributed, collaborative technologies embodied in the NAS Virtual Wind Tunnel.

the VWT has been available to the public for some time, but none of the commercial products can handle the massive datasets created by high-fidelity computational fluid dynamics (CFD) applications. The VWT software was specifically designed for datasets up to 200 GB in size.

In 1998 Green and Whitney, who is now a software developer at Amazon.com, began modifying the VWT’s architecture to allow several people at geographically distant locations to participate in the same virtual session. The new “Distributed, Collaborative VWT” passed its first major test in December 1998, in a demonstration linking two users at NASA Headquarters in Washington, D.C., with two users at the NAS Facility in California.

Continued on next page

Continued from previous page

The system, which will be demonstrated again in November at the SC99 High Performance Networking and Computing Conference in Portland, Oregon, is exactly the kind of application NASA computer scientists have in mind as they construct the Information Power Grid (*see story, page 20*). This distributed supercomputing network will greatly expand the data gathering, computing, storage, and collaboration opportunities available to researchers through their home terminals.

The collaborative VWT is expressly intended to help widely separated researchers explore complex datasets together.

When multiple users are working collaboratively inside the same simulation, a single server continues to calculate and render each scene, while client workstations merely display the results.

From mono to multi

In 1989 Bryson, then a member of the Ames Human Factors Division, and colleague Creon Levit began work on the VWT. At first, the system was intended to run on a single computer and to support a single user. In 1992, two years after joining NAS, Bryson and associate Michael Gerald-Yamasaki devised a way to distribute the computational work needed to

create and update the virtual scenes to several computers or workstations. Their trick was to assign a central server to calculate the geometry of each scene, while letting a separate workstation run the application viewer. This arrangement led directly to Whitney, Bryson, and Green's work over the past two years to enable several viewers to run simultaneously, each with its own point of view. When multiple users are working collaboratively inside the same simulation, a single server continues to compute the geometry, while client workstations render the geometry and display the results according to the individual's viewpoint. "The collaborative aspect was a natural extension of the distributive part," says Green, the project's visualization software engineer.

But "natural" doesn't mean "easy." For every client added to the virtual workspace, more network bandwidth is needed to handle the transfer of geometry from server to client. Multicasting, one of the possible solutions to this problem, is currently under consideration by Bryson and Green. Under multicasting, the geometry is transmitted only once and all users receive the updated material, therefore decreasing the bandwidth necessary to support any given scene. "It's the difference between sending the geometry once total and once per client," Green explains. "Successful implementation of multicasting would help increase performance of the VWT application."

Behind the Scenes

Multicasting is especially useful in a collaborative application like the VWT, because there's a lot happening behind the scenes. The software converts massive columns of numbers (a CFD dataset) into three-dimensional flowfields around a

digital model of an aerospace vehicle. The data are displayed in stereo to create the illusion of depth. Immersive displays, such as the Virtual Workbench in the NAS Visualization Laboratory, put users directly in the path of these flowfields. The user's stereo visor tracks all head movements, causing the scene to shift accordingly in near-real time. Using various manipulation tools, the user can move about within the flowfield and examine the complex structures, such as wake vortices, that arise in time-varying flow simulations. Structures can be picked out and manipulated using a pen-shaped "wand" whose position is also tracked in three dimensions. (With all of this movement going on, Bryson notes, it's important that the computer re-render the scene in 1/10 of a second or less. A longer rendering period creates a discrepancy between hand movement and visual feedback, destroying the virtual reality effect.)

Inside the VWT, flow information can be viewed in many different forms, depending on the viewer's needs. Data can be displayed in the form of streamlines, streaklines, cutting planes, particle paths, isosurfaces, tufts, or simple numerical displays. One of the most distinctive tools is the rake, a moveable line that emits particles into the flowfield, depicting changes in the fluid flow in a particular region. Another tool, the plane, contains nine grab points enabling the user to resize and reorient the tool until it displays the desired cross section. The user also has access to parameters such as pressure, density, and temperature at every point in the flowfield.

