Is it okay to call MapViewOfFile on the same mapping handle simultaneously from different threads?

devblogs.microsoft.com/oldnewthing/20210702-00

July 2, 2021

Raymond Chen

A customer found that a bottleneck in their program was traced to the way they use the `MapViewOfFile` function. Specifically, they use an application-level lock to ensure that two threads do not call use the `MapViewOfFile` on the same mapping handle at the same time. The customer wanted to know whether this lock was required. Is it okay to call `MapViewOfFile` on the same mapping handle simultaneously from different threads?

Certainly the kernel needs to provide a basic level of protection against simultaneous use, in that the kernel can't crash or corrupt memory. That would be a serious security issue. The real question is whether making two simultaneous calls has the same effect as making the calls sequentially: Assuming sufficient resources are available, do both both calls succeed and create distinct views? Or can the kernel fail one of the calls with `ERROR_BUSY` or something like that?

The kernel permits simultaneous calls on the same handle, and they operate independently. You don't need to serialize your calls to `MapViewOfFile`.

The next question is whether removing the application lock will actually help anything. After all, if the kernel is doing its own synchronization, then all you did was move the bottleneck from the application lock to the kernel lock.

Yes, it will help. The kernel needs to take a lock to synchronize changes to the process address space, but the scope of that lock is narrower than the entire mapping operation and therefore will have reduced contention compared to what the application is imposing with its own lock.

Raymond Chen

Follow