Running old programs in a virtual machine doesn’t necessarily create a good user experience

devblogs.microsoft.com/oldnewthing/20051005-09

October 5, 2005

Raymond Chen

Many people suggest solving the backwards compatibility problem by merely running old programs in a virtual machine. This only solves part of the problem.

Sure, you can take a recalcitrant program and run it in a virtual machine, with its own display, its own hard drive, its own keyboard, etc. But there are very few types of programs (games being a notable example) where running them in that manner yields a satisfying experience. Because most programs expect to interact with other programs.

Since the virtual machine is running its own operating system, you can’t easily share information across the virtual machine boundary. For example, suppose somebody double-clicks a .XYZ file, and the program responsible for .XYZ files is set to run in a virtual machine.

- Start the virtual machine.
- Log an appropriate user on. Hopefully, the user has an account in the virtual machine image, too. And of course the user will have to type their password in again.
- Once the system has logged the user on, transfer the file that the user double-clicked into the virtual machine’s hard drive image somehow. It’s possible that there are multiple files involved, all of which need to be transferred, and the identities of these bonus files might not be obvious. (Your word processor might need your spelling exceptions list, for example.)
- Run the target program with the path to the copied file as its command line argument.
- The program appears on the virtual machine operating system’s taskbar, not on the main operating system’s taskbar. Alt+Tab turns into a big mess.
- When the user exits the target program, the resulting file needs to be copied back to the main operating system. Good luck dealing with conflicts if somebody changed the file in the main operating system in the meanwhile.

The hassle with copying files around can be remedied by treating the main operating system’s hard drive as a remote network drive in the virtual machine operating system. But that helps only the local hard drive scenario. If the user double-clicks a .XYZ file from a network server, you'll have to re-map that server in the virtual machine. In all cases, you'll have to worry about the case that the drive letter and path may have changed as a result of the mapping.

And that's just the first problem. Users will expect to be able to treat that program in the virtual machine as if it were running on the main operating system. Drag-and-drop and copy/paste need to work across the virtual machine boundary. Perhaps they get information via e-mail (and their e-mail program is running in the main operating system) and they want to paste it into the program running in the virtual machine. International keyboard settings wouldn't be synchronized; changing between the English and German keyboards by tapping Ctrl+Shift in the main operating system would have no effect on the virtual machine keyboard.

Isolating the program in a virtual machine means that it doesn't get an accurate view of the world. If the program creates a taskbar notification icon, that icon will appear in the virtual machine's taskbar, not on the main taskbar. If the program tries to use DDE to communicate with Internet Explorer, it won't succeed because Internet Explorer is running in the main virtual machine. And woe unto a program that tries to `FindWindow` and then `SendMessage` to a window running in the other operating system.

If the program uses OLE to host an embedded Excel spreadsheet, you will have to install Excel in the virtual machine operating system, and when you activate the object, Excel will run in the virtual machine rather than running in the main operating system. Which can be quite confusing if a copy of Excel is also running in the main operating system, since Excel is a single-instance program. Yet somehow you got two instances running that can't talk to each other. And running a virus checker in a virtual machine won't help keep your main operating system safe.

As has already been noted, the virtual machine approach also doesn't do anything to solve the plug-in problem. You can't run Internet Explorer in the main operating system and an Internet Explorer plug-in in a virtual machine. And since there are so many ways that programs on the desktop can interact with each other, you can think of each program as just another Windows plug-in.

In a significant sense, a virtual machine is like having another computer. Imagine if the Windows compatibility story was "Buy another computer to run your old programs. Sharing information between the two computers is your own problem." I doubt people would be pleased.

For Windows 95, we actually tried this virtual machine idea. Another developer and I got Windows 3.1 running in a virtual machine within Windows 95. There was a Windows 3.1 desktop with Program Manager, and inside it were all your Windows 3.1 programs. (It wasn't a purely isolated virtual machine though. We punched holes in the virtual machine in order to solve the file sharing problem, taking advantage of the particular way Windows 3.1

interacted with its DPMI host.) Management was intrigued by this capability but ultimately decided against it because it was a simply dreadful user experience. The limitations were too severe, the integration far from seamless. Nobody would have enjoyed using it, and explaining how it works to a non-technical person would have been nearly impossible.

Raymond Chen

Follow