IN SPITE OF ALL THE HYPE ABOUT PEER-TO-PEER COMPUTING, SOME USEFUL APPLICATIONS HAVE EMERGED FROM THIS NOT-SO-NEW COMPUTING MODEL.

Peering through the peer-to-peer fog

Peer-to-peer computing is a networking model that people have rediscovered, dusted off, shined up, and hyped to the point that you may be tempted to skip over even this article. But in spite of all that's been written about it, peer-to-peer computing does have some advantages over the client/server model of computing. Most people probably think that computer scientists invented peer-to-peer computing last year based on recent stories. A lot of companies have been quietly using peer-to-peer concepts either in production software or internally for years. What sparked all the attention on this way of connecting computers was the Napster phenomenon. It forced us to realize that there's another way to get information from one computer to another, and it doesn't have to involve a server. The Napster Web site uses an un-Web-like method of sharing files. Sure, some companies have suddenly discovered that their products are based on peer-to-peer technology and are proudly proclaiming it. However, some companies have dropped the term from their marketing while continuing to use the technology. Peer-to-peer computing is neither good nor bad; it's just another way of thinking about and solving a computer-networking problem.

DEFINING PEER-TO-PEER COMPUTING

Computers have been communicating with other peer computers since the early days of computer networks. People have overused and misused the term so much that it's hard for us to know what someone means when he says "peer-to-peer." One reason peer-to-peer computing has gained so much attention is that it breaks from the conventional and omnipresent client/server computer model. If you are reading this article on the Web, your Web browser (the client) requested a file containing this article from a computer (the server) that is part of the www.ednmag.com Web site. If the popularity of the Web is any indication, the client/server model is not going away soon.

Computers in a peer-to-peer network are "peers" in the sense that all the computers can act as clients and servers to each other. For example, Computer A can request a file from Computer B during one transaction, and, in another transaction, Computer B can request a file from Computer A. A strict definition of peer-to-peer computing dictates that no computer in the system can be solely a serv-

er. For those old enough to remember, a group of citizens band radios communicating with each other on the same channel is an example of a pure peer-to-peer system. Napster is not a pure peer-to-peer system because it relies on a central server to hold a directory of songs and their locations for its members. The reasons most people consider Napster a peer-to-peer system are because the actual transfer of an MP3 file occurs directly from one user's computer to another and because the MP3 files are distributed over thousands of computers rather than on one centralized server.

Clay Shirky of The Accelerator Group says that you can define peer-to-peer computing as "a class of applications that takes advantage of resources—storage, cycles, content, human presence—available at the edges of the Internet" (**Reference 1**). He also provides a litmus test for a peer-to-peer system: Does it allow for variable connectivity and temporary network addresses? Does it give the node at the edges of the network significant autonomy? The answer must be yes to both questions for the application to constitute peer-to-peer computing.

You can divide real-world examples into commercial and noncommercial applications. Some of the best-known examples of peer-to-peer computing are noncommercial, such as Gnutella (www.gnutella.wego.com). Justin Frankel and Tom Pepper invented the Gnutella protocol as an experiment in sharing files over the Internet. Frankel and Pepper also created Winamp and the company Nullsoft, which AOL (America Online) purchased in 1999. AOL didn't approve of the Gnutella project Frankel and Pepper had started in March 2000 and cancelled it soon afterwards. Some open source developers resurrected the Gnutella protocol and posted their work on the Web.

A Gnutella network is primarily a searching and discovery network. Each node can respond to a query however it wishes. For example, if you queried the network for "growing tomatoes" a node may respond with the name of a file in its shared directory containing information on tomato gardening. Another node may respond with a URL. Other nodes may ignore the query if they have no relevant information. Nodes must be running Gnutella software to be a part of a Gnutella network. Because there is no central server to connect to when you first start your Gnutella session, your computer must know of at least one other online Gnutella node with which to communicate. Finding these other

AT A GLANCE

- Peer-to-peer computing has many definitions, but it is essentially a decentralized network of computers interacting without the intervention of a server.
- Gnutella, Freenet, and SETI@home are examples of noncommercial implementations of peer-to-peer networks using individual PCs connected to the Internet.
- Large-microprocessor developers are using peer-to-peer techniques to simulate their huge chips.
- Public peer-to-peer networks face issues such as performance, scalability, security, reliability, and trust.
- Microsoft .NET and Sun's JXTA are efforts to standardize peer-to-peer application development.

