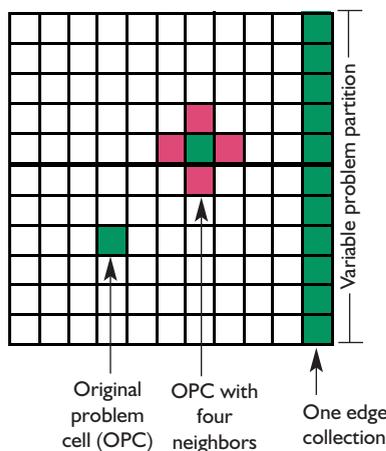


# Grid Computing Made Simple

There has been a surge of interest in grid computing, a way to enlist large numbers of machines to work on multipart computational problems such as circuit analysis or mechanical design. There are excellent reasons for this attention among scientists, engineers, and business executives. Grid computing enables the use and pooling of computer and data resources to solve complex mathematical problems. The technique is the latest development in an evolution that earlier brought forth such advances as distributed computing, the Worldwide Web, and collaborative computing.

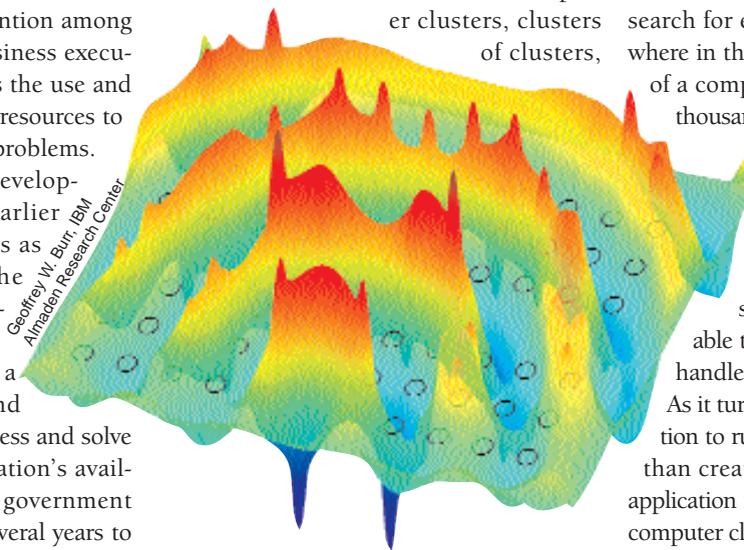
Grid computing harnesses a diverse array of machines and other resources to rapidly process and solve problems beyond an organization's available capacity. Academic and government researchers have used it for several years to solve large-scale problems, and the private sector is increasingly adopting the technology to create innovative products and services, reduce time to market, and enhance business processes.

The term grid, however, may mean differ-



**Figure 1.** A set of methods describes the connectivity of the original problem cell (OPC) with its neighbors and specifies the calculations to be performed by the cell using local data. Groups of OPCs form collections, one or more of which define the variable problem partition assigned to a computer node.

ent things to different people. To some users, a grid is any network of machines, including personal or desktop computers within an organization. To others, grids are networks that include computer clusters, clusters of clusters,



**Figure 2.** OptimalGrid has been used to model the propagation of infrared light through a photonic-bandgap structure of silicon pillars (represented by circles), a calculation that requires interactions between electrical and magnetic fields of nearest-neighbor grid cells and which grows rapidly in memory demands and run-time. The peaks show the value of the magnetic field, which is related to the intensity of light.

or special data sources. Both of these definitions reflect a desire to take advantage of vastly powerful but inexpensive networked resources. In our work, we focus on the use of grids to perform computations as opposed to accessing data, another important area known as data grid research.

## Different systems

Grid computing is akin to established technologies such as computer clusters and peer-to-peer computing in some ways and unlike them in others. Peer-to-peer computing, for example, allows the sharing of files, as do grids, but grids enable users to share other resources as well. Computer clusters and distributed computing require a close proximity and operating homogeneity; grids allow com-

putation over wide geographic areas using computers that are heterogeneous.

Current grid uses such as SETI@home—which taps personal computers on an as-available basis to analyze data obtained in a search for evidence of intelligent life elsewhere in the universe—allow the spreading of a complex calculation over hundreds, thousands, or even millions of machines using a local area network (LAN) or the Internet (Table 2). Although the computational problems solved today by grid computing are often highly sophisticated, the software available to manage these problems cannot handle connected parallel applications. As it turns out, creating a parallel application to run on a grid is even more difficult than creating a large monolithic custom application for a dedicated supercomputer or computer cluster.

Grids are usually heterogeneous networks. Grid nodes, generally individual computers, consist of different hardware and use a variety of operating systems, and the networks connecting them vary in bandwidth. Realizing the vision of ubiquitous parallel computing on a grid will require that we make grids easy to use, and this need applies both to the creation of new applications and to the distribution and management of applications on the grid itself. To accomplish this goal, we need to establish standards and protocols such as open grid services architecture—which allows communication across a network of heterogeneous machines—and tool kits such as Globus, which implement the rules of the grid architecture (Table 1).

