Because of JPC's hardware agnosticism, these stories apply equally well to non-x86 hardware, the ideal solution for thin clients, and users get their favorite x86 environment in all cases.

Flexible Auditing and Support

The screen, keyboard, and mouse of a running JPC instance could be viewed and overridden remotely by a suitably authorized system. Fraud could be effectively monitored by remotely collecting keystrokes and screenshots taken of suspicious activity to form an evidence base. Given the low-level hardware access made possible by JPC, this feature cannot be subverted by the technically able person who attempts to detect and remove monitoring software from within the emulated operating system. Even a user given administrator rights (on the guest system) would not be able to escape the monitoring, no matter how knowledgeable.

An auditing JPC system could simply record activity, scanning and flagging actions in real time, or go one step further and prevent certain actions. For example, in collaboration with suitable server software, a monitored instance could scan the video output for inappropriate images and then obscure them (or replace them with other content) at the virtual video card level. Such low-level monitoring means users could not subvert content protection systems by installing alternative viewing software.

Remote assistance could be far more effective if a help desk could literally see exactly what the entire screen was doing, and directly interact with the keyboard and mouse at the virtual hardware level. This would be possible even when JPC ran operating systems for which remote access was never implemented—for example, DOS, whose use is ongoing in many industries and countries around the world.

Flexible Computing Anywhere

Rather than run the main emulation on a local resource and merely import data from remote resources as necessary, the core emulation could be carried out on a central "JPC" server. Because JPC needs only a standard JVM to operate, this central JPC server could be based on hardware that is completely different from a normal x86 PC. There are some candidates already that could achieve this, and JPC has already been demonstrated on a 96 core Azul compute appliance. Other possibilities include Sun's Niagara-based servers and systems built from mobile phone technology (JPC has already booted DOS on a Nokia N95, an ARM11-based system).

But why centralize a server to run all these JPC instances? Presumably any resource on the Internet could run a JPC instance on behalf of anyone else, with the screen output and user input being piped via the network to the virtual machine owner. In this way of working, the world is viewed as "N" users with "M" machines, and there is no fixed mapping of hardware ownership relationships between users in the former group and machines in the latter. If a machine is idle, any one of the users can use it, remotely launching a JPC instance to work on