Editor: Geoffrey Fox, fox@csit.fsu.edu

COMPUTATIONAL GRIDS

By Geoffrey Fox and Dennis Gannon

LAST ISSUE'S WEB COMPUTING COLUMN DISCUSSED ONE VIEW OF MODERN DISTRIBUTED SYSTEMS, PEER-TO-PEER NETWORKS, WHICH HAVE GROWN UP TO SUPPORT COMMUNITIES AND INFORMATION SHARING. P2P WAS GIVEN AN IMPETUS BY THE new JXTA initiative from Sun, whose Web site (www.jxta.org) discusses interesting architecture issues and announces several open-source software components to support this computing model. Here, we will discuss computational and information grids, which Larry Smarr first popularized when he was director of the National Center for Supercomputing Applications at the University of Illinois.

P2P networks are built on the analogy of providing services in a community of peers. Grids, on the other hand, are built on the analogy of users tapping into ubiquitous computing and information resources, similar to plugging into an electrical grid. Together, these two powerful analogies cover much of the research and indeed commercial deployment of Web-based distributed systems. Underlying both concepts are two inevitable trends: using Internet hardware and software infrastructure to build the communication and control (middleware) of distributed systems, and using a Web browser as the user interface or portal to an application.

The grid analogy applies when computational scientists submit, monitor, and analyze simulation results or when scientists manage and store data in large-scale astronomy and nuclear-physics experiments. As electronic communities support distance collaboration, researchers around the globe are interacting and sharing resources—discussing the latest result of a simulation, the meaning of real-time data streaming in from satellites, or, more reflectively, the latest theoretical breakthrough described in a preprint just released on a colleague's Web site. There are similar applications in other fields, from business to pleasure. For instance, grids and P2P networks for both conventional and distance education can integrate teachers, students, curricula, and administrative resources in a Web infrastructure, for homework or grading purposes. We must support not only these tasks but also their synchronous or asynchronous integration. For example, a Web-linked sensor could stream data to an analysis program that fuses it with distributed resources and pipes the information through distributed filters; visualizing the process, the distributed project team could decide how to use the sensor or analyze the data. In an important paper (www.globus.org/research/papers/anatomy.pdf), Ian Foster and colleagues broadened the grids concept to support virtual organizations and thus link to the P2P arena. However, here we use grids to describe today's more limited view. Nevertheless, these concepts will evolve and overlap more and more.

Even in a particular field, say computational science or education, we need to build systems that can cope with a wide diversity of users and resources. If we customize our solution too much, it is hard to cope with the rapid changes in users' interests, the nature of the resources, and indeed the system's underlying technology. We address this by making system design or architecture choices. We use all relevant standards and perhaps try to work with the community on new standards where needed; the Internet's success has as much to do with well-chosen standards (TCP/IP, HTTP, XML, Java, and so on) as it does with the remarkable software and hardware developed in terms of them. We view all resources as distributed objects; we express capabilities as the composition of basic operations, which we define as services so that they can apply to a wide range of resources. This is a classic research problem, because users can be precise about their requirements only after they have experimented with initial systems. Thus, the process is proceeding as it usually does in science; every now and then, after many different prototypes have been built and tested, consensus develops as to best practice. Future Grid Computing columns will describe particular examples of grids; here we describe their basic principles and common features.

What is a computational grid?

Figure 1 presents a typical grid scenario. Independent clients are given access to a set of resources through a middle tier that routes information and implements the different services. We

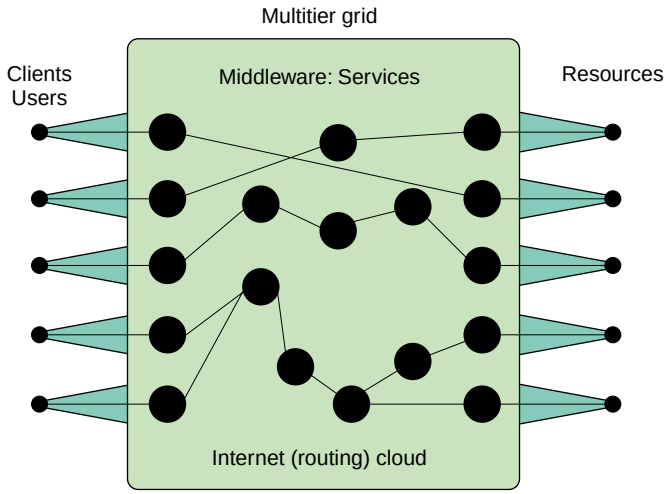

Figure 1. Multitier architecture of a computational grid. Seamless linkage of users to a suite of resources is mediated by a middle tier of Internet servers and brokers.