SIMULATING THE POWER4 MICROPROCESSOR

By John Reysa, Senior Engineer, Power4 Simulation, IBM Corp

Developing IBM's 170 million-transistor Power4 microprocessor required a new simulation methodology due to its complexity (**Figure A**). The 64-bit Power4 microprocessor, operating at clock frequencies of greater than 1 GHz, is an eight-instruction-wide, superscalar, out-of-order-execution design that can have more than 200 instructions pending completion in each of the two on-chip processor cores. Considering the latches, arrays, and input signals on-chip, the unique states it has are approximately 1 followed by 4.2 million zeros. To put this number in perspective, if each state were a golf ball, the universe would be too small by a long shot to hold them. Simulating this many states is impossible.

The challenge instead was to identify the set of meaningful states that needed to be validated. IBM's simulation goal was to ensure first-pass hardware would successfully boot the operating system. By achieving this goal, the company could be confident the initial hardware would have sufficient functions and stability to enable productive testing to meet aggressive product schedules. To accomplish this goal, the company developed a test infrastructure accommodating multiple test philosophies in an automated environment that sustained more than 7 billion productive simulation cycles per week.

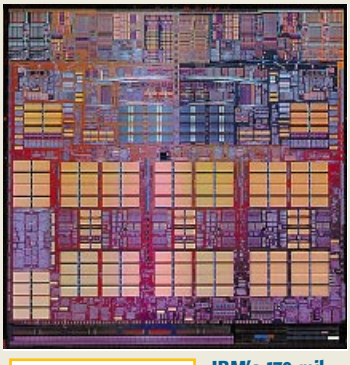


Figure A IBM's 170 million-transistor Power4 microprocessor was simulated using a form of peer-to-peer technology.

To accomplish this herculean task, IBM invested in a large "simulation farm" consisting of thousands of IBM AIX-based RS/6000 workstations and servers in Austin, TX, and Rochester, MN, and interconnected via a 100-Mbps Ethernet network. Of these systems, roughly half perform simulation, and the remainder resides on users' desks. Many of these users are not affiliated with the Power4 project. The project's terabytes of data are distributed across systems using a global DFS (distributed-file system), which allows Power4 team members dispersed across several sites within IBM to share project data. The data is stored on the simulation farm, enabling more effective resource balancing, data backups, and security. Users need not know where their data physically resides; they need to know just its DFS path name.

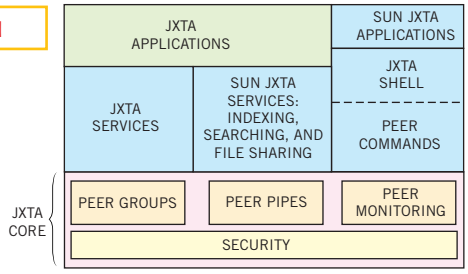
To more intelligently cover the large number of possible states, IBM generated many random-test cases. The test-case generator understands the machine's architecture and uses inputs, or "defs," which the simulation team developed to generate test cases to validate an implementation feature or exercise a corner

Gnutella node addresses can be tricky. Some Gnutella advocates have set up hosts to cache the addresses of Gnutella nodes. Having only a few places to look to find other Gnutella nodes simplifies getting onto a Gnutella network, but host caching is a compromise because it centralizes activity on a network.

You begin a Gnutella session by sending a "ping" message to hosts in your vicinity. When the hosts receive your ping, they acknowledge you by returning a pong. If you send a query to these hosts, they can respond with a reply to your query, forward your query onto other hosts, or both. Eventually, you will receive several replies to your request, and you can directly contact the host that responded with the information you want. The protocol ensures that a query is not forwarded to the same host more than once and that the query terminates after it has been forwarded a predetermined number of times.

Freenet (www.freenet.sourceforge.net) is another not-for-profit peer-to-peer network. It has the stated purpose of preventing censorship of documents while providing anonymity to its users. Freenet's supporters hope that, by distributing files across a decentralized network, information on its network will always be available despite efforts to shut down individual nodes. You initiate a request for a document by forwarding the request to a node you know and trust. If the node doesn't have the document you are looking for, it forwards the request to another node that is more likely to have it. Requests terminate after being forwarded a certain number of times. If a node has the requested document, it sends the document back through the chain of nodes from which the request came. Each node in the chain can store a copy of the document so that future requests for the document will be answered immediately. As messages are passed through the network, nodes learn the addresses of other Freenet nodes, and the network becomes more connected. Freenet has the advantage of making a popular document more available over time as copies of it proliferate over the network. On the other hand, thousands of users trying to access a single document on a centralized server reduces access to that document.

Figure 1



Sun based its JXTA model on peer groups, peer pipes, peer monitoring, and security.