Donning a stereo visor for the first time, the VWT user enters into a world of virtual wonders. With wand in hand, the client may rotate the simulated aerospace vehicle, confronting it head-on or from any other desired angle. Various tools dot the virtual world, waiting to be grabbed by the floating 3-D cross hairs that represent the client's hand. Each tool enables the user to manipulate the flowfields within the virtual workspace differently; one tool, for example, paints the simulated world with an array of colored lines showing where particles would flow according to the laws of fluid dynamics. A rotation of the wrist may land the user in the midst of these lines, face to face with the nose of a computer-generated launch vehicle.

But while the VWT is visually stimulating and fun to play with, it's also a serious research tool for aerospace engineers.

A successful December 1998 demonstration linking Distributed Collaborative Virtual Wind Tunnel users at NASA Headquarters in Washington, D.C., with others at NASA Ames marked an important milestone for the project: It was the first time the VWT's capabilities had been tested on a cross-country network.

It offers a quick and easy way for researchers to explore the behavior of flow patterns at any point around the craft, discovering irregularities that would be impossible to spot under normal conditions.

Real-world tests

As with any piece of complex hardware or software, testing of the VWT has revealed a few glitches and raised interesting new questions, according to Bryson. With multiple users inhabiting the collaborative workspace, for example, the space can become cluttered with tools. Each client can see all the other clients' tools moving around, making it hard to follow events. Preventing this kind of sensory overload when multiple participants are using a virtual space is clearly a problem that will need to be addressed in the VWT and in future collaborative systems, Bryson says. The software also needs to handle conflicts between users' actions more smoothly; right now, when two clients grab a tool at the same time, there is a delay before the "loser" is notified. And when users move around massive amounts of geometry, network traffic jams can occur, increasing the time it takes to re-render each scene. "The more geometry there is to be moved around, the slower it's going to be," remarks Green.

But these problems are mostly minor, as the successful December 1998 demonstration linking NASA Headquarters with NASA Ames demonstrated. This presentation marked an important milestone for the project, says Bryson, since it was the first time the VWT's capabilities had been tested over a cross-country network connection. The VWT application had another important demonstration in Washington last June, as one of only two projects chosen to represent NASA in a presentation of U.S. government information technologies to members of Congress. U.S. Navy officials have also used the VWT and are evaluating it as a potential tool for submarine simulations, according to Green. Regular users of the system, meanwhile, include The Boeing Company and NASA's Ames Research Center, Goddard Space Flight Center, and Langley Research Center.

For the SC '99 conference, the VWT team plans another first: running the software on a multiprocessor supercomputer at the NAS Facility rather than on a workstation at the conference. Bryson and Green hope to use Lomax, the new 512-processor Origin2000 system at NAS, as the "compute engine" for their demonstration (*see story, page 2*). "Having larger computer resources at our disposal will enable us to examine larger data sets" and should also make visualization faster, explains Green. If Lomax is not available during the conference, the researchers will run the VWT demonstration on an SGI Onyx2 platform using pre-computed data.

Back at Ames, meanwhile, the data analysis group is cooking up a new application based on the VWT's technology. Nicknamed "Son of VWT," the new software will run on a cen-

tral environment server and various other servers simultaneously, furnishing the ability to visualize several types of datasets in a single virtual space. The original VWT runs on a single server, limiting researchers to one type of data at a time. "The key to this new application is to coordinate all the servers to maintain the same level of interaction among clients," explains Bryson. Although all interactions will take place within a single system, maintaining a close connection between clients using different servers (in order to view different types of data) will be a challenge for the group.

Using the VWT's new collaborative capabilities, more researchers than ever will be able to experience having massive amounts of aerodynamic data at their fingertips—literally. Since the flowfield quantities displayed by the VWT and other visualization techniques are computed rather than measured, an aerospace design can be thoroughly evaluated long before the first physical prototype is constructed. The method has the potential to "dramatically speed up" the aerospace design process, Bryson says. "Looking at the whole process, the VWT is cheaper than [using] the traditional wind tunnel," Bryson says. Traditional wind tunnels do, however, represent the physical world, the fastest and most accurate "simulation environment" we have. Ultimately, says Bryson, "The technologies are complementary."

The Virtual Wind Tunnel puts massive amounts of information at a researcher's fingertips—literally. This means that an aerospace design can be thoroughly evaluated long before the first physical prototype is constructed.

