The technology that aids scientists in the search for life on other planets might benefit your compute-intensive processing.

Grid Computing 101: What's All the Fuss About?

Is grid computing the Next Big Thing? IT's most insightful pundits can't decide. Some have called grid computing "a killer application," "the Internet's next phase," "changing the future of the workplace," and "too big to be ignored." Others proclaim that it is "an insanely overhyped technology," "an experiment in socialized computing," "years away," and "hopelessly gridlocked."

Scientists and engineers have used computing grids for years, and millions of desktop PCs run a grid application behind a popular screen saver. Your PC could be one of them.

If you've seen the SETI@home screen saver in action, then you've seen one example of grid computing. SETI, the Search for Extraterrestrial Intelligence (http://setiathome.ssl.berkeley.edu), uses a huge number of Internet-connected computers—most of them desktop PCs—to download and analyze radio telescope data, and to upload the results during idle times.

As of late-February 2004, SETI had scavenged 1.83 million *years* of CPU time from 4.9 million users in 226 countries. It had used this grid to perform 4.5×10^{21} floating-point operations. Experts say that 10^{21}—one sextillion—is the approximate number of grains of sand on all of Earth's beaches and deserts. It is just one order of magnitude shy of the estimated number of stars in the visible universe ("Star Survey Reaches 70 Sextillion," *CNN Science and Space*; http://www.cnn.com/2003/TECH/space/07/22/stars.survey).

Inside

WHAT'S IN IT FOR YOU

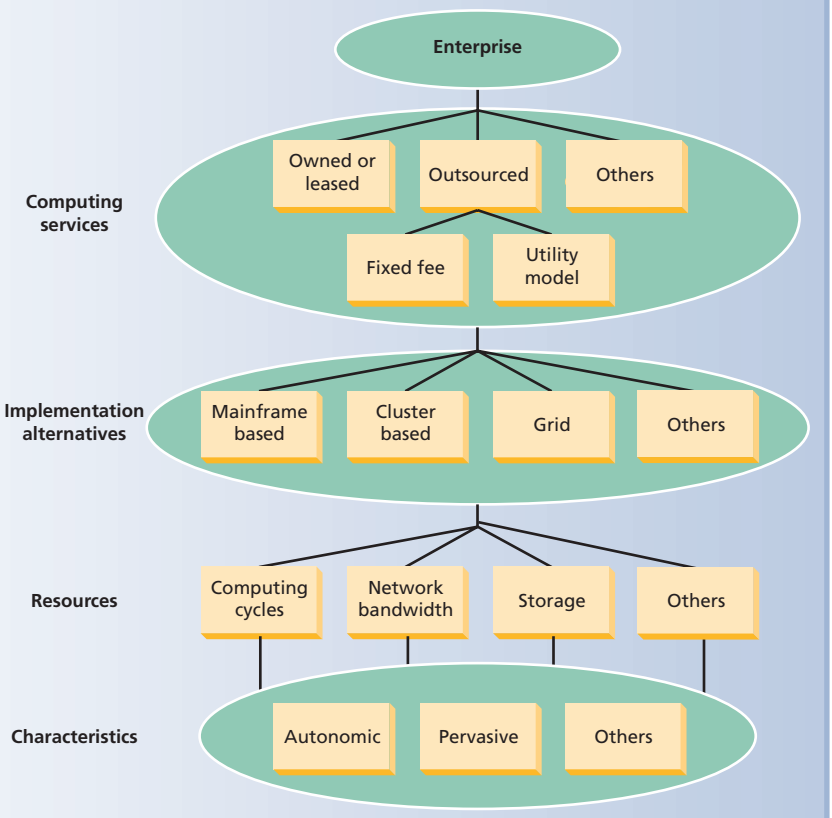
Talking about finding aliens is fun, but why should a harried IT manager care? Consider that in many enterprises, the computing cycles available on idle desktop PCs dwarf the computational resources in the corporate data center. So it's no surprise that IT managers are applying a SETI-like grid of idle PCs to enterprise IT applications, asynchronously downloading and processing data, and uploading results.

For example, some enterprises use a grid of desktop PCs to prepare hundreds of thousands of monthly customer billing statements after business hours, or to analyze individual financial portfolios and investment strategies. SETI-like grid middleware launches thousands of jobs; directs them to the enterprise's idle PCs for processing; traps errors and exceptions; redirects and restarts lost jobs; gathers results; and e-mails statements to customers before dawn.

