their personal disk image data. If the idle machine is suddenly needed for other purposes, the JPC instance can be "frozen" and the state moved to another idle physical resource.

Although this latter modus operandi is perhaps more difficult to imagine for interactive users, for whom the freezing and resuming would take too long over the Internet to be convenient, it makes a lot of sense for users who want to run many simultaneous virtual machines in parallel and without much interaction. This is the experience of users who currently use large "batch" farms to run massively parallel tasks, such as rendering frames of an animated movie, searching for drugs via molecular simulation, optimizing engineering design problems, and pricing complex financial instruments.

Ultimate Security

Allowing unvetted code to run on your machine is fraught with danger, and this danger is getting worse. There is a rapidly growing list of malicious software ("malware") on the Internet, variously known as "trojans," "keyloggers," "hostageware," "spamware," and "viruses." You could fall victim to data loss, identity theft, and fraud, and worst of all, might become implicated in a criminal offense if you did not exercise caution when running software downloaded from an unknown or unverified source.

For every security hole patched by the makers of the popular operating systems and Internet browsers, it seems two more grow in its place. Knowing this, how can you ever run code that might genuinely enhance your browsing experience or provide useful services?

Java code, when run in the Java Applet Sandbox, has provided this level of reassurance for over a decade. Add the extra independent layer of security represented by JPC and you have a double-insulated sandbox in which to run unvetted code. The JPC website (*http://www-jpc.physics.ox.ac.uk*) demonstrates how JPC can boot DOS and run a number of classic games inside a standard applet as part of a web page; in other words, they show an unvetted x86 (DOS) executable running in a completely secure container on any machine.

There is one major downside to running JPC within an applet sandbox: the security restrictions do not allow JPC to create classloaders, and therefore the dynamic compilation that gives JPC much of its speed is disabled. The good news is that by using the inherent flexibility of the JPC design, this can be circumvented without compromising security.

Java code in an applet sandbox can load classes from the network on demand as long as they come from the same server as the applet code originally did. Exploiting this, we have built a remote compiler that compiles classes on demand from JPC instances running in applets, sending them back to these instances when asked by the JVMs responsible for running them. The local JVM merely regards these classes as static resources that just happen to be needed rather later than the rest of the classes inside JPC, whereas in fact these classes have been compiled as needed by the JPC applet instances.