Collaborative Virtual Wind Tunnel...

continued from previous page

Putting evolution to work...

continued from page 7

.....

Holly Amundson is an intern in the NAS publications and media group. She majors in materials science and engineering at California Polytechnic State University in San Luis Obispo.

Radical new architecture...

Continued from page 15

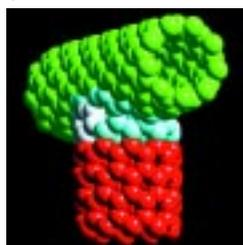
SC '99 exhibit features 37 demos and videos on NASA supercomputing

NASA will appear in full force at the SC99 High Performance Networking and Computing Conference in Portland, Oregon, November 13-19, 1999, with presentations on everything from nanotechnology to Mars exploration and clusters of galaxies.

Ames Research Center

Collaborative Virtual Wind Tunnel

Multiple participants at remote locations can now participate in virtual reality sessions allowing three-dimensional visualization of simulated airflow around aircraft. (See article, page 24.)



Virtual Mechanosynthesis

Computational steering of molecular dynamics simulations creates a virtual environment where researchers can explore diamondoid and fullerene molecules and use a force-feedback device to experience the forces between atoms.

Mars Landing Web Site

A web-based collaborative facility allows Mars researchers to share and peruse proposed landing sites for upcoming Mars Surveyor missions, using 3-D VRML scenes, science evaluations, and an interactive Mars atlas.

The Virtual Collaborative Clinic

Advanced high-fidelity 3-D imaging and high-bandwidth networks "bring the clinic to the patient" by allowing physicians to share complex patient data for remote consultation, diagnosis, treatment planning, and simulation.

Learning Technologies Interactive Media

The NASA Learning Technologies Project offers 60 interactive multimedia packages communicating NASA-derived content to the classroom.

Inside the Information Power Grid

The Grid, a consistent, open, standardized environment for distributed heterogeneous computing, will give scientists and engineers broader access to resources such as supercomputers, data mining systems, scientific instruments, and human collaborators.

Scalable Debugging for the Grid

Using a client-server architecture, the Portable Parallel Distributed Debugger (p2d2) isolates code that is not portable from serial to parallel form and provides graphic views of distributed data arrays.

Legacy Code Modernization Tools

New software tools automate mundane, error-prone parts of code migration from serial to parallel platforms. The tools analyze data affinity, specify data alignment, find optimal data redistributions, and translate from source to source.

Parameter Study Creation and Submission

One obstacle to widespread acceptance of the Globus metacomputing toolkit—the complex job control language needed to set up and execute Globus jobs—has been side-stepped using a new GUI-based tool for the creation and submission of parameter studies on the IPG.



512 Processor Origin2000 System

As the result of an agreement between NASA and Silicon Graphics Inc., a 512-processor Origin2000 computer was assembled at Ames this summer.

The machine has 192 GB of main memory and a 2 TB RAID disk subsystem, and in early tests its cost/performance ratio outstripped that of the CRAY C90 by a factor of 35.

The Charon Portable Parallelization Toolkit

Charon is a library of C and Fortran functions based on the Message Passing Interface that safely help translate legacy code into high-performance, highly scalable parallel code.

Programming Baseline for NPB

The new Programming Baseline for the NAS Parallel Benchmarks facilitates comparisons of parallelizing compilers and tools by providing sample implementations of the NAS Parallel Benchmarks with OpenMP, HPE, and Java.

Time-Varying Volume Rendering

By subdividing volumes into sub-blocks and updating only fast-changing regions, a new hardware-assisted 3-D texture mapping tool speeds volume rendering in time-varying simulations.

Multisource Data Analysis

The VISOR system (Visual Integration of Simulated and Observed Results) is a client-server testbed application for the visualization and analysis of data from multiple sources,

Continued on next page

Continued from previous page
such as wind tunnel measurements and computational fluid dynamics simulations.

High Performance Computing and Communications / National Research and Education Network Video

The NASA Research and Education Network (NREN) is NASA's cornerstone of the Next Generation Internet. Working with federal agencies, academia and industry partners, NREN is aggressively meeting the challenge to provide high performance networks enabling exciting, next generation applications.