Companies aren't limiting this resource sharing to just CPU cycles and files—some grids mine databases for insights into customer buying habits, while others opportunistically scavenge network bandwidth for large file transfers. Scientists use grids to systematically attack enormously complex problems in the life sciences and in biotechnology, particularly those that require the sharing of geographically dispersed data and computing resources. The Smallpox Research Grid uses a SETI-like model to analyze interactions between

1520-9202/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society

Figure 1. Computing-services taxonomy.



virus protein targets and a catalog of tens of millions of drug molecules (http://www.grid.org/projects/ smallpox). Individuals will eventually be able to attack their own problems using a peer-to-peer *consumer grid*, which would let anyone use idle PCs worldwide to perform truly distributed computing over public networks.

To understand grid computing, you'll need to know some terminology and the components, issues, and resources involved.

TERMINOLOGY

A look at the simple taxonomy in Figure 1 puts this technology into perspective. Enterprises have several options for providing computing services, such as purchasing, leasing, or outsourcing. They can implement each option in several ways, such as with a mainframe or a computing cluster. Each implementation has a pool of resources at its disposal (computing cycles or network bandwidth, for example), and each resource has a set of characteristics and attributes. The taxonomy will help you sort out the universe of grid and grid-related types of computing.

Grid computing

In this enterprise computing taxonomy, grid computing is an implementation. It really consists of a family of technologies for dynamically and opportunistically provisioning computing power from a pool of resources. The resource pool can include some or all of the following: computing cycles, file and data storage, caching, network bandwidth, databases and data warehouses, and application software. The resource pool can be geographically dispersed, and individual resources can vary widely in capability, capacity, and availability. Several grid-aware applications can share the resource pool. *Provisioning* includes methods and mechanisms for locating, authorizing, assembling, scheduling, releasing, and accounting for resources and their usage. Resource provisioning is dynamic—today's configuration will almost certainly not be the same as tomorrow's configuration. *Grid middleware* is the software glue that binds applications to resources.

Figure 2 shows how you can drill down into the grid implementation block of Figure 1. It shows that grids are multidimensional; you can classify them by their dimensions (Bart Jacob, *Enabling Applications for Grid Computing with Globus*, IBM, June 2003; http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/SG246936.html). One dimension reflects the types of resources that the grid can use. So-called *compute* and *data grids* share computational and data resources. Some grids share both and can also share network bandwidth, storage and caching resources, and application software.

A second dimension describes a grid's geographic or administrative reach, as in the terms *campus*, *statewide*, and *global grids*. A third dimension reflects how companies can obtain these resources. A service provider is likely to own or host pools of dedicated grid resources and strictly monitor and control their use, an example of a *utility grid*. SETI-like *scavenging grids* are far more opportunistic and depend on the cooperation of PC owners and departmental servers to cede control of their resources when they are available.

A fourth dimension reflects membership or partnership; examples include enterprise-owned and enterprise-internal *intragrids*, and multisite or multiorganizational *extragrids*. Other types are business partner, Internet, and peer-to-peer consumer grids. Some believe that *partner grids* might be the most cost-effective convergence platform for supplier or provider networks. For example, a large discount chain and its suppliers might decide to do

business via a partner grid rather than through thousands of bilateral agreements.

A fifth dimension reflects the type of application—whether it is for science, biology, sensors, or access, for example. So SETI has the dimensions of a compute, global, scavenging, Internet, and science grid. These variations add to the confusion.

Several factors drive the rising interest in grid computing. The first is resilience. Pools of resources spread over an enterprise or service provider's area provide redundancy and reliability.