We will also require specialized middleware (the software glue that connects an application to the “plumbing” needed to make it run) that effectively hides the complexity of creating and deploying parallel grid applications. Such user-friendly middleware for connected parallel processing does not yet exist, but its development should automate the process and make it possible for people to run connected parallel problems without detailed knowledge of the grid infrastructure.

Grid computing is becoming a critical component of science, business, and industry. Making grids easy to use could lead to advances in fields ranging from industrial design to systems biology to financial management. Grids could allow the analysis of huge investment portfolios in minutes instead of hours, significantly accelerate drug development, and reduce design times and defects. With computing cycles plentiful and inexpensive, practical grid computing would open the door to new models for compute utilities, a service similar to an electric utility in which a user buys computing time on-demand from a provider.

Some industrial applications are important enough to warrant the use of dedicated high-end computers (supercomputers or clusters of computers and/or supercomputers). A much larger body of scientific and engineering applications stands to benefit from grid computing, including weather forecasting, financial and mechanical modeling, immunology, circuit simulation, aircraft design, fluid mechanics, and almost any problem that is mathematically equivalent to a flow.

## Complexity

The simplest class of applications addressed with a computational grid has been *independently parallel* problems (sometimes called embarrassingly parallel because they are relatively straightforward to solve with a grid). These applications work in a simple scatter-gather model; that is, a problem is divided into pieces of data, and a separate data set is sent via the Internet to different nodes, each of which works independently and without communicating with the other nodes to derive its results. SETI@home and Folding@home (which uses thousands of computers in an effort to understand how proteins fold precisely into the structures that enable them to carry out their biological functions) are good examples of independently parallel applications.

Such problems are well suited for the distributed computing power of a grid, and they are straightforward to create. However, they do not require or use autonomic fea-

tures—which are automatic and provide feedback—to actively manage and maximize the effective use of available grid resources. Issues such as failed nodes or missing data sets are dealt with by re-running the calculation. This simple architecture is possible because failure at one node does not affect the calculations made by other nodes.

**TABLE I. GRID STANDARDS AND TOOLKITS**

The Globus Project	<a href="http://www.globus.org/">http://www.globus.org/</a>
The Grid Forum	<a href="http://www.gridforum.org/">http://www.gridforum.org/</a>
Open Grid Services Architecture	<a href="http://www.globus.org/ogsa/">http://www.globus.org/ogsa/</a>
The Condor Project	<a href="http://www.cs.wisc.edu/condor/">http://www.cs.wisc.edu/condor/</a>
<b>EMERGING GRID APPLICATIONS</b>	
SETI Project	<a href="http://setiathome.berkeley.edu">http://setiathome.berkeley.edu</a>
Protein Folding	<a href="http://folding.stanford.edu">http://folding.stanford.edu</a>

A larger and more general class of applications can be described as *connected parallel* problems, which require more sophisticated management in almost every area, including problem definition, problem partitioning, code deployment, grid-node management, and system coordination. These applications include finite element model (FEM) techniques, commonly encountered in industry and the commercial world because they often are used to study problems related to physical objects or process flow, and cellular automata problems, which include areas such as fractals and pattern formation.

FEM problems are solved using a set of well-understood techniques, which have been applied in areas such as physics, financial systems, life sciences, and complex simulations. In a typical FEM application, such as determining the stress on an airplane wing, the object is divided into finite elements and the appropriate equations are solved for each element. However, the solution to the problem depends not just on the answer in each element but on data from all adjacent regions. Thus, rapidly solving such a problem requires that each cell be connected to the other cells and that the cells communicate with one another. In cases such as this, it has been difficult to create large-scale, connected

parallel applications because of the special expertise and access to expensive resources needed to do so.

## OptimalGrid

Most people working on grid computing today focus on the challenges of its physical operations, such as how to determine what computer and database resources are available and how to organize them into a functioning system. Our group, instead, is attempting to create a means of easy, seamless grid access and operation for anyone who needs to solve a connected parallel problem, no matter what grid the person uses—be it an in-house supercomputer or a group of 10,000 personal computers situated around the world.

To demonstrate how one might simplify the creation of applications on a grid, we have developed a prototype called OptimalGrid, which handles both independently parallel and connected parallel problems (Figure 2). OptimalGrid is available for download for evaluation at [www.alpha.works.ibm.com/tech/optimalgrid](http://www.alpha.works.ibm.com/tech/optimalgrid). This self-contained middleware uses a much different approach than existing grid tool kits and serves as a model for the next generation of grid operations. It provides a coordinating interface between the software that manages the grid nodes and the application software, and it incorporates a new programming model that provides autonomic functions to hide the complexity of creating and running parallel applications. OptimalGrid requires only that the networked computers all have a Java run-time installed.

Not even an expert administrator could orchestrate the complex connected problems of a heterogeneous distributed-computer system. To this end, the OptimalGrid system incorporates instrumentation, feedback, and a certain amount of knowledge, or rules, to maintain a balanced performance on the grid and react to various kinds of failures. Its users do not have to struggle with challenges such as partitioning the problem, finding available grid nodes, delivering pieces of code to them, or reappportioning the pieces of the problem among the nodes to balance the workload.

