SCIENTIFIC APPLICATIONS OF GRID COMPUTING

The use of parallel computation for scientific research is so widespread today that it's easy to forget it began on a large scale as recently as the late 1980s. With the increasing attention given to parallel algorithms for scientific research, researchers soon realized that certain common applications, such as Monte Carlo algorithms, were embarrassingly parallel: they required very little interprocessor communication and could be effectively deployed on networks of modest communications bandwidth, including the Internet.

This quickly led to the idea of using large geographically distributed networks of computers for scientific research, a concept that first captured the public's imagination with the SETI@home project. When researchers on the Search for Extraterres-

1521-9615/05/$20.00 © 2005 IEEE
Copublished by the IEEE CS and the AIP

BRUCE M. BOGHOSIAN
Tufts University
PETER V. COVENEY
University College London

trial Intelligence (SETI) project needed to analyze enormous amounts of data from the Arecibo radio telescope, the Space Sciences Laboratory at the University of California, Berkeley, decided to make a free screen saver available to the public. When activated, this program downloads time sequences of radio telescope data and searches them for signatures characteristic of localized radio sources or narrowband digital transmission. Today, SETI@home has more than 5 million participants, and has inspired other scientific applications in need of vast computing resources:

- Founded in 1997, distributed.net (http://distributed.net) applied similar methods to crack RSA Lab's RC5 64-bit encryption algorithm; it completed the task on 14 July 2002 after using more than 300,000 distributed computing resources over 1,757 days.
- The Stanford University-based folding@home project (http://folding.stanford.edu) applies distributed computing to large-scale molecular simulations aimed at the discovery of new drugs.
- CERN's LHC@home project (http://LHCathome.cern.ch) uses geographically distributed computing to follow particle trajectories, which is helping researchers design the Large Hadron Collider.
- Oxford University's ClimatePrediction.net is using distributed computing to produce a forecast of the world's climate in the 21st century.

All these SETI spin-offs have produced results of scientific interest, and the list of similar projects is rapidly growing.

A New Utility?

Of course, some parallel scientific applications don't easily lend themselves to geographical distribution. Spatial grid applications, for example, can be domain-decomposed, but the communications requirements between neighboring sublattices are severe. Most such work is done on single multiprocessors with fast interconnection hardware. This isn't to say that geographically distributed spatial domain decomposition is never a good idea: if your application requires a spatial grid so large that it won't fit onto any existing multiprocessor, you have little choice but to geographically distribute the calculation. It's also possible to use geographically distributed computers to "task farm" calculations on large spatial lattices with different input parameters, or to preprocess or postprocess data from such calculations.

As instances of geographically distributed applications continue to proliferate, some argue that computational power will become a utility, like electrical power; this important (if somewhat hackneyed) metaphor is often used to justify geographically distributed, or *grid*, computing.[1,2] Let's explore this metaphor further.

Electrical power's ubiquity in our daily lives occurred in three distinct stages: In the first stage (the mid 1800s), individual experimenters gained general access to the understanding and construction of electrical batteries and generators. In the second (the early 1900s), large enterprises such as individual factories developed the ability to generate enough electricity for serious commercial use. The third stage (the 1930s) saw the first vast electrical power distribution grids and the federal Rural Electrification program in the US, which enabled the geographic, economic, and cultural separation of power providers from power consumers. Only then did the discovery of electricity impact most peoples' lives.

Interestingly, computation reversed the first two steps of this chronology. Large enterprises dedicated to both research and manufacturing have used centralized computers since the 1950s, but electronic processors with integrated circuitry became inexpensive and widely available to individual experimenters only with the advent of the 8080 microprocessor in the mid 1970s. Between these two milestones lies the creation of the Internet, which began as the Arpanet in 1969. The computational metaphor's chronology is entirely wrong: it would be as if Thomas Edison had predated Michael Faraday, and the construction of power lines began somewhere in between. Nevertheless, the widespread distribution of resources on a grid is likely to be the last stage in both cases.

Grid Computing

The concept of grid computing has advanced in recent years, but one of its more immediate problems is its various definitions. For the purposes of this article, we adopt the following general definition:

> "Grid computing is distributed computing performed transparently across multiple administrative domains."[3]

Here, the word "computing" is used in the most general sense possible: it refers to any kind of digital activity, from database searches and queries to simulation, visualization, computer-aided manufacturing, access to computer-controlled laboratory experiments, and so on. Think of all these different types of activity as possible nodes on a geographically distributed "grid."

