CHAPTER 1

Parallel Computing

in which we discover what parallel computing is; we learn how it can help us solve computational problems; and we survey several problems that benefit from parallel computing

CHAPTER 1 Parallel Computing

1.1 Bigger Problems, Faster Answers

Many of today's computer applications require massive amounts of computation. Here's an example of a computational medicine problem involving **magnetic resonance imaging (MRI)**. MRI is a technique for making images of the inside of an organism, such as a living person's brain, without cutting the patient open. The MRI scanner sends a brief, high-intensity radio frequency pulse through the patient. The pulse reverses the orientation of the spins of the atoms in the patient. The MRI scanner then measures the atomic spins in a two-dimensional plane, or slice, through the subject as the atoms "relax," or return to their normal spin orientations. Each measurement takes the form of an $N \times N$-pixel image. The MRI scanner takes a sequence of these image snapshots at closely spaced time intervals (Figure 1.1).

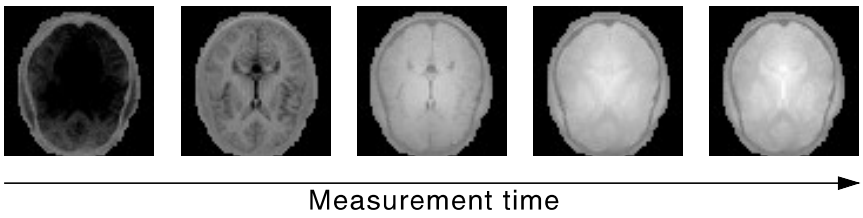

Figure 1.1 Sequence of magnetic resonance images for one slice of a brain

The rates at which the atoms' spins relax helps a doctor diagnose disease. In healthy tissue, the spins relax at certain rates. In diseased tissue, if abnormal chemicals are present, the spins relax at different rates. The sequence of measured spin directions and intensities for a given pixel can be analyzed to determine the spin relaxation rates in the tissue sample corresponding to that pixel. Such a **spin relaxometry analysis** requires sophisticated and time-consuming calculations on each pixel's data sequence to recover the underlying spin relaxation rates from the typically imperfect and noisy images. (In Chapter 36, we will examine MRI spin relaxometry analysis in more detail.)

One computer program that did the spin relaxometry analysis took about 76 seconds to do the calculations for a single pixel. To analyze all the pixels in, say, a 64×64-pixel image, a total of 4,096 pixels, would take about 311,000 seconds—over 3.5 days—of computer time. And this is for just one slice through the subject. A complete MRI scan involves *many* slices to generate a three-dimensional picture of the subject's interior. Clearly, the calculations need to be completed in a drastically shorter time for spin relaxometry analysis to be a useful diagnostic technique.

One alternative for reducing the calculation time is to get a faster computer. The earlier-noted timing measurement was made on a computer that was several years old. Running the same program on an

up-to-date computer that is 5 times faster than the original computer would take about 62,000 seconds to analyze a 64×64-pixel image, or about 17 hours instead of 3.5 days.

It used to be that computer clock speeds doubled roughly every two years. However, that trend finally may be ending. By 2004, CPU chips had achieved clock speeds in the 3 GHz range. If the trend had continued, clock speeds should have reached 12 GHz by 2008—but they did not. Instead, in late 2004, chip makers started moving away from the strategy of boosting chip performance by increasing the raw clock speed, opting instead to introduce architectural features such as "hyperthreaded" and "multi-core" chips (we will say more about these later). Although clock speeds do continue to increase, in the future there is little hope left for dramatically reduced calculation times from faster chips.

Another alternative for reducing the calculation time is to switch to a faster algorithm. Suppose we could devise a different program that was 100 times faster than the original; then running on the faster computer, it would take about 620 seconds—about 10 minutes—to analyze a 64×64-pixel image. Putting it another way, the calculation time per pixel would be 0.15 seconds instead of 76 seconds. However, algorithmic improvements can take us only so far. There comes a point where the fastest-known algorithm on the fastest available computer is still simply not fast enough.

A third alternative is *to have several computers working on the problem simultaneously*. Say we have K computers instead of just one. We divide the problem up into K pieces and assign one piece to each computer. For the MRI spin relaxometry analysis problem, each computer analyzes $(N \times N)/K$ pixels. Then all the computers go to work on their respective pieces at the same time. We say that the computers are executing **in parallel**, and we refer to the whole conglomeration as a **parallel computer**. Henceforth we will refer to the individual units within the parallel computer as **processors** to distinguish them from the parallel computer as a whole.

