If I want to maintain a free list of pages, should I use MEM_RESET or MEM_DECOMMIT?

devblogs.microsoft.com/oldnewthing/20170317-00

March 17, 2017

Raymond Chen

A customer had a memory-intensive application, and one of the things they do to avoid address space fragmentation is to maintain a list of recently-freed memory and satisfying future allocations from that free list. (The free list has a cap to avoid permanent memory growth, because a cache with a bad policy is another name for a memory leak.)

The customer saw two possible ways of managing the memory on the free list:

1. Use `VirtualAlloc(MEM_RESET)` when the pages go on the free list. When a page is allocated from the free list, just hand it out. The old contents may be lost, but that's okay.
2. Use `VirtualAlloc(MEM_DECOMMIT)` when the pages go on the free list. When a page is allocated from the free list, use Use `VirtualAlloc(MEM_COMMIT)` to put memory back in place.

"We don't care about the contents of the free pages. We just want to reuse the virtual address space. We definitely don't want the pages swapped out to disk, because the contents are by definition uninteresting." The customer asked for advice on choosing between the two options.

One tweak you can make to the `MEM_RESET` algorithm is to couple it with `VirtualUnlock` to remove the page from the working set. This reduces physical memory usage while maintaining the commit charge for the page. The downside is that if you remove pages from the working set, then you will incur CPU cycles when the pages are soft-faulted in, and you may create contention on the working set lock.

As for `MEM_DECOMMIT` algorithm, one of the things you have to watch out for is that the `MEM_COMMIT` may fail, and you now have an error case to deal with, Mind you, this is probably an error case you already have to deal with, because if the free list is empty, you need to go allocate memory the old-fashioned way, and that allocation may fail.

On the other hand, repeatedly committing, accessing, and decommitting memory can be expensive. Decommitted pages go onto the system free list, and they need to be zeroed out by the operating system before they are given back. This is probably going to be significantly slower than `MEM_RESET`.

Those are some of the pros and cons. The customer is advised to run their own performance tests to see which way works best for them. Fortunately, this appears to be a relatively simple thing to test both ways because the behavior is isolated in the application's internal page manager.

Raymond Chen

Follow