Also critical to the definition is the word "trans-

parently." After all, the Internet has made it possible to access computational resources remotely, but individual users must still extract those resources from specific accounts on specific supercomputers. Just as we don't care which electrical power plant is responsible for powering the lights over our heads, so we would prefer not to care which remote computer is solving our scientific problems. Advances in AC power and transmission line theory long ago solved the physical problems associated with distributing electricity among multiple and widely separated power sources and sinks, but distributing computational resources among multiple and widely separated sources and users is an arguably more difficult *algorithmic* problem.

Grid computing is distinguished from parallel computing on one or more multiprocessors insofar as it occurs across different administrative domains. Without standardization, grid computing practitioners would have to acquire accounts at many different computer centers, managed by different people and organizations, and employing different security and authentication protocols and accounting practices. The variety of software suites often installed at these centers makes debugging, verification, and validation of results an important logistical issue. Open grid specifications and standards, such as those championed by the Global Grid Forum (GGF; www.gridforum.org), help practitioners bridge these separate resources, but standardization problems exist at all levels of software. LHC@home researchers, for example, have discovered that different rounding characteristics of the tangent and exponential functions on different computer hardware results in large variability in results involving chaotic particle orbits; this underscores the need for standardization of mathematical software libraries in grid computing applications.

Middleware and Workflows

Although the Internet has existed in various forms since the late 1960s, the advent of the World Wide Web was what made communication via the Internet simple, transparent, and usable for a range of different purposes. What the Web did for communication, grids endeavor to do for computation.

To that end, developers have devoted a great deal of effort to designing middleware that eases grid users' experience and provides them with the right level of abstraction. This middleware aims to build on and extend the Web's information and database management capabilities, thus allowing remote deployment of computational resources.

The most widely adopted grid middleware effort currently in existence is the open-source Globus Toolkit (www.globus.org). The Globus Alliance, which is developing the toolkit, includes both academic and industrial partners from around the world, and it sponsors workshops, tutorials, publications, and books on Globus and its applications.

Most grids today use version 2 of the Globus Toolkit (GT2), although others use earlier versions all the way back to GT1.2; GT4, expected in the near future, will also enable database access and integration. (For reasons too tedious to discuss here, GT3 was obsolesced before release.) Because the Globus Toolkit is a substantial software package with a concomitantly long learning curve, and because it doesn't yet provide certain services required by scientific programmers, leaner packages are available for both accessing and augmenting the elements of its functionality that are most useful for scientific computing.

MPICH-G2, for example, lets users access grid resources with the same programming model and user-level API offered by the message-passing interface (MPI) library, and its developers will soon enhance it with the ability to spawn remote processes on grids running GT2. In this special issue of *Computing in Science & Engineering*, the article by Steve Dong, Nick Karonis, and George Karniadakis describes the use of MPICH-G2 for geographically distributed spatial domain decomposition in computational fluid dynamics.

Although MPICH-G2 works directly on top of GT2 grids, other approaches provide capabilities beyond the GT2 level. WSRF::Lite (www.sve.man.ac.uk/Research/AtoZ/ILCT/), for example, is a Perl implementation of the Web Services Resource Framework (WSRF), whose continuing development is supported by the UK's Open Middleware Infrastructure Institute (OMII; www.omii.ac.uk). The Perl implementation's advantage is that it affords an easily adjustable middleware environment that can conform to any proposed grid specification or standard relatively quickly. In this issue, Peter Coveney, Jonathan Chin, Matt Harvey, and Shantenu Jha discuss the use of such lightweight middleware for computationally steered soft-condensed-matter physics and molecular dynamics. As their article illustrates, workflow management is one of the more challenging problems involved in grid-enabling scientific applications. Grid-based applications must be able to access remote databases, conduct simulations, and launch new remote processes conditionally.

Workflow management is particularly demanding for the problem of weather prediction, in which databases constantly change with the arrival of new measurements. In their article, "Cooperating Services for Managing Data-Driven Compu-

tational Experimentation," Beth Plale, Dennis Gannon, Yi Huang, Gopi Kandaswarny, Sangmi Lee Pallickara, and Aleksander Slominski describe a project that lets users straightforwardly specify a workflow, manage associated metadata, and receive notifications about the progress of their computations in real time.

