Why does SetThreadPriority sometimes take a really long time?

devblogs.microsoft.com/oldnewthing/20161028-00

October 28, 2016

Raymond Chen

A customer had a video application that gathers information as it runs. Every so often, they need to take all this information, process it, and save it to a database. They perform this processing every fifteen seconds, but they found that whenever the processing occurred, it created enough of a CPU hiccup that the application would drop frames whenever the updates were being processed.

The customer solved the problem by calling `SetThreadPriority` with the `THREAD_MODE_BACKGROUND_BEGIN` priority level at the start of the update, and then `THREAD_MODE_BACKGROUND_END` when the update was complete. This reduces the impact of the update on foreground activity, and it seemed to do a good job of getting rid of the glitches.

And then later, they were investigating an unrelated problem, and as part of their investigation, they noticed that the very first time to enter background mode sometimes took over two seconds to complete. This initial update happened to take place as part of the application's startup, which meant that the long delay was directly affecting the application's startup time.

Further investigation revealed that this extra-long delay occurred only if the application was launched shortly after the machine was rebooted. If the system had been running for a while before the application was run, then the delay was not observed.

What's going on here?

What's going on here is that the system is doing exactly what you asked it to do. You said, "Please reduce the priority of this thread to minimize its impact on the rest of the system." And shortly after a system boots up, there's a lot of activity. Since you said that the thread should have minimal impact on the rest of the system, the thread gets stalled behind all those other activities.

"It's okay. Don't worry about me. I'll be fine."

"Hey, why aren't you paying attention to me?"

The thread said that it wants to run at very low priority, and the main consequence of running at very low priority is that it gets very low priority. The application didn't notice this most of the time because the system usually had enough spare capacity that it could deal even with very low priority things relatively promptly. But once the system gets busy, those very low priority things may end up having to wait a long time.

One the customer understood that this was a problem of their own creation, there were a few possible solutions.

One solution was to suppress the call to lower the thread's priority if the application is still starting up. During startup, update the database at normal priority instead of low priority.

What the customer chose to do was simply to suppress database updates completely when the application is starting up. The application will update the databases in the background after everything else is off and running.

Raymond Chen

Follow