A Conversation with David Anderson

Supercomputing ON THE GRASSROOTS LEVEL

It's supercomputing on the grassroots level—millions of PCs on desktops at home helping to solve some of the world's most compute-intensive scientific problems. And it's an all-volunteer force of PC users, who, with very little effort, can contribute much-needed PC muscle to the scientific and academic communities.

At the forefront of this volunteer-computing effort is David P. Anderson, a research scientist at the U.C. Berkeley Space Sciences Laboratory, where he directs the SETI@home and BOINC (Berkeley Open Infrastructure for Network Computing) projects. SETI@home uses hundreds of thousands of home computers in the search for extra-terrestrial intelligence.

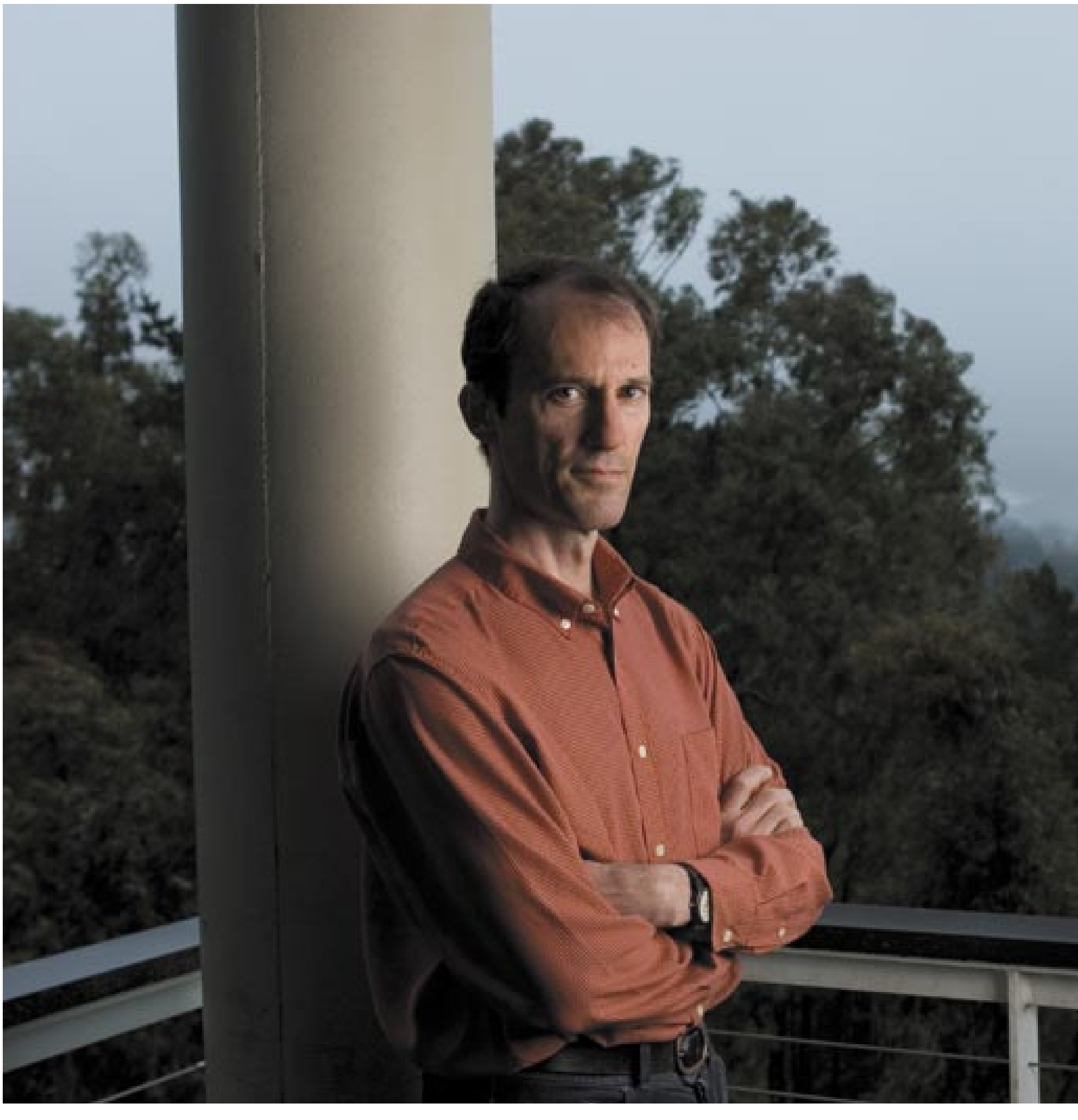
Anderson received graduate degrees in mathematics and computer science at the University of Wisconsin. From 1985 to 1992 he served on the faculty of the U.C. Berkeley computer science department. His research interests include distributed operating systems, realtime and multimedia systems, computer graphics, and computer music.