Perhaps the most well known noncommercial peer-to-peer application is the SETI@home project (**Reference 2**). SETI (Search for Extraterrestrial Intelligence) is an area of research directed toward detecting intelligent life beyond Earth. The SETI@home project collects radio-telescope data in a 2.5-MHz band centered at 1.42 GHz, the *hydrogen line*. The hydrogen line is the resonant frequency of hydrogen molecules that fill interstellar space. SETI@home's proponents hope that aliens are clever enough to broadcast at the resonant frequency of the most common element in the universe.

case. Currently, random-test-case generation and simulation consume most of the computing resources.

On the client side, a job submitter controls how users submit the defs to the simulation farm. For example, a user can specify the def's relative priority and how many tests to submit before the def retires. In addition, a "fail threshold" instructs the job submitter to stop creating jobs once the threshold is exceeded for that def to limit the amount of fail data. On the server side, several submitter servers read the user-control directives and submit jobs to the simulation farm through a batch system, which balances the workload across the farm. The batch system detects failed systems and reschedules jobs as needed. The batch system permits users to allocate portions of the simulation farm to specific test areas. If a test area does not use its allocated portion, users submit other jobs in its place to maximize available computing resources.

Test-case execution begins when the batch system delivers the job to the simulation farm. The system gathers the data required for the simulation, typically tens to hundreds of megabytes. In most cases, a "model server" provides the source for the VHDL model and the C++ simulation driver. The model server replicates the data across several systems to handle the demand of thousands of machines requesting the data. The system obtains the remainder of the data through the DFS. The system invokes the random-test-case generator to create test cases from the def. A "cycle simulator" then executes the test case. The system sends the high-level passing or failing results of the simulation to an internally developed flat-file database that collects results from all simulations and replicates the data in different forms to quickly provide results for large queries. For example, to see the pass/fail regression results of 250,000 tests requires only one or two minutes. For failing tests, the debug files go to "results processors" that parse the generated data to gather interesting debugging information. The processed output files are then stored in DFS.

An engineer can analyze the fail data using a tracking tool that allows access to any information regarding the fail. The tracking tool correlates data from the output file, test-case files, debugging records, bug database, and flat-file database. Fast sorting and collapsing features and high-level debugging information permit multiple analysts to quickly and simultaneously attack the problem. An engineer can also rerun the test exactly as before with controlled variations to further isolate the failure or verify a fix. If an engineer finds a bug, he uses the fail-tracking tool to ensure that the test becomes part of a regression suite for testing successor models.

IBM's simulation farm allows the simulation of more than 1 billion cycles a day. The peak computing power of the farm is approximately 2 TFLOPS. This much computing capacity is equivalent to the fifth-ranked supercomputer in the www.top500.org list as of November 2000. Code automation allows the use of the available capacity of all IBM's workstations 24 hours a day every day. IBM believes that the effort and expense of the simulation environment is worth the investment. Power4's first-pass silicon successfully booted Unix, and the chip remains on its initial product-development schedule.

Once SETI collects this data, it divides it into 0.3-Mbyte *work units*. The SETI@home software that you run on your PC is a screensaver, which works on the data only when the PC is idle. Depending on the speed of your PC, the task may take an hour to several days to complete. When the software finishes, it connects to the SETI@home server, sends the results, and downloads another work unit.

These noncommercial examples may give you the impression that peer-to-peer computing is a computer-science experiment with a social bent. But companies such as Groove Networks and NextPage are determined to make peer-to-peer computing turn a profit. Ray Ozzie, the creator of Lotus *Notes*, founded Groove Networks in 1997. Groove Network's Groove software allows members of a project to collaborate directly and securely with each other via their networked PCs. Each member has a Shared Space that other members can access. The Shared Space can contain calendars, contact managers, instant-messaging tools, and a Sketchpad for illustrating a point for other members of the group. Groove synchronizes data between each member's Shared Spaces in the background through an encrypted connection. Groove Networks also offers a development kit for tying Groove into your current environment.

NextPage's Nxt 3 e-Content Platform allows you to manage, access, and exchange content across a Content Network, a secure distributed intranet of servers, through your Web browser. The content appears to you to reside in a single location but may in fact be in several locations around the world. For example, you can create a document and keep it on your PC, but other members of the Content Network can access your document as if it were on a central server. Nxt 3 also distributes document searches for efficiency by forwarding the query to the distributed servers, which perform the search on their own file systems.

Large-chip designers also use a form of peer-to-peer computing to simulate microprocessors containing tens of millions of transistors. Intel claims its Netbatch project saves $500 million a year by making the computing and storage capacity of 10,000 workstations available for simulation. IBM has been using a similar approach since the early 1990s (see **sidebar** "Simulating the Power4 microprocessor"). Developing chips with 100 million transistors wouldn't be feasible without a distributed-computing approach.

