The Challenges of Virtualization

At a high level, operating system virtualization is used to multiplex several virtual machines onto a single physical machine. The virtual machines run operating systems; the physical machine can run operating systems. So what is the difference between a virtual machine and the physical machine?

Hardware is the most obvious difference. On a physical machine, the operating system has direct control of all attached hardware: network cards, hard drives, the graphics card, the mouse and keyboard. However, the virtual machines cannot have direct access to this hardware, or else they will undermine the isolation between each virtual machine. For example, a virtual machine (or VM) might not want other VMs to see what it stores in its secondary storage, or to read its network packets. Moreover, it would be difficult to ensure fair use in this scheme. You could have one device of each type for each virtual machine, but this would negate the cost and power savings of virtualization. The solution is to give each virtual machine a set of *virtual hardware*, which provides the same functionality as real hardware, but which is then multiplexed on the physical devices.

A more subtle difference arises when an operating system runs in a virtual machine. Traditionally, the operating system kernel is the most privileged software running on a computer, which allows it to execute certain instructions that user programs cannot. Under virtualization, the hypervisor is most privileged, and operating system kernels run at a lower privilege level. If the operating system now tries to execute these instructions, they will fail, but the way in which they fail is crucial. If they cause an error, which the hypervisor then traps, the hypervisor can correctly emulate the instruction and return control to the virtual machine. On the x86, however, there are some instructions that behave differently in lower privilege levels—for example, by failing silently without a trap to the hypervisor. This is bad news for virtualization because it stops operating systems from working properly in virtual machines. Obviously it is necessary to change these instructions, and the prevailing technique (at least, before Xen) was to scan the operating system code at runtime, looking for certain instructions and replacing them with code that calls the hypervisor directly.

Indeed, before Xen, most virtualization software aimed to make virtual hardware look exactly like physical hardware. So the virtual devices behaved like physical devices, emulating the same protocols, while the code rewriting ensured that the operating system would run without modifications. Although this gives perfect compatibility, it comes at a heavy cost in performance. When Xen was released it showed that by abandoning perfect compatibility, performance increased dramatically.

Paravirtualization

The idea of *paravirtualization* is to remove all the features of an architecture (such as the x86) that are difficult or expensive to virtualize, and to replace these with *paravirtual* operations