TABLE 2. SEARCHING FOR EXTRATERRESTRIALS ON THE GRID

	TOTAL	LAST 24 h
Users	4,570,474	1,226
Results received	944,671,502	1,168,568
Total CPU time	1,525,673.220 y	1,226.757 y
Floating point operations	$3.268404 \times 10^{25}$	$4.557415 \times 10^{18}$ (52.75 teraflops/s)
Average CPU time per work unit	14 h, 08 min, 51.5 s	9 h, 11 min, 46.3 s

**SETI@home is a scientific experiment that uses Internet-connected computers in the search for extraterrestrial intelligence. You can participate by running a free program that downloads and analyzes radio telescope data. A table on the Web site summarizes results.**

Users simply have to supply the code that represents their basic problem algorithm, and OptimalGrid manages everything else.

Each node on the grid receives a piece of the problem, which consists of a collection of original problem cells (OPCs) (Figure 1). An OPC is the smallest piece into which the problem is divided, and each one needs to communicate and share data with its neighbors. OptimalGrid automates this communication and attempts to minimize the amount of network communication needed to solve a problem. When the program for the application is loaded, the middleware automatically partitions the problem using the following procedures:

1. Determine the complexity.
2. Identify the number of nodes available.
3. Use algorithms to predict the optimal number of grid nodes needed to solve the problem.
4. Optionally interact with the user to divide the problem into an optimal number of pieces. Whether the user or OptimalGrid partitions the problem, the middleware predicts the computation time for the problem.
5. Partition the application data into OPCs.
6. Allow the user the option to customize the data. In assessing stress on an airplane wing, for example, the user might decide to remove one or two rivets from a particular place.
7. Launch the program.

## Autonomic features

Autonomic computing is a prerequisite to creating grids that solve connected parallel problems, because as the number of applications and the volume of data on a grid increase, the need to coordinate and set priorities grows exponentially. Thus, systems that self-manage a grid and diagnose and resolve problems are vital to its successful operation. OptimalGrid attempts to implement three autonomic features: self-configuration, self-optimization, and self-healing.

When the OptimalGrid system initializes itself to solve a problem, it automatically retrieves from the grid a list of available computer nodes. It also obtains the grid's performance characteristics. At run-time, Optimal-

Grid measures ongoing performance, including communication time, computation time, and the complexity of the problem pieces. OptimalGrid uses this information to configure the grid by calculating the optimal number of computer nodes, partitioning the problem, and distributing its pieces in a way that obtains the best possible performance on whatever grid is used.

The middleware monitors the run-time performance of each node with respect to the particular piece of the problem that it is handling, which helps it continue to optimize the computation and the network. These measurements enable the system's autonomic program manager, which serves as the application coordinator, to reassign problem pieces among grid nodes and make other needed adjustments.

OptimalGrid allows the system to self-heal if one or more computer nodes fail during a computational sequence. The loss of a grid node during a sequence does not mean the complete loss of the calculation performed thus far by the node, but only some of the results obtained by the failed node. When OptimalGrid detects the failure of a computer node, it stops calculations across the grid until the failed node recalculates the results lost during the sequence. Although the grid must remain idle during this catch-up phase, a short delay is preferable to having to restart the problem solution from the beginning. Once the node finishes its recalculation, the grid continues working on the overall problem.

The OptimalGrid system is designed to bring the immense potential of grid computing within easy reach of users who are not grid-infrastructure experts. By

including autonomic features such as self-configuration, self-optimization, and self-healing, OptimalGrid seeks to deliver a robust system capable of handling truly connected problems to meet a broad class of user needs for a broad

range of industrial and scientific applications. OptimalGrid is a new programming model designed for the grid environment. It is optimal in the sense that the system attempts to optimize and balance the pieces of the workload to make the best use of any existing grid infrastructure. Initial results look promising.

## Further reading

Gelemter, D.; Bernstein, A. J. Distributed Communication via Global Buffer. In *Proc. of the ACM Principles of Distributed Computing Conference*; Association for Computing Machinery: New York, 1982; pp. 10–18.

Gelemter, D. Generative Communication in Linda. *TOPLAS* 1985, 7 (1), 80–112.

OptimalGrid; [www.alphaworks.ibm.com/tech/optimalgrid](http://www.alphaworks.ibm.com/tech/optimalgrid).

Shread, P. Even Small Companies Can Benefit from Grid Computing; available at [http://www.gridcomputingplanet.com/features/article/0,,3291\\_946331,00.html](http://www.gridcomputingplanet.com/features/article/0,,3291_946331,00.html). 

## B I O G R A P H Y

James H. Kaufman (kaufman@almaden.ibm.com), Glenn Deen@alma.den.ibm.com, Toby J. Lehman (toby@almaden.ibm.com), and John Thomas (jthomas@cruzio.com) are researchers on the OptimalGrid Project at the IBM Almaden Research Center in San Jose, California. Kaufman is a member and former chair of the American Physical Society's Forum on Industrial and Applied Physics (FIAP). For more information about the Forum, please visit the FIAP Web site (<http://www.aps.org/FIAP/index.html>), or contact the chair, Kenneth C. Hass (khass1@ford.com).