OVERCOMING SOME HURDLES

Although peer-to-peer computing offers advantages over centralized computing, it also faces some important obstacles. One of the most serious concerns is security. If you are creating a peer-to-peer network over the Internet, you need to consider the issues of trust, identity, accountability, and reputation. For peer-to-peer networks such as Freenet, in which

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to our information-request page at www.ednmag.com/info. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

Groove Networks
1-978-720-2000
www.groovenetworks.com
Enter No. 362

IBM
www.ibm.com
Enter No. 363

Intel
www.intel.com/cure
Enter No. 364

JXTA
www.jxta.org
Enter No. 365

Microsoft .NET
www.microsoft.com/net
Enter No. 366

NextPage
1-801-768-7500
www.nextpage.com
Enter No. 367

O'Reilly P2P Web site
www.openp2p.com
Enter No. 368

Peer-to-Peer Working Group
www.p2pwg.org
Enter No. 369

SETI@home
www.setiathome.ssl.berkeley.edu
Enter No. 370

Sun Microsystems
www.sun.com/p2p
Enter No. 371

OTHER COMPANIES MENTIONED IN THIS ARTICLE
Applied Meta Computing
www.appliedmeta.com
Hewlett-Packard
www.hp.com

SUPER INFO NUMBER
For more information on the products available from all of the vendors listed in this box, Enter No. 372 at www.ednmag.com/info.

anonymity is important, it is difficult to hold peers accountable for their behavior. One approach is to "punish" a misbehaving peer by having other peers ignore it. Identity usually involves verifying the identity of someone you already trust.

The problems facing the socially oriented peer-to-peer networks are trickier than networks running on a company's intranet. With a secure intranet, you worry less about trust and identity. But for Internetwide peer-to-peer networks, it takes only a few bad apples to spoil the fun. Freenet, Gnutella, and others are still working out these problems.

Other issues include scalability and performance. What happens when the number of peers in a network doubles? Does performance degrade or increase as you add new peers? Reliability is another concern. A peer-to-peer protocol has to deal with the fact that peers will come online and go offline in an unpredictable manner. The fact that a peer can disappear from the network without disrupting other peers is one of the advantages and even a definition of a peer-to-peer network.

DEVELOPING PEER-TO-PEER NETWORKS

Several companies are working to provide you with a standardized way of developing peer-to-peer networks. Sun's JXTA (derived from "juxtapose") is a peer-to-peer protocol that is designed to make it easier for you to take advantage of distributed computing. JXTA provides a way to communicate with nodes or devices on a network without the intervention of a centralized server. The JXTA model is based on four key concepts (**Figure 1**): peer groups, peer pipes, peer monitoring, and security. Peer groups are collections of peers or other groups. JXTA allows you to create groups and allow peers to become aware of and trust other peers. Peer pipes are based on Unix pipes and allow peers to connect to each other over the network in a distributed manner. Peer monitoring allows you to keep track of peer behavior and establish control policies among peers. The security features of JXTA allow you to ensure privacy, confidentiality, identity, and controlled access to services.

Microsoft is also in the peer-to-peer game. The company's .NET Framework allows you to create peer-to-peer programs based on four application models. The Web Services model allows you to quickly write a class that listens for and processes incoming requests and sends back objects that the peer application understands. The Windows Forms model is for writing Windows-based GUI applications for such tasks as logging in or sharing content. The Web Forms model allows you to write an application for returning HTML content to a peer. The Service Process model is for applications that don't use the HTTP protocol.

Intel, IBM, Hewlett-Packard, Applied Meta Computing, and other companies are also involved in the effort to develop peer-to-peer standards. In August 2000, these companies formed the Peer-to-Peer Working Group to "foster standards, create the necessary infrastructure for this technology, and develop applications that would help implement it." Intel also has a philanthropic project under way that uses peer-to-peer technology to help find cures to life-threatening diseases.

It may seem odd that Intel is promoting peer-to-peer computing because the technology can make more efficient use of PCs. But if the technology becomes easy to use and applicable to more computing problems, peer-to-peer computing may actually increase the demand for more powerful PCs. If you are an IS manager buying several new PCs for a staff that mostly does word processing, you will probably buy the least powerful computers that will still do the job. However, if your company has a way of using idle computing power or unused disk space, you may be able to justify buying the most powerful CPUs and largest disk drives you can afford. And that's good news for the processor and hard-drive manufacturers.□

You can reach Technical Editor Greg Vrana at 1-512-338-0129 fax 1-512-338-0139, e-mail gvrana@earthlink.net.

Author's biography

Technical Editor Greg Vrana was using peer-to-peer before peer-to-peer was cool at IBM in the early 1990s.

References

1. Oram, Andy, Editor, *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, O'Reilly & Associates Inc, 2001.

2. Moretti, Gabe, "How it works: Is Anyone Talking?" *EDN*, May 10, 2001, pg 32.