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

INTRODUCTION TO WEB COMPUTING

By Geoffrey Fox

THE WORLD WIDE WEB OFFERS MANY OPPORTUNITIES FOR COMPUTING IN SCIENCE AND ENGINEERING. INTERPRETING MY CHARGE AS DEPARTMENT EDITOR QUITE BROADLY, I'D LIKE TO REVIEW SOME OF THESE POSSIBILITIES. IF I OMIT AREAS of particular interest to you, please let me know.

Web clients as a computing resource

At the recent Supercomputing 2000 conference in Dallas (www.supercomp.org), Steve Wallach's keynote explained how it would be possible to combine optical interconnects with the inevitably improving microprocessor technology to build supercomputers with petaflops (10^{15} operations per second) performance within the coming decade. Of course, the same technology trend implies that the total computing power of clients interconnected by the Web will be some thousand times greater—exaops performance.[1,2]

In addition, most CPU cycles are "wasted" as desktop machines sit patiently waiting while their owners chat on the phone, drink coffee, or sleep. Thus, SC 2000 saw several commercial companies supporting what Larry Smarr called *megacomputing* in a panel discussion and what industry sometimes mysteriously terms *peer-to-peer computing* (www.peer-to-peerwg.org). Now, instead of purchasing a new departmental machine, you can send your computing tasks into the Internet "cloud" to be computed between keystrokes on some Web client. This powerful idea promises cheaper computing with greater total performance. Of course, this model has its limitation: the Internet cloud cannot easily support the large-scale parallel computations of our current teraflops and future petaflops dream. Loosely coupled machines often have lower communication bandwidth and always have much higher latency than classic massively parallel machines. Still, we can formulate many problems as a multitude of largely independent tasks—for example, factoring large numbers to break RSA cryptography and searching biological databases for patterns. Future editions of this column will examine these technologies and explore the applications and algorithms that can exploit megacomputing. Some of the key issues of security and fault tolerance are also fair game.

XML, Java Grande, and distributed objects

The Internet has spawned many technologies that are contributing to more powerful programming environments. Good examples are XML (www.xml.org) as the basis of new scientific data standards and Java as a potential new scientific programming language. Another is MathML, which will Web-enable us to render mathematical equations accurately in distributed documents, whiteboards, instant messengers, and other productivity tools. This is an example of ScienceML, a term that captures a plethora of emerging standards which enable the universal expression of scientific data and greater program interoperability. Our Web-enabled systems' input and output will no longer look like the cryptic `2I5,2F10.4` but will be the more understandable `<MYDATA YEAR=1999 NUM_ITERATIONS=10 POROSITY= 0.03 DEPTH=10 DEPTH_UNITS= METERS>Specify my ecology </MYDATA>`.

Nobody knows what the scientific programming language in 2010 will be, but to borrow a well-known cliché, I suggest that it surely will not be Fortran—or even be called Fortran. We can debate for a long time the software engineering advantages of languages such as C++ and Java versus Fortran's simplicity and high performance, but the next generation of students—immersed from the cradle on in the Internet, its technologies, and its power—will not care. The best of these students will not choose computational science unless this field adopts the very best information technology. Students will learn Java in high school and will not be happy to switch to Fortran for science and engineering computing.

Although Java was originally popularized in the form of applets to develop dynamic clients, its main industrial application is to build portable

middleware—large server-side applications to support database access and other e-commerce applications. Thus, it makes sense for this column to examine whether Java can be used to code large numerical simulations. The conferences and forums of the Java Grande group (www.javagrande.org) have identified several problems in this area, including Java's lack of a complex data type and rectangular arrays; moreover, there is the need for Java bindings to libraries in areas ranging from mathematical functions to parallel message-passing. Java's floating-point rules inhibit well-known compiler optimizations, while commercial offerings currently do not include Java compilers (just interpreters, albeit very clever, so-called just-in-time systems that use dynamic compilation). We have made progress—even though our field, which is 1% or so of the total computer market, will find it hard to influence commercial Java activities. This column can discuss this as well as the progress in C++, which has successfully tackled the issues of performance and expressivity needed for computational science.

C++ and Java are supported by powerful development environments, which could form the basis of better scientific-programming systems. Industry has also built distributed-object models (Corba, Java with RMI and Jini, COM, and SOAP(XML)) that appear very helpful in managing large-scale software and data systems. The important integration of distributed-object and Internet technologies is often called the Object Web. This provides a powerful model of the distributed systems that underlie modern Web computing. Perhaps we will discuss some of these points in future columns as well.

Computational science portals

Web systems are also used to produce integrated environments that support computing. These used to be called problem-solving environments or workbenches, but in Web lingo we usually call them *portals* to ride the current commercial thrusts bringing us the Yahoo portal (to everything), enterprise information portals, and so on. EIPs constitute a US$10-billion-a-year business, providing Web-based corporate information systems that access databases, email, and Web pages with a variety of communication tools. Here, I use the adjective "Web" to describe a distributed, networked system using Internet or Object Web technologies.

Nobody knows what the scientific programming language in 2010 will be, but it surely will not be Fortran.

In the computing field, we build a portal by assembling a network of Web servers and clients and then use them as an interface to computing resources. This approach (see, for example, https://hotpage.npaci.edu, www.cactuscode.org, and www.gatewayportal.org) gives users a single Web interface, enabling such capabilities as job application submission and monitoring, access to information resources, visualization of results, and application linking via data streaming or files. Such portals can also support communication tools such as (scientific) whiteboards, audio- and video-conferencing, and so on. Users can often choose which application to run, at what level of detail, and with what modifications (for example, by Web-inputting parameters or choosing a library).

We can consider such portals the front ends to computational grids (see www.gridforum.org), which are typically formed by a network of computational resources and thus are thought of as being more structured than the megacomputers discussed earlier. Increasingly, such grids use Object Web Technologies (see www.globus.org/cog). In this column, we will discuss both computing portals and their technologies.

I need your input!

We have looked at Web computing from three different points of view—stressing the computer, the technologies, and the systems formed from the Web. What interests you? Please send me any and all topics that appeal to you.

References

1. G. Fox, "Computing on the Web: New Approaches to Parallel Processing—Petaop and Exaop Performance in the Year 2007," http://old-npac.csit.fsu.edu/users/gcf/01/terri/SCCS_784/index.html (current 7 Dec. 2000).
2. W. Furmanski, "Petaops and Exaops: Supercomputing on the Web," *IEEE Internet Computing*, vol. 1, no. 2, 1997, pp. 38–46.

Geoffrey Fox is a computer science professor at Florida State University as well as the associate director of its School of Computational Science and Information Technology, director of its Computational and Information Science Laboratory, and chief technologist of its Office of Distributed and Distance Learning. Contact him at Computational Science and Information Technology, Florida State University, 400 Dirac Science Library, Tallahassee, FL 32306-4130; fox@csit.fsu.edu.