When there is a long line of people waiting for a shared resource, you want to investigate the person who is hogging the resource, not the people waiting in line for it

devblogs.microsoft.com/oldnewthing/20091204-00

December 4, 2009

Raymond Chen

If you see a long line of people waiting for a phone booth (note: this analogy assumes you remember how phone booths work), and you want to understand the reason for the long line, do you

- Go to a person waiting in line and begin your investigation there?
- Go to the phone booth (and the person inside) and begin your investigation there?

If there is a long line of people waiting for a single resource, a resource that there is not normally a long line for, you would probably look at the person who is using the resource to see if, for example, they are a chatterbox who will be on the phone for an hour, or if the phone is being repaired or is otherwise not working properly.

Similarly, if you find that in your 20-thread program, 17 of them are waiting for a single critical section, then you probably want to investigate the thread that owns the critical section to see whether (and why) it isn't releasing it.

> When testing a program, I encountered a hang that occurred after doing X. There are a few threads stuck in `LoadLibrary`, and about 40 threads stuck here:
>
> ```
> ntdll!KiFastSystemCallRet
> ntdll!ZwWaitForSingleObject+0xc
> ntdll!RtlpWaitForCriticalSection+0x132
> ntdll!RtlEnterCriticalSection+0x46
> ntdll!_LdrpInitialize+0xf0
> ntdll!KiUserApcDispatcher+0x7
> ```
>
> Here is one of the threads that is stuck in `LoadLibrary` [stack trace deleted]. You seem to be one of the people who work on the component that is trying to load the library. Can you investigate why the program is stuck?

This person picked one of the people waiting in line and decided that they were the ones responsible for the problem. But if course, that person waiting in line is just another victim of the person at the head of the line who is hogging the critical section. In this case, the critical section is the infamous *loader lock*. That it's the loader lock is obvious from the symptoms: What critical section does every thread require when it starts up? What critical section does `LoadLibrary` require?

You can use the `!critsec` debugger command to identify the current owner of the loader lock, and then start studying that thread to see what the hold-up is.

Note that I'm not saying that the thread that owns the resource is necessarily the culprit. The problem could be in the resource itself, or it could be in the pattern of usage associated with that resource. But starting your investigation with the owner of the resource is a good bet, because most of the time, the reason for the long wait queue is that the current owner of the resource isn't releasing it.

Raymond Chen

Follow