He agreed to sit down recently with *Queue* editors and discuss the value of the volunteer computing movement and the accelerating interest in it.

QUEUE Can you tell us a little about BOINC and the history behind it?

PHOTOGRAPH BY STEVE SKOLL

DAVID ANDERSON BOINC is a middleware system that lets scientists do supercomputing using idle time on PCs owned by the general public. This is an approach that we call "volunteer computing." Volunteer computing started in the mid-1990s with two groundbreaking projects, GIMPS (Great Internet Mersenne Prime Search) and distributed.net, in which tens of thousands of PCs work together on a single task. A couple of years later SETI@home and Folding@home started and attracted even more participants. (SETI, hosted by the Space Sciences Lab at U.C. Berkeley, is dedicated to the search for extraterrestrial intelligence. Folding@home is hosted by Stanford University and studies protein folding, a biochemical process with implications for fighting disease.)



There are hundreds of millions of Internet-connected PCs, and they're getting more powerful all the time, so volunteer computing can provide computing power and storage capacity way beyond what can be achieved with supercomputers, clusters, or grids. The volunteer computing approach works best for applications that don't need to move a lot of data between processors, but this limitation will diminish as the Internet gets faster.

Volunteer computing requires infrastructure software: a server that distributes work and keeps track of participants, and a client that communicates, manages computation and storage, and shows graphics. The early volunteer computing projects developed their own infrastructures, and two problems emerged: first, developing the software was harder and more expensive than anticipated; second, the resulting systems were closely tied to one project and therefore couldn't work together, so a PC could participate in only one project at a time—and when that project was down, the PC was idle.

In 2000 several companies, including Popular Power, United Devices, and Entropia, were formed with business models involving volunteer computing. These models didn't work out well, and the companies changed their focus or went out of business.

In 2002 we started the BOINC project to create an infrastructure that could be used by existing projects such as SETI@home and Climateprediction.net, a University of Oxford–led climate research project, and by a wide range of future projects as well. BOINC (http://boinc.berkeley.edu) is funded by the National Science Foundation, and our software is released under the Lesser GNU Public License.

Q Besides SETI@home and Climateprediction.net, what other projects are using BOINC?
DA Many projects are using BOINC, including Predictor@home, another protein folding project by Scripps Institute; LHC@home, a high-energy particle physics project from CERN (European Organization for Nuclear Research); and Einstein@home, a multi-institution project in gravitational wave observation. These projects have participation varying from tens to hundreds of thousands of CPUs.

Several other BOINC projects are currently under development, including Folding@home, Lattice (biotechnology from the University of Maryland), and Planet-Quest (astronomy). It's important to note that BOINC doesn't *host* these projects. Each project runs its own servers and is completely independent of other projects. BOINC doesn't review or authorize projects; we just supply the software.

Participants can sign up for whatever projects they want and can control the allocation of their resources among these projects. They install the BOINC client software on their PCs, and it takes care of downloading and running the project applications. You can think of BOINC as a specialized type of distributed operating system.

