The somewhat misguided question of whether MapViewOfFile or WriteProcessMemory is faster

devblogs.microsoft.com/oldnewthing/20130131-00

January 31, 2013

Raymond Chen

A customer asked, "Which is faster for copying data from one process to another; `MapViewOfFile` or `WriteProcessMemory`?"

This is one of those "Huh?"-type questions, where the customer has solved half of a problem and is looking for help with the other half, the half that makes no sense.

First of all, the question is malformed because `MapViewOfFile` does not copy any data at all. It takes existing data and maps it into a process. No copy takes place; both processes are seeing the same memory block, and if the memory is modified via one mapping, the change will be visible in other mappings.

It's like asking "Which company has better wireless telephone coverage: FedEx or Sprint?"

Okay, so maybe the question is really "Which is a faster way of transferring data between two processes?" (In the same way the FedEx/Sprint question might really be asking "Which company has a larger service area for me to communicate with my customers?")

But before you choose one or the other based on performance, you need to make the decision be appropriate from a correctness standpoint.

"We want to transfer some data from a client process to a server process. We found that mapping the shared memory causes more virtual memory to be consumed by the server, which is already constrained by the large number of components loaded into the server process. We were hoping we could switch to `WriteProcessMemory`, assuming the performance characteristics are acceptable."

Okay, now you sort of go "Whoa."

The `WriteProcessMemory` function requires `PROCESS_VM_WRITE` permission on the target process. That permission allows you to write to *any byte in the process*. If you allow a client process full write access to the complete memory space of a server, then the server is pwned!

It doesn't matter how much faster (if any) `WriteProcessMemory` is compared to `MapViewOfFile`, because you can't use it if you want to maintain any shred of security.

"I don't want my packages to get wet, so I taped the key to the front door with instructions to the FedEx delivery person to put the package in the living room."

That key is not a key to the living room. That is a key to *the entire house*.

If you want to reduce the amount of address space consumed by a `MapViewOfFile`, you can pass a nonzero value for `dwNumberOfByteToMap` and map a sliding window into the data rather than mapping it all at once.

And then looking at the question again, it's not clear that the `WriteProcessMemory` function will help in the first place: If the problem is address space exhaustion, then switching to `WriteProcessMemory` won't change your address space profile: The memory written by `WriteProcessMemory` must all be committed in the target process. Whereas `MapViewOfFile` lets the server control how much address space is consumed by the view, the `WriteProcessMemory` function requires it all to be committed up front.

We never did hear back from the customer, so it's not clear whether they understood that their question was confused and misguided, or whether they were just frustrated with us for "not answering their question."

Raymond Chen

Follow