We could apply a parallel computer to the MRI spin relaxometry analysis problem in either of two ways. Suppose we have a 16-processor parallel computer. We divide the 4,096 pixels among the 16 processors. Because each processor has to do the calculations for only 256 pixels, and because all the processors are running at once, the computation takes only 256 × 0.15 seconds, or about 39 seconds, instead of 10 minutes. The parallel computer let us *reduce the running time* by a factor of 16 while holding the problem size constant. Used in this way, a K-processor parallel computer ideally gives a **speedup** of K (Figure 1.2).

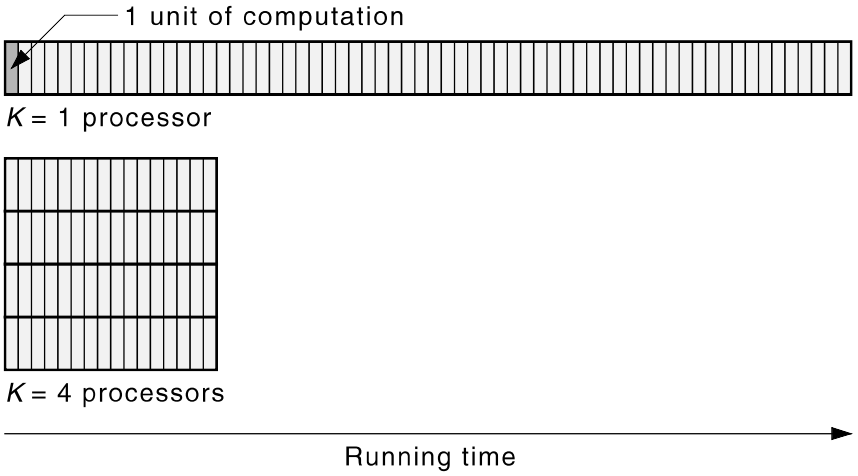

Figure 1.2 Speedup with a parallel computer—same problem size, (1/K) × running time

On the other hand, suppose we use our 16-processor parallel computer to analyze a magnetic resonance image with 16 times as many pixels—a 256×256-pixel image, which is actually the typical size of a magnetic resonance image used for a medical diagnosis. Either the image encompasses a larger area, or the image covers the same area at a finer resolution. Then the computation still takes the same 10 minutes, but we have analyzed a larger image. The parallel computer has let us *increase the problem size* by a factor of 16 while holding the running time constant. Used in this way, a *K*-processor parallel computer ideally gives a size increase, or **sizeup**, of *K* (Figure 1.3).

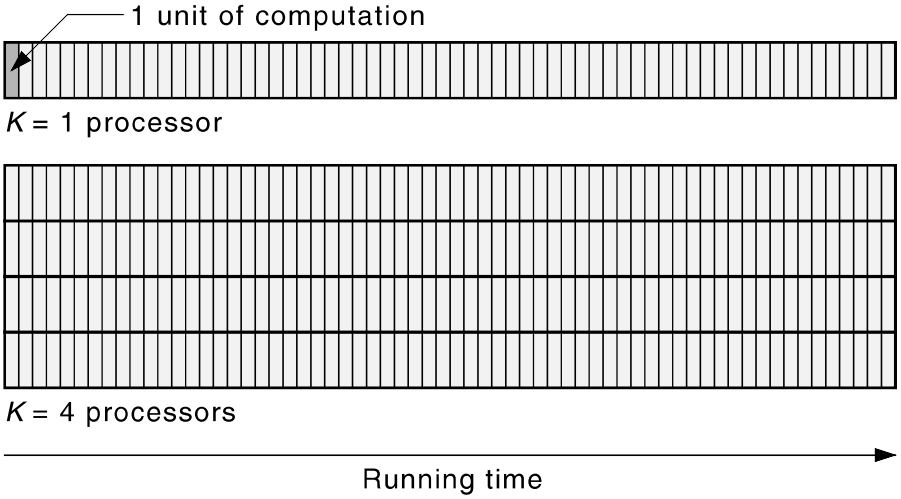

Figure 1.3 Sizeup with a parallel computer—*K* × problem size, same running time

Of course, we can employ both strategies. A 64-processor parallel computer, for example, would let us analyze a 256×256-pixel image in one-fourth the time it takes a single computer to analyze a 64×64-pixel image. The more processors we add, the bigger the images we can analyze, and the faster we can get the answers.