Q What would you say are the main goals and principles behind the design of BOINC?
DA Rather than try to be all things to all people, we concentrate on supporting a few existing projects. Each project has a unique set of requirements. We try to accommodate these requirements in a general, but more importantly, simple, way. We tend to develop functionality ourselves, rather than use existing software that is larger and more complex than we need. For example, we do our own XML over HTTP instead of using something like SOAP.

BOINC is largely self-contained, but we exposed a few important interfaces. For example, the BOINC client provides an RPC interface so people can build their own GUIs to control clients. BOINC projects export their statistics data (i.e., how much work has been done by each user, host, and team) so that people can develop Web sites showing this data.

The BOINC server software is designed to be scalable, so that you can support millions or tens of millions of participants by adding a few server machines. High availability is not a goal. If a BOINC project is down for a few hours, its participants get work from other projects. When it comes back up, an exponential back-off policy prevents connection overload.

Q Can you tell us what the main hurdles, technical and otherwise, were in developing BOINC?
DA From the technical and algorithmic points of view, there's nothing particularly exciting about BOINC. The hard parts have been release engineering and project management. The BOINC client software does a lot of things, and we want it to work on almost all computers. Thousands of things can interact with it—old operating system versions, missing libraries, HTTP proxies, ISPs that alter data streams, anti-virus software that locks files unexpectedly, timing issues resulting from slow CPUs, and so on. Developing and debugging BOINC applications, and BOINC itself, is difficult because we don't have direct access to most of the target hosts. This is exacerbated by the fact that BOINC has a short development and release cycle, so that new projects can come online quickly.

There's also the issue of customer support, both for the hundreds of thousands of participants, and for BOINC-based projects. These groups vary widely in knowledge and sophistication.

BOINC has only two full-time staff (me and one other programmer), so we've tried to fully leverage outside resources. We have volunteers doing programming, testing and QA, language translations for Web pages and GUI interfaces, documentation, porting, Web design, and graphic design. Some of these resources come from BOINC-based projects, and the rest are people from all over the world who want to help the volunteer-computing movement.

In each of these areas, we've spent a lot of time thinking about information flow and setting up appropriate mechanisms. At this point we have public projects for alpha and beta testing, several e-mail lists, two bug-tracking databases, and a dynamic FAQ system so that participants can provide customer support for one another. It's not perfect, but the important communication paths are there.

Q One would expect that dealing with so many platforms and operating systems would be challenging. How do you handle it?

DA The BOINC server runs on most Unix variants. It works right out of the box if you have current versions of MySQL, autoconf, Python, and so on. The client runs on Windows variants back to Win95, on almost all Unix variants, and on Mac OS X. The user interface is similar across platforms. The science applications may be new C++ programs or legacy Fortran code, and they may involve a pipeline of subprograms. They can show graphics as a screensaver or in a window; they also work on systems with no graphics. The installation/configuration process has simple defaults but provides lots of configuration and preference options for power users.

We use PHP-based Web interfaces instead of client-side GUI where we can. Our client-side GUI uses WxWidgets, a cross-platform toolkit that has worked well for us. Application graphics use OpenGL and GLUT. For Unix/X11 we developed a dynamic-library method so that applications work even on systems that don't have OpenGL or X11.

The BOINC client and application libraries are written in C++, and the source code is about 95 percent platform-independent. We develop using Visual Studio .NET on Windows and gcc on other systems. We rely heavily on autoconf and automake. The only platform-specific `#ifdefs` are for Windows.

As a solution of last resort for heterogeneity problems, we developed an "anonymous platform" mechanism. In this scheme users can build the BOINC software and applications from source. When their client talks to a server, it tells the server what applications it has, rather than asking for applications. This lets people participate in BOINC projects using any combinations of hardware and operating systems they want. It also supports people who run only software they've compiled themselves or who want to tweak applications to run faster on particular processors.

Q What are the major security concerns, and how are you addressing them?
DA Our highest priority is to protect participants, and in particular to prevent BOINC from being used as a vector for viruses. We use a digital signature mechanism for this. Applications aren't sandboxed, so participants must trust projects to provide valid applications.

