Sure, I'm supposed to pass WT_EXECUTELONGFUNCTION if my function takes a long time, but how long is long?

devblogs.microsoft.com/oldnewthing/20111209-00

December 9, 2011

Raymond Chen

A customer contacted me to tell a story and ask a question. The customer discovered that in their code base, all calls to `QueueUserWorkItem` passed the `WT_EXECUTELONGFUNCTION` flag, regardless of whether the function actually took a long time or not. Their program creates a large number of work items at startup, and the result of passing `WT_EXECUTELONG-FUNCTION` for all of them was that the thread pool created a new thread for each queued work item, resulting in a bloated thread pool that thrashed the CPU. When he asked the other people on his team why they were passing the `WT_EXECUTELONGFUNCTION` flag unconditionally, they pointed to this article from 2005 on the importance of passing the `WT_EXECUTELONGFUNCTION` flag to the `QueueUserWorkItem` function. As I've mentioned before, Good advice comes with a rationale so you can tell when it becomes bad advice, but the people who applied my advice didn't understand the rationale and merely concluded, "It is important always to pass the `WT_EXECUTELONGFUNCTION` flag!" The `WT_EXECUTELONG-FUNCTION` flag is two-edged. If you pass the flag when queueing a task, then the thread pool will more aggressively create a new thread when that task is running. The upside is that other tasks don't get stuck waiting for your long-running task. The downside is that this creates more threads. And if you set the flag for *all* of your tasks, then you don't really have a thread pool at all, since you basically told the thread pool, "Run every task on its own thread, stat!" But this raises the question of "How long is long?" How long does a task need to run before you declare it a long-running task? There is no magic number here. The definition of a long-running task depends on the nature of your application. Let's consider, for concreteness, a task that takes one second. If this task is not marked as a long-running task, then the thread pool will wait for it to complete rather than creating a new thread. What are the consequences for your application of the thread pool choosing to wait for one second rather than creating a new thread? If your application doesn't generate tasks at such a high rate that a one-second pause would be a significant problem, then it's not a long-running task. On the other hand, if your application is a service that is handling thousands of requests per second,

then waiting for a one-second tasks means that a thousand tasks pile up in the meantime, and that may be enough to push your service to the brink of death because it has started falling behind on its processing and may never catch up.

Which category does your application fall in? That's for you to determine.

Raymond Chen

Follow