To sum up, **parallel computing** is the discipline of employing multiple processors running all at once to solve the same problem in less time (speedup), to solve a larger problem in the same time (sizeup), or both. Another term often used is **high-performance computing (HPC)**, which emphasizes the improved performance parallel computing provides in solving larger problems or solving problems faster.

1.2 Applications for Parallel Computing

In 2004, the U.S. Office of Science and Technology Policy released a report titled "Federal Plan for High-End Computing." This report lists four broad application areas—climate and weather, nanoscale science and technology, life sciences, and aerospace vehicle design—with problems requiring massive amounts of computation, that can and do benefit from parallel computing. Problems in other areas, such as astrophysics, mathematics, games, and animation, are also attacked using parallel computing. Here are a few examples of such problems:

Weather forecasting. In August 2005, Hurricane Katrina devastated the U.S. Gulf Coast, flooding the city of New Orleans, killing more than 1,800 people, and causing $100 billion in damage. Computer models of the atmosphere, such as the Weather Research and Forecasting (WRF) Model, can predict

a storm's track (the path it will follow, where it will hit land) and intensity (wind speed). The WRF program takes a portion of the earth's atmosphere—a given region of the earth's surface, up to a given height above the surface—and divides this three-dimensional region into many small 3-D cells. The WRF program then uses physical models to calculate atmospheric conditions in each cell, as influenced by the neighboring cells, and advances the forecast in a series of many small time steps. The WRF program uses parallel computing to handle the calculations for large numbers of cells and time steps.

Accurate hurricane track and intensity forecasts can help officials decide how to prepare for a storm and where to evacuate if necessary. However, current hurricane forecasting programs are not all that accurate. With current models, track forecast errors can be as large as 70 kilometers (km), and wind speed forecast errors can be as large as 37 kilometers per hour (kph). A wind-speed shift of 37 kph can change a mere tropical storm to a Category 3 major hurricane. A track error of 70 km could cause officials to evacuate Boca Raton, Florida when they should have evacuated Miami.

To get more accurate predictions, the cells and the time steps in the model must be made smaller; this means that the model must include *more* cells and *more* time steps to cover the same geographic region and the same time span. For example, if the cell's dimensions are decreased by a factor of 2, the number of cells must increase by a factor of 8 to cover the same 3-D region. If the time step size is also decreased by a factor of 2, the number of time steps must increase by a factor of 2 to cover the same time span. Thus, the total amount of computation goes up by a factor of 16. This in turn means that even more powerful parallel computers and parallel programs will be needed to calculate the models.

Climate modeling. On what date will the rainy season begin in Brazil this year, so farmers will know when to plant their crops? Why is rainfall decreasing in the Indian subcontinent—could it be caused by pollution from burning wood for fuel and cooking? What effect will increased levels of atmospheric carbon dioxide have on the world's climate—none at all, or drastic warming that will melt the polar ice caps and inundate coastal cities? Computer-based climate models can answer these questions (and fan the controversies). The Community Climate System Model (CCSM), for example, models the atmosphere, ocean, sea ice, and land surface using a three-dimensional grid of cells like the WRF model. The CCSM program runs on a parallel computer to simulate the earth's climate over the entire globe for time spans of thousands of years. Because there is no end to the number of climatological features and the level of detail that can be included in climate simulation programs, such programs will continue to need the power of parallel computers well into the future.

Protein sequence matching. Imagine you are a biochemist. You have isolated a new protein from the creature you are studying, but you have no idea what the protein does—could it be an anti-cancer agent? Or is it just a digestive enzyme?

One way to get a clue to the protein's function is to match your protein against other proteins; if your protein closely matches proteins of known function, chances are your protein does something similar to the matching proteins. Chemically, a protein is a group of amino acids linked together into a long string. Twenty different amino acids are found in proteins, so a protein can be represented as a string of letters from a 20-character alphabet; this string is called the protein's "sequence." Protein sequence databases collect information about proteins, including their sequences and functions. The Swiss-Prot database, for example, contains well over 385,000 protein sequences ranging in length from 2 to 35,000 amino acids, with a median length of around 300 amino acids. You can determine your new protein's sequence and match it against the protein sequences in the database. Doing so is more complicated than looking up a

credit card number in a financial database, however. Rather than finding a single, exact match, you are looking for multiple, inexact but close matches.

The Basic Local Alignment Search Tool (BLAST) program is at present the premier tool for solving the preceding protein sequence matching problem. The BLAST program combines a "local alignment" algorithm, which matches a piece of one protein sequence against a piece of another protein sequence, with a search of all protein sequences in the database. Because local alignment is a computationally intensive algorithm and because the databases are large, parallel versions of BLAST are used to speed up the searches. In Chapter 37, we will design a parallel program for protein sequence database searching.