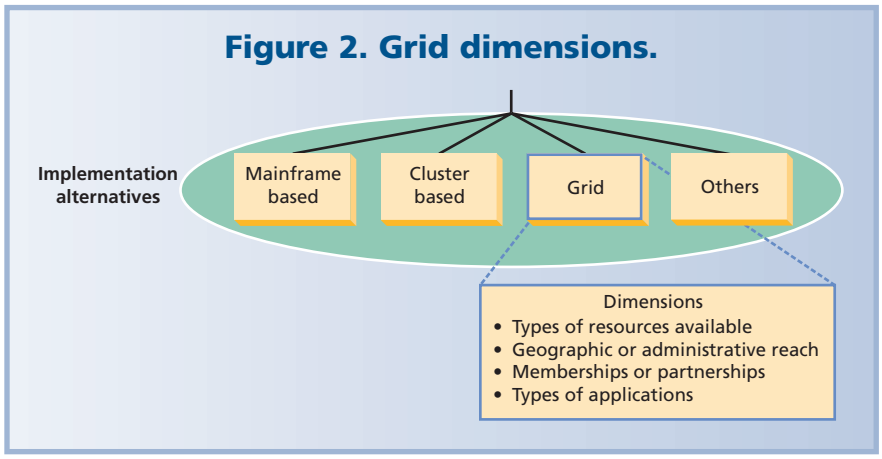


Figure 2. Grid dimensions.

The second factor is economics. An enterprise usually sizes its computing capacity and network bandwidth to accommodate peak demand and estimated growth. Computing resources, however, are also coarsely grained. So although you might need somewhat more than the compute capability of a dual-processor system, the next step up might be an eight-processor configuration. This much of a leap is expensive if you only need the computing power of three CPUs during weekly payroll processing, your peak-demand task. This cost is especially onerous considering that off-peak network bandwidth requirements rarely exceed 10 percent of peak demand (Andrew Odlyzko, "Data Networks are Mostly Empty and For Good Reason," *IT Professional*, Mar.-Apr. 1999, pp. 67-69).

Grid computing technologies help with this problem, allowing an enterprise or service provider to share processors, servers, and bandwidth. The advantages of sharing resources, or *multiplexing gain* in engineering parlance, is quantifiable. If the resource and user pools are large, and if the ratio of average to peak demand is small, then the rewards (the multiplexing gain) can be significant.

The third factor is hype. Type "+grid +computing" into Google and you'll find more than 2.5 million hits. Page down and you'll see terms like "utility computing," "cluster computing," "peer-to-peer computing," "autonomic computing," and "pervasive computing."

Ian Foster, professor of computer science at the University of Chicago and head of the Distributed Systems Lab at Argonne National Laboratory, defines a computing grid as a system that "1) coordinates resources that are not subject to centralized control 2) using standard, open, general-purpose protocols and interfaces 3) to deliver nontrivial qualities of service." (Ian Foster, "What is the Grid? A Three Point Checklist," Argonne National Lab.; http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid. pdf). Let's use Foster's definition and the taxonomy to see what the other terms have to do with grid computing.

Utility computing

People often confuse *utility* or *service-oriented* computing with grid computing. In utility computing, the idea is to offer computing resources as an on-demand service to customers in much the same way that utilities offer electrical, gas, water, and telephone services to households and businesses. The utility-computing service provider offers hosted computing resources, and the utility-computing customer pays for these services based on usage.

The idea isn't new—time-sharing bureaus have been around for decades. One distinction is that the on-demand computing resources can comprise a grid in the service provider's realm, and the grid can span several sites in the provider's service area. Outsourcing applications and services also isn't a new idea, but grid economics and scalability add a new dimension.

Utility-computing proponents believe that service providers will eventually host reliable and cost-effective grid-based services. They foresee grids replacing commodity servers in service provider networks similar to the way that servers have replaced mainframes in those same networks.

So in the taxonomy, utility computing is an outsourced service that a service provider might implement as a grid. (In large enterprises, the service provider could be the IT department.) The service provider views resources as a commodity. Pooling them into a grid provides redundancy and scalability, and improves reliability. Resource pools find wide use in science and engineering, so the concept is feasible.

Cluster computing

Cluster computing is another technology often confused with grid computing. *Clusters* are usually sets of processors interconnected in close proximity by high-speed, short-reach networks. They can be geographically distributed, but are more often closely coupled (usually in the same room). Clusters can also consist of heterogeneous processors and peripherals, but are more often homoge-

neous, and use ultra-high-performance, special-purpose interconnection networks. Clusters such as Beowulf (http://www.beowulf.org) are quite popular as commodity-priced alternatives to large supercomputers for scientific applications, and as mainframe replacements for enterprise applications. But a cluster isn't a grid.

In Figure 1's taxonomy, cluster computing is an implementation. It satisfies Ian Foster's second and third principles, but not the first. Computing clusters require a much higher degree of centralized control, a clear distinction between clusters and grids.