Projects can't trust participants. There will always be hackers who return invalid results or who try to get unearned credit. This is addressed by a redundant computing mechanism that verifies each result by comparing it with independent instances of the same computation. For applications that use floating-point math, correct results may differ widely if they were computed on different platforms. BOINC gives projects the option of supplying a fuzzy comparison function or insisting that replicas of a task be sent to numerically equivalent hosts.

Some projects may want to hide their input or output data from participants, but this is not possible in general. Even if you encrypt data on the network and on disk, hackers can attach a debugger to the application to see the data unencrypted in memory.

Q We're hearing a lot about various forms of distributed computing these days, such as grid, utility, on-demand, and SOA (service-oriented architecture). What's your take on these approaches, and how do they relate to volunteer computing?
DA These terms have been widely abused by marketers, and their meanings have blurred. The underlying ideas originated in the academic computer science of the 1980s—client-server and RPC architectures, transparency with respect to location and heterogeneity, replication and fault-tolerance, market-based resource allocation, and so on. Many of these ideas didn't find their way into widely used systems. They lay dormant and are now being rediscovered and renamed.

It's particularly important to distinguish between grid computing and volunteer computing. Grid computing involves the sharing of resources within or between organizations such as universities, research labs, and companies. The resources are secure, trusted, and centrally managed. The organizations are accountable to each other, and their relationships are symmetric.

Volunteer computing, on the other hand, involves an asymmetric relationship: participants donate their CPU time to projects. Participants aren't accountable to projects, so the projects can't trust them. The infrastructure software must reflect this, as evidenced by the redundancy mechanism I mentioned earlier.

Q Following on that, where's the intersection (if any) between volunteer computing and such familiar technologies as clustering and SMP (symmetric multiprocessing)?
DA Clustering and SMP are all about bang for the buck, and there are two main ideas at work here. One is to exploit consumer-product R&D by using commodity hardware. This gives clusters an increasing edge over SMP, and it has created an interest in general-purpose computing using graphics coprocessors. The second trend is to generalize system software so that you can use any hardware resource for any purpose. Volunteer computing combines these two ideas.

Q What forces do you feel led to the rise of volunteer-

donated distributed computing for applications such as SETI@home?
DA Echoing my comments about "bang for the buck," volunteer computing arose because projects such as SETI@home needed $100 million worth of computing power but didn't have the money. But there's no free lunch—a project must give participants something in return for their computer time. People clearly get something out of running SETI@home—the science, the community, the competitive aspect. BOINC provides support for some of these, but at the end of the day, if you want to get lots of participants, you need to do something that people find worthwhile or interesting.

Q Where is BOINC headed over the next few years?
DA BOINC has several long-term goals. First, we would like to increase participation by two orders of magnitude, so that BOINC runs on most of the computers in the world. This means making it easier to participate. We're working on support for account management systems that let people browse and sign up for projects with a few mouse clicks. Second, we would like to see hundreds of projects in all areas of science. This will require us to reduce the system administration and database expertise needed to create and operate projects. Finally, we would like BOINC to handle a wider range of applications, including those that have very large data sets or that require low-latency communication. Some nice mechanisms have been developed in the context of file-sharing and instant-messaging networks, such as BitTorrent, jxta, and Jabber, and we hope to incorporate aspects of these into BOINC.

Q Are there any expected commercial applications?
DA BOINC's focus is on volunteer computing for academic scientific research, but it can be used for non-scientific applications, too. People are looking at Web indexing, game-playing (Chess, Go), movie-quality computer graphics rendering, and genetic algorithms for unusual purposes such as creating art. BOINC works fine for intra-organizational computing also. CERN, for example, is experimenting with BOINC for a data-intensive application on its internal PCs. Q

LOVE IT, HATE IT? LET US KNOW
feedback@acmqueue.com or www.acmqueue.com/forums