Quantum computer simulation. A quantum computer exploits quantum mechanical effects to perform large amounts of computation with small amounts of hardware. A quantum computer register with n qubits (quantum bits) can hold 2^n different states at the same time via "quantum superposition" of the individual qubits' states. Performing one operation on the quantum register updates all 2^n states simultaneously, making some hitherto intractable algorithms practical. For example, in 1994, Peter Shor of AT&T Bell Laboratories published a quantum algorithm that can factor large composite numbers efficiently. If we could do that, we could break the RSA public key cryptosystem, which is the basis for secure electronic commerce on the Internet. The potential for solving problems with polynomial time algorithms on a quantum computer—that would otherwise require exponential time algorithms on a classical computer—has sparked interest in quantum algorithms.

Although small, specialized quantum computers have been built, it will be quite some time before useful general-purpose quantum computers become available. Nonetheless, researchers are forging ahead with quantum algorithm development. Lacking actual quantum computers to test their algorithms, researchers turn to quantum computer simulators running on classical computers. The simulators must do massive amounts of computation to simulate the quantum computer's exponentially large number of states, making quantum computer simulation an attractive area for (classical) parallel computing. Several parallel simulator programs for quantum computers have been published.

Star cluster simulation. Astrophysicists are interested in the evolution of star clusters and even entire galaxies. How does the shape of the star cluster or galaxy change over time as the stars move under the influence of their mutual gravitational attraction? Many galaxies, including our own Milky Way, are believed to have a supermassive black hole (SMBH) at the center. How does the SMBH move as the comparatively much-lighter stars orbit the galactic center? What happens when two galaxies collide? While it's unlikely for individual stars in the galaxies to collide with each other, the galaxies as a whole might merge, or they might pass through each other but with altered shapes, or certain stars might be ejected to voyage alone through intergalactic space.

There are theories that purport to predict what will happen in these scenarios. But because of the long time scales involved, millions or billions of years, there has been no way to test these theories by observing actual star clusters or galaxies. So astrophysicists have turned to observing the evolution of star clusters or galaxies *simulated in the computer*. In recent years, "computational astrophysics" has revolutionized the field and revealed a host of new phenomena for theorists to puzzle over.

The most general and powerful methods for simulating stellar dynamics, the so-called "direct N-body methods," require enormous amounts of computation. The simulation proceeds as a series of time steps. At each time step, the gravitational force on each star from all the other stars is calculated, each star's position and velocity are advanced as determined by the force, and the cycle repeats. A system of N stars requires $O(N^2)$ calculations to determine the forces. To simulate, say, one million stars requires

10^{12} force calculations—*on each and every time step*; and one simulation may run for thousands or millions of time steps. In Chapter 27, we will examine an *N*-body problem in more detail.

To run their simulations, computational astrophysicists turn to special purpose hardware. One example is the GRAPE-6 processor, developed by Junichiro Makino and his colleagues at the University of Tokyo. (GRAPE stands for GRAvity piPE.) The GRAPE-6 processor is a parallel processor that does only gravitational force calculations, but does them much, much faster than even the speediest general-purpose computer. Multiple GRAPE-6 processors are then combined with multiple general-purpose host processors to form a massively parallel gravitational supercomputer. Examples of such supercomputers include the GRAPE-6 system at the University of Tokyo and the "gravitySimulator" system built by David Merritt and his colleagues at the Rochester Institute of Technology.

Mersenne primes. Mersenne numbers—named after French philosopher Marin Mersenne (1588–1648), who wrote about them—are numbers of the form 2^n–1. If a Mersenne number is also a prime number, it is called a Mersenne prime. The first few Mersenne primes are 2^2–1, 2^3–1, 2^5–1, 2^7–1, and 2^{13}–1. (Most Mersenne numbers are not prime.) The largest known Mersenne prime is $2^{43,112,609}$–1, a whopper of a number with nearly 13 million decimal digits, discovered in August 2008 by the Great Internet Mersenne Prime Search (GIMPS) project.