Other types of clusters (such as server farms) are also not really grids. Instead, think of a cluster of servers as a resource in the taxonomy. The IEEE's Task Force on Cluster Computing has a slightly different definition of clusters (http://www.clustercomputing.org/cluster_ white_paper.pdf).

Grid computing tends to be opportunistic—it must wait for resources to become available.

Peer-to-peer computing

Some consider peer-to-peer applications like Napster to be a type of grid computing. Others view the grid as an overarching service infrastructure that enables the location and use of any resource type. In this view, peer-to-peer computing is one type of application that uses grid services to advertise, find, and share files.

Foster and researcher Adriana Iamnitchi believe that grids and peer-to-peer computing have much in common, and might in fact be converging (Ian Foster and Adriana Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," http://people.cs.uchicago.edu/~anda/papers/foster_grid_vs_p2p.pdf). They argue that both of these approaches to distributed computing have the same objective—the use of geographically distributed resources—but each addresses the problem in different ways. The grid community tends to focus on top-down issues such as

- resource integration,
- performance,
- reliability,
- service quality, and
- security.

The peer-to-peer community tends to focus on bottom-up issues such as

- narrowly defined and specialized services, and
- support for tens of thousands of concurrent participants.

The ultimate solution will require elements from both communities.

One point of convergence is the *consumer grid*, a peer-to-peer system for sharing computational resources with a large community of Internet-connected individuals. Users launch applications in much the same way as they launch file-sharing peer-to-peer applications. But instead of sharing files, applications for consumer grids share computing power.

Autonomic computing

People occasionally lump *autonomic* or *recovery-oriented* computing with grid computing because both seek to provide a high degree of reliability, availability, and service quality (Foster's "nontrivial qualities of service"). Autonomic computing architectures monitor utilization and performance, tune and manage themselves, and adapt to failure. Some share resources and schedule tasks with other systems. The fundamental components of an autonomic computing system provide functions that computing grids will almost certainly need to operate effectively, such as the ability to recover lost computational subtasks.

As grids evolve, they might take on many of the characteristics of an autonomic computing system: self-monitoring, diagnosis, and adaptability in their youth; sophisticated resource scheduling and forecasting; just-in-time resource provisioning by adolescence (after a few years); and perhaps a vertebrate-like involuntary autonomic nervous system at maturity (after a few decades). So the real issue is the extent to which a grid has adopted the characteristics of autonomic computing.

Pervasive computing

Some consider *pervasive* or *everywhere* computing to be a type of grid computing because both must address issues related to coordinating computing resources that are free of centralized control. Pervasive computing architectures focus on how a pool of heterogeneous, autonomous devices and resources—smart appliances, smart spaces, personal digital assistants, cell phones, embedded systems, and sensors—interconnect and communicate. Grid computing is a top-down approach—an application needs these resources to accomplish this task. Pervasive computing, on the other hand, has more of a bottom-up focus: How does an application, for example, filter and analyze the chatter from a dozen monitors to say that Building 327 has no power? Pervasive computing systems might call upon grids to perform the analysis.

As with autonomous computing, grids will likely take on many of the characteristics of a pervasive-computing architecture as they mature, particularly at the interfaces between users and grid applications, services, and resources. You might also view a pervasive computing system (a sensor network, for example) as a grid resource in the taxonomy.

When Does a Grid Make Sense?

To illustrate how to measure the benefit that grid computing offers, consider an application that human resources personnel must launch for each newly hired employee. The application consists of 65 distinct tasks. For example, task two might be a privacy-compliance subtask, and tasks nine to 14 might post transactions to various enterprise subsystems—payroll, personnel, benefits, security, and traffic and parking. Assume that each task requires one minute of clock time, and that the application can perform some tasks independently and in parallel, as Figure A shows.