Grid enablement is especially exciting for very large suites of codes, dedicated to a broad area of scientific application. For such projects, it's worthwhile to think carefully about how to bundle the various component software, the precise nature of interaction between the components, and the standards for component development. Dali Wang, Eric Carr, Louis Gross, and Michael Berry present such an analysis for the very complicated application of ecosystem modeling and natural resource management.

The TeraGrid and Grid Applications

In 2004, the US National Science Foundation-funded TeraGrid went into production, with more than 20 Tflops of performance distributed among nine sites across the US.[4] Similar efforts are under way internationally, with the European Union's EGEE (Enabling Grids for E-sciencE, http://public.eu-egee.org)[5] and DEISA (Distributed European Infrastructure for Scientific Applications; www.deisa.org) projects, and the UK's National Grid Service (NGS; www.ngs.ac.uk) initiative. If present levels of grid funding are sustained, we could expect that all the largest endeavors in both scientific and business computing will employ a geographically distributed grid infrastructure in five years' time.

To date, computer scientists and grid technologists have driven most of the TeraGrid effort, but that's slowly beginning to change. The number of natural scientists exploiting the TeraGrid is small but expected to grow rapidly as the approach's power and potential become more evident. Since its first call for proposals, for example, the NGS has funded a mixture of computer and natural scientists focused on application development.

In any case, it's clear that future development of computational grids will depend on the active participation of scientific application developers; unfortunately, most are reluctant to embrace the new computing paradigm because much of the middleware is still difficult to use and has a steep learning curve. Maximally effective use of computational grids will require close interdisciplinary collaboration and cooperation among biologists, chemists, physicists, engineers, mathematicians, and computer scientists. The associated communication barriers are significant, but they aren't insurmountable.

Although grid computing is now a vast enterprise, this issue of *CiSE* focuses only on its successful large-scale scientific applications and the tools aimed at facilitating such applications. All the projects described in this issue have been successfully deployed on existing computational grids or are expected to be deployed soon, and all are motivated by the ultimate goal of helping scientific researchers conduct computational science research that wouldn't otherwise be possible. We hope these articles convey something of the pioneering spirit of grid computing's early adopters for scientific applications and of the excitement of harnessing computational resources on an unprecedented scale.

References

1. I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 1999.
2. F. Berman, G. Fox, and A.J. Hey, *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, 2003.
3. P.V. Coveney, "Scientific Grid Computing," *Philosophical Trans. Royal Soc. London A*, vol. 363, 15 Aug. 2005; www.doi.org (10.1098/rsta.2005.1632).
4. P.H. Beckman, "Building the TeraGrid," *Philosophical Trans. Royal Soc. London A*, vol. 363, 15 Aug. 2005; www.doi.org (10.1098/rsta.2005.1602).
5. F. Gagliardi et al., "Building an Infrastructure for Scientific Grid Computing: Status and Goals of the EGEE Project," *Philosophical Trans. Royal Soc. London A*, vol. 363, 15 Aug. 2005; www.doi.org (10.1098/rsta.2005.1603).

Bruce Boghosian is a professor of mathematics and adjunct professor of computer science at Tufts University. His research interests include theoretical and computational fluid dynamics and grid computing. Boghosian has a BS in physics and an MS in engineering from the Massachusetts Institute of Technology, and a PhD in applied science and engineering from the University of California, Davis. He is a fellow of the American Physical Society, and a member of Sigma Xi and the American Mathematical Society. Contact him at bruce.boghosian@tufts.edu.

Peter Coveney is a professor in physical chemistry and director of the Centre for Computational Science at University College London, where he is also an honorary professor of computer science. His research interests include theoretical and computational science, statistical mechanics, high-performance computing, and visualization. Coveney has a BA, an MA, and a DPhil in chemistry from the University of Oxford. He is a fellow of the Royal Society of Chemistry and of the Institute of Physics. He is also the chair of the UK Collaborative Computational Projects Steering Panel (www.ccp.ac.uk). Contact him at p.v.coveney@ucl.ac.uk.