On using FILE_FLAG_WRITE_THROUGH and FILE_FLAG_NO_BUFFERING for memory-mapped files

devblogs.microsoft.com/oldnewthing/20200819-00

August 19, 2020

Raymond Chen

A customer wanted to use the `FILE_FLAG_WRITE_THROUGH` and `FILE_FLAG_NO_BUFFERING` flags for a memory-mapped file, based on this guidance in the documentation for `CreateFile`:

> For this reason, the **FILE_FLAG_WRITE_THROUGH** flag is often used with the **FILE_FLAG_NO_BUFFERING** flag as a replacement for calling the FlushFileBuffers function after each write, which can cause unnecessary performance penalties. Using these flags together avoids those penalties.

The customer was concerned whether this combination of flags will affect data consistency.

Actually, the customer's problems with data consistency started even before they got around to worrying about these flags.

Since they are using a memory-mapped file, they don't have any direct control over when the memory gets written to disk. Page from memory-mapped files are written to disk at the operating system's discretion. Therefore, if they write information into two pages of a memory-mapped file, the pages can be written to disk in any order.

Since they're asking about data consistency, they must be worried about power loss or system crashes before the data can be written to disk. And since the pages can be written in either order, all four outcomes of two dirty pages are possible.

Page 1 written to disk	Page 2 written to disk
No	No
Yes	No
No	Yes
Yes	Yes

So much for data consistency.

Setting those flags on a memory-mapped file controls *how* the operating system writes the memory to disk, but it doesn't provide any control over *when* the memory is written to disk. And without that control, you don't really have data consistency.

Usually, when designing a system for consistency, you have a specific order in which data needs to be written to the disk. For example, you might decide to write the new data to the disk, and then once that's safe, you write new metadata (say, by updating an index) that causes the new data to become the active values, and the old data to be ignored. Those are the writes that would be able to take advantage of the write-through and buffering flags.

Bonus chatter: Using `FILE_FLAG_NO_BUFFERING` with a memory-mapped file doesn't really serve any purpose. The "no buffering" flag means that the writes go straight to the disk without being cached in memory. But the whole point of a memory-mapped file is to be cached in memory!

Raymond Chen

Follow