Starting in 1996, the GIMPS project has been testing Mersenne numbers to find ever-larger Mersenne primes. The GIMPS project uses a "virtual parallel computer" to test candidate Mersenne numbers for primality in parallel. The GIMPS parallel computer consists of PCs and workstations contributed by volunteers around the globe. Each volunteer downloads and installs the GIMPS client program on his or her computer. The client runs as a lowest-priority process, and thus uses CPU cycles only when the computer is otherwise idle. The client contacts the GIMPS server over the Internet, obtains a candidate Mersenne number to work on, subjects the candidate to an array of tests to determine whether the candidate is prime, and reports the result back to the server. Because the candidate Mersenne numbers are so large, the primality tests can take days to weeks of computation on a typical PC. Since commencing operation, the GIMPS project has found 12 previously unknown Mersenne primes with exponents ranging from 1,398,269 to the aforementioned 43,112,609.

While Mersenne primes have little usefulness beyond pure mathematics, there is a practical incentive for continuing the search. The Electronic Frontier Foundation (EFF) has announced a $100,000 prize for the first discovery of a ten-million-digit prime number—a prize now claimed by the GIMPS project. The EFF has also announced further prizes of $150,000 for the first 100-million-digit prime number and $250,000 for the first one-billion-digit prime number. While any prime number (not necessarily a Mersenne prime) qualifies for the prizes, the GIMPS project has perhaps the best chance at reaching these goals as well. According to their Web site, with these prizes, "EFF hopes to spur the technology of cooperative networking and encourage Internet users worldwide to join together in solving scientific problems involving massive computation."

Search for extraterrestrial intelligence (SETI). Since 1997, researchers at the University of California, Berkeley have been using the radio telescope at Arecibo, Puerto Rico to search for signs of extraterrestrial intelligence. As the telescope scans the sky, the researchers record radio signals centered around a frequency of 1.42 GHz. (Because hydrogen atoms throughout the universe emit energy at this frequency, a fact of which any advanced civilization ought to be aware, SETI researchers feel that extraterrestrials who want to announce their presence would broadcast signals near this frequency.) These recorded signals are then analyzed to determine if they contain any "narrowband" signals—signals

confined to a small frequency range. Whereas most of the energy in the signal is "broadband" background noise spread out over a wide frequency range, a narrowband signal—like an AM radio, FM radio, television, or satellite signal—is more likely to have been generated by an intelligent being. Any detected narrowband signals are subjected to further scrutiny to eliminate signals of terrestrial origin. Signals that cannot be identified as terrestrial might just be extraterrestrial.

Because of the enormous amounts of radio signal data collected and the extensive computations needed to analyze the data to detect narrowband signals, the Berkeley researchers realized they needed a massively parallel computer. Rather than buy their own parallel supercomputer, they used the same approach as the GIMPS project and created the SETI@home project in 1999. Volunteers install the SETI@home client program on their computers. Running as a screen saver or as a low-priority background process, each client program contacts the SETI@home server over the Internet, downloads a "workunit" of radio signal data, analyzes the workunit, and sends the results back to the server. The SETI@home virtual supercomputer has analyzed nearly 300 million workunits so far, each workunit occupying 350 kilobytes of data and requiring 10 to 12 hours of computation on a typical PC, and has detected over 1.1 billion narrowband signals.

The SETI@home approach to parallel computing was so successful that the Berkeley researchers developed the Berkeley Open Infrastructure for Network Computing (BOINC), a general framework for Internet-based client-server parallel computation. Volunteers can donate compute cycles to any project that uses the BOINC infrastructure. Some 50 projects now use BOINC; they range from SETI@home itself to protein structure prediction to climate modeling.

Has SETI@home found any signs of extraterrestrial intelligence? The researchers are still working their way through the 1.1 billion narrowband signals that have been detected. So far, none can be conclusively stated as being of extraterrestrial origin.

Chess. On May 11, 1997, Garry Kasparov, chess grandmaster and then world chess champion, sat down opposite IBM chess computer Deep Blue for a six-game exhibition match. Kasparov and Deep Blue had met 15 months earlier, on February 10, 1996, and at that time Kasparov prevailed with three wins, two draws, and one loss. After extensive upgrades, Deep Blue was ready for a rematch. This time, the results were two wins for Deep Blue, one win for Kasparov, and three draws, for an overall score of Deep Blue 3.5, Kasparov 2.5. It was the first time a computer had won a match against a reigning human world chess champion. After his defeat, Kasparov demanded a rematch, but IBM refused and retired the machine. Deep Blue's two equipment racks are now gathering dust at the National Museum of American History and the Computer History Museum.

Deep Blue was a massively parallel, special-purpose supercomputer, consisting of 30 IBM RS/6000 SP server nodes augmented with 480 specialized VLSI chess chips. It chose chess moves by brute force, analyzing plays at a rate of 200 million chess positions per second. It also had an extensive database of opening positions and endgames.

