Performance consequences of polling

devblogs.microsoft.com/oldnewthing/20060124-17

January 24, 2006

Raymond Chen

Polling kills. A program should not poll as a matter of course. Doing so can have serious consequences on system performance. It's like checking your watch every minute to see if it's 3 o'clock yet instead of just setting an alarm. First of all, polling means that a small amount of CPU time gets eaten up at each poll even though there is nothing to do. Even if you tune your polling loop so its CPU usage is only, say, a measly one tenth of one percent, once this program is placed on a Terminal Server with 800 simultaneous connections, your 0.1% CPU has magnified into 80% CPU. Next, the fact that a small snippet of code runs at regular intervals means that it (and all the code that leads up to it) cannot be pruned from the system's working set. They remain present just to say "Nope, nothing to do." If your polling code touches any instance data (and it almost certainly will), that's a minimum of one page's worth of memory per instance. On an x86-class machine, that 4K times the number of copies of the program running. On that 800-user Terminal Server machine, you've just chewed up 3MB of memory, all of which is being kept hot just in case some rare event occurs. Finally, polling has deleterious effects even for people who aren't running humongous Terminal Server machines with hundreds of users. A single laptop will suffer from polling, because it prevents the CPU from going to more power-efficient sleep states, resulting in a hotter laptop and shorter battery life.

Of course, Windows itself is hardly blame-free in this respect, but the performance team remains on the lookout for rogue polling in Windows and "politely reminds" teams they find engaging in polling that they should "strongly consider" other means of accomplishing what they're after.

Raymond Chen

Follow