Basic Amounts of Parallelism

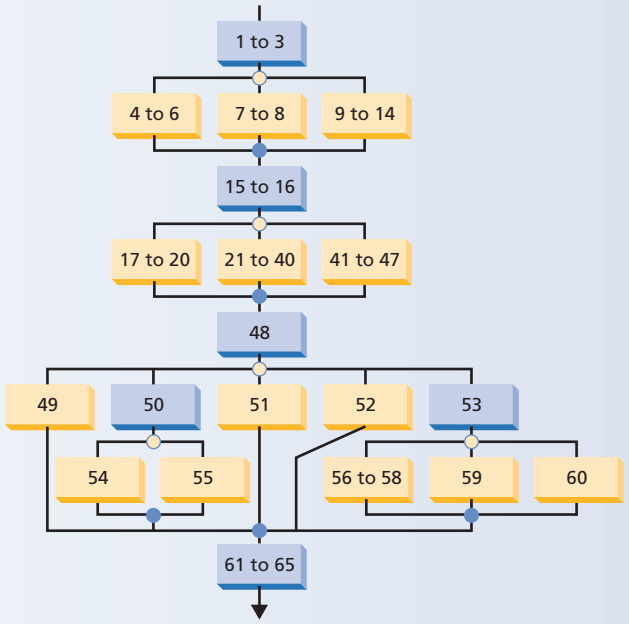
The system processes tasks one through three in sequence at a cost of three minutes. Then grid middleware locates, authorizes, assembles, and schedules resources for parallel execution of three sets of subtasks (tasks four to six, seven to eight, and nine to 14). The open circles in Figure A indicate the provisioning steps. After execution of tasks four to 14, the application calls on grid middleware to assemble the results, release resources, and schedule the next task, which the closed circles in Figure A indicate).

The parallel task sets below blocks one to three require 2, 3, and 6 minutes of clock time. The provisioning and release operations also require time, so the earliest that processing can begin for block 15 is 9 minutes after the application launches, plus whatever time the grid middleware requires. Continuing down the task tree in Figure A, you can see that the earliest completion time is 41 minutes after launch (plus middleware time), a 37-percent improvement over purely sequential execution. This example offers a potential time savings because you could easily decompose some of its parts into tasks that the system can perform in parallel, perhaps using idle desktop resources.

Embarrassing Parallelism

The second example is what supercomputing practitioners refer to as an embarrassingly parallel problem. Human resources prepares a benefits statement for each of 1,200 employees every month. The application consists of nine tasks per employee. If each task requires one minute of clock time, the total job requires about 7.5 days. However, if the application can scatter the tasks over 100 computing platforms, it can complete the job in a few hours. The provisioning, scattering, gathering, and release operations incur significant overhead; even so, the application can finish the job before dawn. This is an ideal grid application.

Figure A. Map of application tasks, showing how some tasks can execute in parallel.



Costly Scatter and Gather

The third example is one in which the scattering and gathering overhead is significant. The longest path through the task tree of Figure A has four scatter and four gather blocks. The work flow has an advantage of 65 – 41 = 24 minutes without overhead. If each scatter and gather block adds one minute, the 37-percent improvement falls to 25 percent. At 3 minutes per scatter or gather, this example breaks even. The lesson here is don't forget the overhead and don't forget the cost of gridifying applications.

Others

You might have heard of the following terms for computing or services: adaptive, agile, as-needed, collaborative, commodity, distributed, dynamic, on-demand, virtualized, ubiquitous, or organic. These terms are usually vendor speak for some variation of grid, utility, cluster, peer-to-peer, autonomic, or pervasive computing. Buyer beware.

The "When Does a Grid Make Sense?" sidebar gives an

Sample Products and Strategies

Here is a sampling of commercial grid-related products and strategies.

- **Access Grid (http://www.accessgrid. org): An ensemble of resources to support group-to-group interactions across a grid.**
- **Avaki (http://www.avaki.com/products/index.html): Data Grid software.**
- **Computer Associates (http://www3.ca.com/Solutions/Collateral.asp?CID=52328&ID=2835): Managing on-demand computing.**
- **DataSynapse (http://www.datasynapse.com): GridServer software.**
- **Entropia (http://www.entropia.com): PC grid computing.**
- **Fujitsu (http://pr.fujitsu.com/en/news/2002/04/22.html): Grid solutions for the sciences.**
- **Gateway (http://www.gateway.com/work/services/managed.asp?seg=cp): Processing-on-demand services.**
- **HP (http://h71028.www7.hp.com/enterprise/cache/6842-0-0-0-121.aspx): Adaptive enterprise.**
- **IBM (http://www.ibm.com/grid): Grid applications for vertical markets, including banking and financial, life sciences, automotive, aerospace, chemical, electronics, petroleum, education, and government.**
- **Kontiki (http://www.kontiki.com): Grid delivery technology.**
- **Microsoft (http://research.microsoft.com/~Gray/papers/Microsoft_and_Grid_Computing.doc): Microsoft's view of grid computing.**
- **Oracle (http://www.oracle.com): Database management for grids.**
- **Platform Computing (http://www.platform.com): Open-source, commercially supported tool kit for building grids.**
- **SAS (http://support.sas.com/rnd/scalability/grid/): SAS claims to support grid computing through its MP Connect functionality in its base product.**
- **Sun (http://www.sun.com/solutions/infrastructure/grid): Sun infrastructure solution for grid computing.**
- **Veritas (http://www.veritas.com/index.jhtml): Utility computing software.**
- **United Devices (http://www.ud.com/home.htm): Grid computing software and services.**
- **Wolfram (http://www.wolfram.com/products/gridmathematica/): Wolfram offers gridMathematica, a technical computing application for the grid.**