While Deep Blue is gone, computer chess research continues. The focus has shifted away from specialized parallel hardware to software programs running on commodity parallel machines. In November 2006, the Deep Fritz chess program, a parallel program running on a PC with two Intel dual-core CPU chips, beat then world chess champion Vladimir Kramnik with two wins and four draws. Many people now believe that in the realm of chess, human dominance over computers is at its end. Of course, it's not really man versus machine, it's human chess players against human computer builders and programmers.

Animated films. In 1937, Walt Disney made motion picture history with the animated feature film *Snow White and the Seven Dwarfs*. While Disney had been making animated short films since the 1920s, *Snow White* was the world's first *feature-length* animated film. In those days each frame of the film was laboriously drawn and colored by hand on celluloid. All that would change in 1995, when Pixar Animation Studios and Walt Disney Pictures released *Toy Story*, the world's first feature-length *computer*-animated film. Six years later, in 2001, DreamWorks Animation SKG debuted the computer-animated film *Shrek*, the first animated film to win an Academy Award. Since then, scarcely a year has gone by without several new feature-length computer-animated film releases.

During the early stages of production on a computer-animated film, the artists and designers work mostly with individual high-end graphics workstations. But when the time comes to "render" each frame of the final film, adding realistic surface textures, skin tones, hair, fur, lighting, and so on, the computation shifts to the "render farm"—a parallel computer with typically several thousand nodes, each node a multicore server. For *Toy Story*, the render farm had to compute a 77-minute film, with 24 frames per second and 1,536×922 pixels per frame—more than 157 billion pixels altogether. Despite the render farm's enormous computational power, it still takes hours or even days to render a single frame. As movie audiences come to expect ever-more-realistic computer-animated films, the render farms will continue to require increasingly larger parallel computers running increasingly sophisticated parallel rendering programs.

We've only scratched the surface, but perhaps you've gotten a sense of the broad range of problems that are being solved using parallel computing. The largest and most challenging of today's computer applications rely on parallel computing. It's an exciting area in which to work!

1.3 For Further Information

On the end of the trend towards ever-increasing CPU clock speeds:

- Herb Sutter. A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3):16–22, March 2005.
- Craig Szydlowski. Multithreaded technology and multicore processors. *Dr. Dobb's Journal*, 30(5):58–60, May 2005.

On applications for high performance computing:

- U.S. Office of Science and Technology Policy. Federal plan for high-end computing. May 10, 2004. http://www.nitrd.gov/pubs/2004_hecrtf/20040702_hecrtf.pdf

On hurricane forecasting and the Weather Research and Forecast Model:

- Thomas Hayden. Super storms: No end in sight. *National Geographic*, 210(2):66–77, August 2006.

- J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang. The Weather Research and Forecast Model: software architecture and performance. In *Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, 2004. http://www.wrf-model.org/wrfadmin/docs/ecmwf_2004.pdf
- The Weather Research and Forecasting Model. http://www.wrf-model.org/index.php

On using parallel computing for weather modeling. "The Weather Man," a fascinating science fiction story written in 1962 when computers were still young, computer networks not yet invented, and parallel computers barely beginning, nonetheless managed to give a remarkably prescient depiction of parallel computing on what are now known as networked workstation clusters:

- Theodore L. Thomas. "The Weather Man." *Analog*, June 1962.

"The Weather Man" is reprinted in a couple more recent science fiction collections:

- Isaac Asimov and Martin H. Greenberg, editors. *Isaac Asimov Presents the Great SF Stories #24 (1962)*. DAW Books, 1992.
- David G. Hartwell and Kathryn Cramer, editors. *The Ascent of Wonder: The Evolution of Hard SF*. Tor Books, 1994.

On the Community Climate System Model:

- Special issue on the Community Climate System Model. *Journal of Climate*, 19(11), June 1, 2006.
- W. Collins, C. Bitz, M. Blackmon, G. Bonan, C. Bretherton, J. Carton, P. Chang, S. Doney, J. Hack, T. Henderson, J. Kiehl, W. Large, D. McKenna, B. Santer, and R. Smith. The Community Climate System Model Version 3 (CCSM3). *Journal of Climate*, 19(11):2122–2143, June 1, 2006.
- Community Climate System Model. http://www.ccsm.ucar.edu/

On protein sequence matching:

- The Universal Protein Resource (UniProt), including the Swiss-Prot protein sequence database. http://www.uniprot.org/
- National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov/
- S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 5, 1990.