# How does Windows exploit hyperthreading?

devblogs.microsoft.com/oldnewthing/20040913-00

September 13, 2004

Raymond Chen

It depends which version of Windows you're asking about.

For Windows 95, Windows 98, and Windows Me, the answer is simple: Not at all. These are not multiprocessor operating systems.

For Windows NT and Windows 2000, the answer is "It doesn't even know." These operating systems are not hyperthreading-aware because they were written before hyperthreading was invented. If you enable hyperthreading, then each of your CPUs looks like two separate CPUs to these operating systems. (And will get charged as two separate CPUs for licensing purposes.) Since the scheduler doesn't realize the connection between the virtual CPUs, it can end up doing a worse job than if you had never enabled hyperthreading to begin with.

Consider a dual-hyperthreaded-processor machine. There are two physical processors A and B, each with two virtual hyperthreaded processors, call them A1, A2, B1, and B2.

Suppose you have two CPU-intensive tasks. As far as the Windows NT and Windows 2000 schedulers are concerned, all four processors are equivalent, so it figure it doesn't matter which two it uses. And if you're unlucky, it'll pick A1 and A2, forcing one physical processor to shoulder two heavy loads (each of which will probably run at something between half-speed and three-quarter speed), leaving physical processor B idle; completely unaware that it could have done a better job by putting one on A1 and the other on B1.

Windows XP and Windows Server 2003 are hyperthreading-aware. When faced with the above scenario, those schedulers will know that it is better to put one task on one of the A's and the other on one of the B's.

Note that even with a hyperthreading-aware processor, you can concoct pathological scenarios where hyperthreading ends up a net loss. (For example, if you have four tasks, two of which rely heavily on L2 cache and two of which don't, you'd be better off putting each of the L2-intensive tasks on separate processors, since the L2 cache is shared by the two virtual processors. Putting them both on the same processor would result in a lot of L2-cache misses as the two tasks fight over L2 cache slots.)

When you go to the expensive end of the scale (the Datacenter Servers, the Enterprise Servers), things get tricky again. I refer still-interested parties to the Windows Support for Hyper-Threading Technology white paper.

**Update 06/2007**: The white paper appears to have moved.

**Update 10/2016**: The white paper moved again.

Raymond Chen

**Follow**