example of how to determine whether it makes sense to "gridify" an application.

COMPONENTS

Grid computing generally requires several components, which in total must be capable of performing the following high-level functions and services.

User or application interface

The grid interface provides users and applications with access services for secure sign-on, authentication (to verify identity), and access management. The latter confirms authority, coordinates access rights and privileges, exchanges credentials and certificates, and asserts trust relationships.

Resource management

Grids require mechanisms to manage the resources available to users and applications. Functions include discovery, the ability to locate a needed resource; advertising and registration (for resources joining or leaving the pool); resource attributes and characteristics (configuration, availability, cost, usage policy, and constraints); and state (operational status and load). Resources can include computing cycles, storage, caching capability, network bandwidth, databases, and application software. Two other functions are especially important:

- *Provisioning*. This function identifies and locates resources that are appropriate for jobs and tasks, and negotiates and authorizes their use. The application might make these decisions based on resource characteristics, service quality, availability, state, location (for delay-sensitive applications), and cost.
- *Data management*. As part of resource management, the grid middleware might have to move data about the grid via third-party transfers. This requires reliable and secure communications (mechanisms and protocols), and directory services.

Job management

Grids require mechanisms to manage user- and application-initiated jobs. A job consists of one or more of the following tasks:

- *Initiation*. The user or application submits one or more jobs. The application might also schedule jobs in advance.
- *Scheduling*. Schedulers dispatch jobs and tasks, and coordinate grid chores that are not subject to centralized control. Schedulers are hierarchical—

More on Grids

- **Business Process Grid Research/Working Group (http:// tab.computer.org/tcsc/businessgrid/): Group devoted to defining the business process integration and collaboration infrastructure.**
- ***Enabling Applications for Grid Computing with Globus*, Bart Jacob and colleagues, IBM, June 2003 (http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/SG246936. html): Contains tips on gridifying applications.**
- **"E-Science Gap Analysis," Geoffrey Fox and David Walker, 30 June 2003 (http://www.grid2002.org/uke-science/ gapresources/GapAnalysis30June03.pdf): An in-depth overview from Cardiff University, covering grid technologies, applications, tools, test beds, and issues; based on interviews with 80 scientists.**
- **Globus Toolkit, version 3 (http://www.globus.org): An open architecture and open source implementation of a grid infrastructure. It is a product of the Global Alliance, an organization that conducts research and development to create fundamental grid technologies. The organization focuses on four activities: research, applications development, software tools, and test beds.**
- **"Grid Services for Distributed System Integration," Ian Foster and colleagues, *Computer*, June 2002: Gives an authoritative overview of the evolving framework that will support grid services.**
- **GridLab (http://www.gridlab.org): A European-Commission-funded project seeks to develop an easy-to-use, flexible, generic, and modular Grid Application Toolkit.**
- **"Introduction to a Grid Architecture and Toolkit for Building Grid Solutions," Liang-Jie Zhang, Qun Zhou, and Jen-Yao Chung, 3 Dec. 2002 (http://www-106.ibm. com/developerworks/grid/library/gr-grid2/): Discusses the Open Grid Services Architecture.**
- ***Introduction to Grid Computing with Globus*, Luis Ferreira, IBM, Oct. 2003 (http://publib-b.boulder.ibm.com/ Redbooks.nsf/RedbookAbstracts/sg246895.html): IBM Redbooks offer general information that supports the use of IBM products.**
- **"Merging Grid and Web Services," Martin C. Brown, Sept. 2003 (http://www-106.ibm.com/developerworks/grid/ library/gr-web/): Discusses the benefits of using Web services in grid applications.**

one level might schedule jobs for a particular time, such as Saturday evening, for example. A second-level runtime scheduler might assign job tasks to particular resources. Third- (or resource-) level schedulers manage devices and computing clusters. Schedulers also arbitrate conflicts arising from contention for the same resource.