define *computational grid* as a collection of computers, online instruments, data archives, and networks that are connected by a shared set of services which, when taken together, provide users with transparent access to the entire set of resources.

Grids offer a wide range of services.

- *Single sign-on authentication, authorization, and security*. These services let users launch applications on any of the grid's resources by means of a standard authentication certificate. By using "proxy certificates," a user's application running on one machine can launch and communicate with another application running on another machine. Authorization services are automatically invoked to assure that the user or the user's agent has the authority and funding to access the resources.
- *A standardized grid-wide name space for files and other resources*. The World Wide Web uses the ubiquitous Uniform Resource Identifier as a wide-area name space for delivering content between Web servers and browsers. Web caches provide mechanisms that allow data to be replicated and moved to locations closer to the user. Tools such as FTP allow file transfer between resources, and the single sign-on enables the files to cross security domains.
- *Resource registration and discovery*. Grids tend to be supported through familiar Internet services such as DNS and other hierarchical schemes like LDAP. Many grids support access to "large" resources (such as supercomputers), and the issues of discovery and registration for such resources are clearer than for, say, the collection of MP3 files on an undergraduate's laptop (a more typical P2P registration and discovery problem). This view of computers and programs as resources is essentially equivalent to viewing them as distributed objects, accessible through one of the four major distributed-object systems—SOAP, Java, COM, or Corba. Making resource components such as software packages accessible in this fashion is sometimes called "wrapping as a distributed object."
- *Resource accounting*. Grids must enable users to be integrated into the accounting service of whatever resource they access. Each resource typically has some accounting service, possibly just through the ability to login, and the single sign-on lets users access that service through the grid.
- *Resource scheduling and coscheduling*. The latter is exemplified by distributed applications wishing to reserve simultaneous access to multiple resources such as online instruments and supercomputers. This is particularly difficult as traditional batch-queuing techniques will not work.
- *Job monitoring and performance services*. Users, grid managers, and applications need to be able to get information about the state of the grid at any given time. The service can tell users how their job is progressing or give application schedulers access to performance analysis tools such as network traffic monitors and resource load predictors. These are a special case of a more general information service, which provides access to the rich repository of knowledge needed to support computational science research. This includes online technical reports, details of the computational resources, and the dynamic data generated by job and performance monitors.
- *Specialized services*. Examples include job submission, programming, job parameter specification, and the ability to compose applications from a set of basic capabilities available on individual sites. The last service would have been used in the example described earlier of sensor data streaming thorough multiple filters to multiple visualization devices.
- *Event service*. The multiple servers defining an operational computational grid communicate by time-stamped messages defining state changes and control instructions. The event service handles the fault-tolerant and high-performance delivery of messages. This enables synchronization of state between multiple clients and resources. Also, a grid-wide trouble-tracking system can be built on top of the event service.
- *Object management*. The classic Web

requires only basic Internet services to operate, whereas a grid requires sophisticated tools to manage the many different objects that it supports. Users, devices (computers, sensors, PDAs, storage systems), jobs, events, and software are all objects to be managed as distributed objects. New distributed XML and database technology is being developed to address this critical area.

The grid's service-based architecture implicitly defines or assumes a network architecture that is relatively fixed and dominated by hierarchical access to a set of relatively well-defined resources. Substantial effort is needed to provide good quality of service to support the high-bandwidth communication needed to link from resources to both other resources and users.