No information was available at press time about the following presentation from Ames Research Center:

High Performance Computing and Communications / Computational Aerosciences Program Video

Glenn Research Center

Exploration and Collaboration on the Virtual Reality Immersadesk

Researchers at Glenn Research Center are exploring the benefits of large-scale immersive displays for navigating complex geometries and flow fields or stimulating collaborative research.

National Propulsion Simulation System Video

A video from Glenn Research Center will describe the Numerical Propulsion System Simulation, a high-fidelity, full-engine simulation designed to reduce aircraft development time.

Dynamic Load Balancing Tool

The Dynamic Load Balancing tool evens out the computational workload in parallel computations across all available processors. It determines the optimal load distribution using an algorithm incorporating the computation and communication costs from each process.

Aeroshark Pentium II Cluster

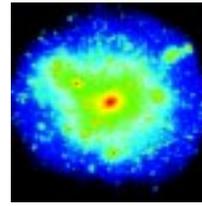
At Glenn Research center, a cluster of 64 Pentium II 400 MHz processors with 16 GB of RAM and a Gigabit ethernet backbone has achieved 50% of the performance of a 24-CPU SGI Origin2000 computer at a fraction of the cost.

No information was available at press time about the following presentations from Glenn Research Center:

Heterogeneous Cluster - NPARC

Heterogeneous Cluster - NPSS 1D

Goddard Space Flight Center



The Digital Earth Workbench

The Digital Earth Workbench uses immersive, interactive virtual reality tools to display multiple layers of information on the geological, biological, climatic, anthropogenic, and other forces affecting the Earth.

Simulating Clusters of Galaxies

Using a the PKDGRAV gravitational simulation tool and a new "volume renormalization" technique, the evolution of clusters of galaxies spanning a significant portion of the Universe was simulated at varying resolutions.

Images of Earth and Space: SC99

This narrated video presents visualizations of observed and simulated data from NASA-sponsored studies of the X-33 aerospace vehicle, Mississippi river flooding, Mars rover simulations, huge basins on Mars, merging neutron stars, and other subjects.

Jet Propulsion Laboratory

Distributed Mission Simulation

This JPL project aims to develop virtual platforms for testing and validating the integrated systems involved in NASA space missions, from spacecraft and payload to instrumentation and data uplinks and downlinks.

Digital Sky Virtual Observatory

A prototype high-performance computing system being developed by the Digital Sky Project will access and inter-relate large, geographically distributed datasets from multiple sky archives.



Martian Rover Simulations

Rover flight software can be tested using simulated Martian terrain and simulated rover hardware, aiding design analysis and mission planning risk analysis.

ParVox: Visualization of Large 4-D Datasets

The ParVox scalable parallel volume rendering system is designed to help researchers visualize very large, time-varying datasets with multiple variables. The program runs on the Cray T3E and the HP Exemplar using MPI 1.4.

Remote Exploration and Experimentation

A video and demonstration will lay out the goals of the REE project: to equip future space science missions with supercomputing technology, support onboard processing of science data, fully utilize space-based instruments, and mitigate communications bottlenecks.



Pyramid

Pyramid is an advanced object-oriented software library supporting parallel adaptive mesh refinement for large-scale scientific and engineering simulations.

MODTOOL

Software integrating optics design, CAD modeling, and structural tools is being used at JPL to reduce the time invested in the design and analysis of microwave- and millimeter-wave sensing instruments.

Conterminous US Landsat Mosaic

JPL supercomputers have constructed a 200-gigabyte mosaic of 430 Landsat images of the conterminous United States, to be used for U.S. Air Force flight simulators

Nanoelectronic Modeling

A simulation tool being developed at JPL allows researchers to explore the physics of nanoscale devices and objects such as resonant tunneling diodes, quantum wells, quantum dots, and molecules.

Langley Research Center



Multidisciplinary Design Optimization Video

The MDO process uses fast, high-fidelity simulations of realistic aerospace vehicle designs to reduce design cycle time. Langley researchers are working to integrate MDO elements across networked, heterogeneous workstations and supercomputers.



Intelligent Synthesis Environment (ISE)

NASA's ISE initiative aims to give scientists and engineers the ability to work together in virtual environments and to simulate the complete life cycle of products and missions before physical construction begins.