- *Monitoring*. Grids require the monitoring and control of jobs and resource assignments. Monitoring mechanisms trap, log, and diagnose errors and exceptions.
- *Completion*. As tasks and jobs finish, completion mechanisms assemble and distribute results (from various sites), and release resources.
- *Accounting*. If a grid must purchase resources, it will require accounting and billing mechanisms.

Application software

Developers require tools to build grid-enabled applications. They will need application programming interfaces, software development kits, software libraries and header files, and runtime environments.

ISSUES

The success of grid computing depends on fundamental issues in four main areas: economics, security, performance, and standards.

Economics

Pay close attention to the economics of grid computing. Jim Gray, a Microsoft distinguished engineer and the 1998 ACM Turing Award winner, has investigated the cost of moving tasks around a grid (Jim Gray, "Distributed Computing Economics," http://research.microsoft.com/research/pubs/view.aspx?tr_id=655). He argues that tasks have four demands—networking, computing, database access, and database storage—and that although the economics of these demands are somewhat interchangeable, investing in a grid makes sense only after considering the relative costs. For example, Gray found that one dollar buys about one gigabyte of wide-area-network (WAN) traffic, about 10 trillion CPU instructions, 10 million database accesses, or one gigabyte of disk space. In the time domain, Gray found that a minute of computation equals about one megabyte of network traffic.

Gray notes that WANs tend to be relatively expensive in most markets, so you should keep computing and data resources as close as possible to minimize WAN costs. Grid computing over the long distances typical of WANs might only be economical for compute-intensive applications. If networking costs are low—the enterprise owns the network and costs are sunk, or distances are short—then what constitutes "short" and "long" will vary.

Resources

Utility computing

- **Utility Computing (http://www.utilitycomputing.com): A portal serving the utility-computing industry.**

Cluster computing

- ***Cluster Computing: The Journal of Networks, Software Tools and Applications* (http://www.kluweronline.com/issn/1386-7857/contents/): Peer-reviewed journal covers networks, software, tools, and applications.**
- ***High Performance Cluster Computing: Architectures and Systems, Volume 1*; and *High Performance Cluster Computing: Programming and Applications, Volume 2*, Rajkumar Buyya, Prentice-Hall, 1999: A comprehensive survey of highly parallel architectures, programming, and applications.**

Peer-to-peer computing

- **"A Peer-to-Peer Approach to Resource Location in Grid Environments," Adriana Iamnitchi and Ian Foster; *Grid Resource Management: State of the Art and Future Trends*; Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, eds.; Kluwer, 2003.**
- **"On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," Ian Foster and Adriana Iamnitchi (http://people.cs.uchicago.edu/~anda/papers/foster_grid_vs_p2p.pdf): Researchers express their ideas on the eventual convergence of the two disciplines.**

Autonomic computing

- **"Autonomic Computing," W. Wayt Gibbs, *Scientific Am.*, 6 May 2002 (http://www.sciam.com/article.cfm?articleID= 000B0152-8C15-1CDA-B4A8809EC588EEDF&page Number=1): *Scientific American*'s calls autonomic computing "computing with a sense of self."**
- **Autonomic Computing (http://www.research.ibm.com/autonomic): IBM site addresses research and business implications.**
- **"Helping Computers Help Themselves," David Pescovitz, *IEEE Spectrum*, Sept. 2002.**

Pervasive computing

- **NIST Pervasive Computing Program (http://www.itl.nist.gov/pervasivecomputing.html): A National Institute of Standards and Technology portal focusing on tools and technologies.**
- **"The Semantic Grid" (http://www.semanticgrid.org/pervasive/): Explores the relationships between pervasive and grid computing.**
- **"Towards a Pervasive Grid," Vipul Hingne and colleagues, *Proc. Int'l Parallel and Distributed Processing Symp.* (IPDPS 03), IEEE CS Press, 2003: Identifies research issues and challenges to pervasive computing.**

Application pricing is another consideration. How do you deploy licensed applications on a grid? How do you gridify a mission-critical, commercial application? Relatively few applications can take advantage of grids. Ask about grid capabilities before you buy.