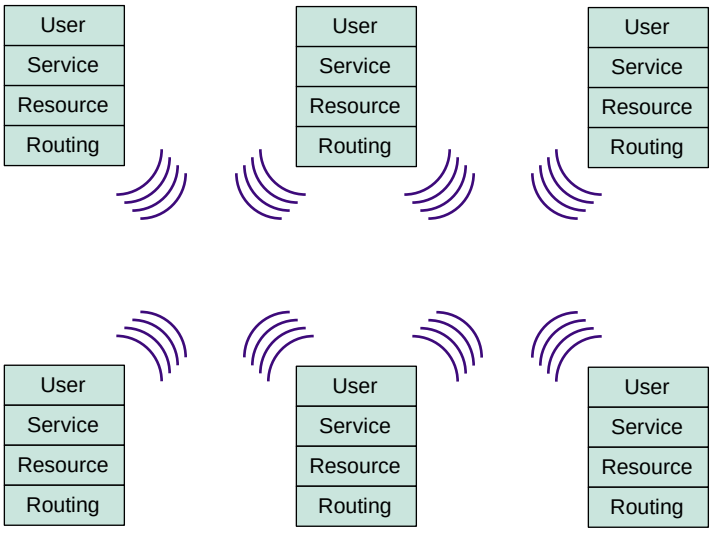

Figure 2. The architecture of a democratic P2P network where all nodes serve all purposes and link to all their peers.

How are grids being used?

Many different projects are implementing these services. The best-known system for building the software linking resources to the middle tier is the Globus project (www.globus.org), led by Ian Foster and Carl Kesselmann. Several groups are building prototype "end-to-end" systems implementing some grid services, with the variety of choices one can expect in a rapidly changing field. Sixteen such portals are described at www.computingportals.org/cbp.html in a relatively uniform fashion, so you can compare their different architecture and functionality choices. These and many other grid activities are being integrated by the Global Grid Forum (www.gridforum.org), which acts as a facilitator for the community to interact, develop standards, and establish best practices.

Several computational grid services are very specific to the computational science field, but there are also general services such as object registration, discovery and persistence, security, collaboration, events and transactions, and information systems. Thus, it is useful to consider computational grids in a general context and compare them to the P2P concepts discussed in the last issue. Whereas grids perhaps focus on access to resources, P2P networks focus on building communities—not just of people linked electronically but rather a fabric of users and resources forming a next-generation electronic community combining the best of instant messengers, grids, Napster-like services, and audio-video conferencing. Contrast the structured grid in Figure 1 with the "pure" P2P network shown in Figure 2, which shows each node acting as user interface, service provider, message router, and resource repository. Links between such multipurpose nodes tend to be dynamic. P2P nodes "multicast" information between themselves in the same way that rumors spread in a milling crowd; computational grids, on the other hand, make careful use of highly optimized networks in a way similar to how information flows in a hierarchical telephone tree or across levels in a structured organization.

At a superficial level, grids and P2P networks are optimized for structured and unstructured ways, respectively, of accessing resources and building communities. Both approaches have value. If you wanted to light a room, you could plug a lamp into the electrical grid powered by "supergenerators" on the electrical grid; alternatively, you could set up a P2P network of candle makers and go round to your friends to power the candlelight dinner that will charm your guests. Even their names reflect the fact that P2P networks and grids offer similar services; however, their different underlying models are reflected in their diverse functionalities and trade-offs.

Security in a P2P network is rather different from that in a grid, where users usually respect the privacy and anonymity of each other and their postings (resources). This has its special challenges: Electronic identities can be "spoofed," and reputation and trust (key P2P concepts) can be faked (one user could create multiple electronic instances that act as boosters for each other). Sun's JXTA implements security using a so-called "web of trust," where communities are bootstrapped by link-

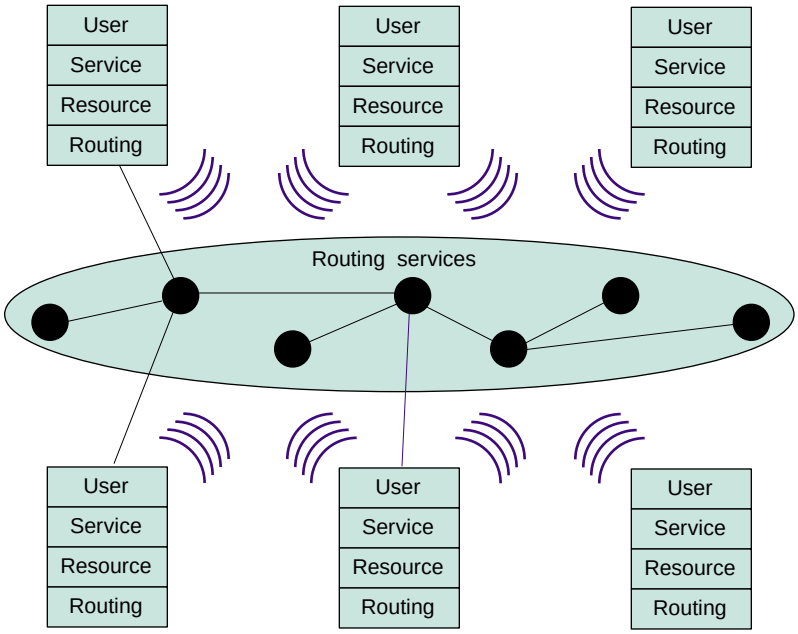

Figure 3. A hybrid linkage of peers. The strategy is analogous to information spreading by broadcasting internally to many groups, which are themselves linked by specialized lines (such as phones).

ing trusted subgroups. This contrasts with the resource-specific and centralized security of current grids. File access in P2P networks is illustrated well by Napster, which has a dynamic set of resources linked through rather different registration and discovery mechanisms from those used in grids. Accounting in P2P networks typically does not assume that each resource has its own natural accounting mechanism, as grids do. Instead, you must use "digital cash" or bartering (for instance, you can use access files on my file system if you give me the same privileges). Quality of service is as important in P2P networks as in grids, but it is handled as successful societies do, rather than through charging for large, well-defined resources, as in grids. QoS in P2P networks is often related to the "tragedy of the commons" (overuse of common, free resources in social groups) rather than to nifty new hardware multicast or routing priority as in grids. Networks in P2P systems are interesting structures, created dynamically; one can show how random routing leads to robust networks with good worst-case performance. Another concept from society, the "small-world effect," shows that a few "long links" (as opposed to nearest-neighbor ones) are essential to enable P2P networks to route information globally. You can find both of these ideas in the computer science research literature for parallel computing, where they were developed for precisely the reasons that they are valuable in P2P networks. However, the new incarnation of these ideas seems richer; we are not just deciding how best to link (at most) thousands of nodes. We are understanding how to link variable-sized subsets of nodes picked somewhat arbitrarily from the hundreds of millions of Internet clients and servers. There are other analogies with parallel computing; original systems such as the Caltech Hypercube integrate processing and networks on the same node (as in the P2P case). Currently, the highest-performance parallel systems tend, like grids, to build separate network and processing systems as in an idealized grid architecture.

We can expect future systems to combine ideas from both camps, leading to a hybrid architecture such as that shown in Figure 3. Thinking of capabilities as services should allow us to integrate these concepts by, for instance, building the security service to support the needs of both grids and P2P networks.

Geoffrey Fox is professor of computer science at Florida State University and associate director of its School of Computational Science and Information Technology; fox@csit.fsu.edu.

Dennis Gannon is a professor and chair in the Department of Computer Science at Indiana University and director of IU's Extreme Computing Lab. Over the past five years he has led the HPC++ initiative, which has produced a set of libraries for object-oriented runtime systems for large-scale parallel and distributed computing. In addition, he is a partner in the NSF Computational Cosmology Grand Challenge project, the DOE 2000 Common Component Architecture software tools group, and the NCSA Alliance. He also chairs the Java Grande Forum's Concurrency and Parallelism subgroup with Denis Caromel. His current research focuses on constructing distributed applications based on software component technology, integrating parallel and distributed programming systems, and designing problem-solving "workbenches" and distributed Grid services. He has a PhD in computer science from the University of Illinois and a PhD in mathematics from the University of California, Davis; gannon@iuvax.cs.indiana.edu.