Security

Pay close attention to security; it could be a showstopper. You don't want a rogue SETI-like application browsing your hard drive for credit card numbers, and you don't want artifacts from your payroll processing to remain in some third party's cache. Security issues include secure sign-on, authentication, authorization, access rights, and privileges. You must also determine how to provide credentials, certificates, and certified delivery to maintain trust relationships in the exchange of data. Reliable and secure communications, perhaps with encryption, are also a requirement.

Protections must be in place to prevent passive intrusion, intercepted data transfers and transactions, tampering, and network disruptions. Maintaining confidentiality and privacy will also be issues if you are transferring personal data. Be sure to account for the additional cost and performance overhead these measures might incur.

Grids must deal with every security issue that any enterprise-owned or outsourced computing model faces. Be especially cautious about unproven technologies, because they can be more vulnerable to a security breach than more mature, time-tested technologies. Be aware of the trust relationships that you might need to share resources. In many ways, grids are the antithesis of the security-through-obscurity method that some enterprises practice. Assume that the enemy not only knows the system, but might also be sharing it with you (Lavanya Ramakrishnan, "Securing Next-Generation Grids," *IT Professional*, Mar.-Apr. 2004).

Performance

Foster's third checklist item is "to deliver nontrivial qualities of service," so performance is key. Some grid services

might fall short because the scattering and gathering steps can incur significant delay. And some problems just aren't amenable to grid computing. Parallel-processing performance depends on the slowest responder, and computing on far-flung resources is complicated and can add seconds to each transaction when processors and data are far apart. (Distributed database management is still a niche market for this very reason.) Grid computing tends to be opportunistic—it must wait for computing resources to become idle—which means that performance can be nondeterministic (difficult or impossible to finitely determine). Grid performance issues include resource availability and reliability, utilization and load, response time, delay, and delay variation (jitter).

Data integrity is another consideration. The main question is, Did a transaction complete successfully? Overhead (including checksums and additional processing) might be necessary to validate every data transfer. Grid computing has significant benefits, but can also incur significant additional overhead. Do the math.

Standards

Pay close attention to standards. Foster's second checklist item calls for "standard, open, general-purpose protocols and interfaces." The Global Grid Forum (GGF, http://www.ggf.org) and the Globus Alliance focus on architectures, software tool kits, and applications development. The GGF's Grid Service Specification outlines how grid clients and grid services interact (http://www-unix.globus.org/toolkit/draft-ggf-ogsigridservice-33_2003-06-27.pdf).

The Globus Alliance's Open Grid Services Architecture (OGSA) focuses on architectural issues related to interoperable grid services (http://www.globus.org/research/papers.html#OGSA). The Open Grid Services Infrastructure (OGSI) focuses on interoperable service features, interfaces, and protocols, including service invocation, management, and data and security interfaces. Together, OGSA and OGSI provide a common framework for grid services.

The Globus Toolkit (version 3) is an open architecture and open-source implementation of OGSI, with components for security, job submission and control, data transfer, databases, and consistent system and service interfaces (http://www-unix.globus.org/toolkit). The toolkit also includes a Java software developer kit and four Java hosting runtime environments. The Globus Toolkit Public License (GTPL) "allows software to be used by anyone and for any purpose, without restriction." Be aware that although many vendors are GGF members and endorse the Globus Alliance's goals, their products might not be open and interoperable. Also be aware that some grid standards are works in progress. If interoperability is important to you, ask about conformance to standards, and test before you buy.

In some ways, grid computing almost seems like a throwback to the old days of batch processing and mainframes, because most of today's grid computing occurs in off hours. But decentralizing the control; adding an opportunistic, scavenging flavor; and extending the geographic reach make grid computing a completely different type of animal, fraught with performance, management, security, and cost issues. You should consider these issues carefully before committing vital business processes to a grid. The goal of this article was to introduce basic concepts of grid computing. The "Securing Next-Generation Grids" article that follows in this issue, as well as articles in later issues, will focus on particular applications. ■

IT Professional*'s editorial board compiled this tutorial.*

For further information on this or any other computing topic, visit our Digital Library at http://computer.org/publications/dlib.