If I zero out my memory pages, does that make them page in faster?

devblogs.microsoft.com/oldnewthing/20160907-00

September 7, 2016

Raymond Chen

In an earlier discussion of discardability, I noted that if you allocate some zero-initialized pages and never modify them, then the memory manager can just throw the memory away because it can "recover" the page full of zeros by simply zeroing out some memory. Commenter L wanted to know if this means that zeroing out memory can help program performance.

No, zeroing out memory does not help.

The reason the memory manager knows that it can throw the zero-filled memory away is that the page is not dirty.

When a page faults in (either because it's a zero-initialized page being accessed for the first time, or because it needs to be loaded from disk), the memory manager assigns a physical page, fills the page with the appropriate data (zeroes or data from disk), and points the page table entry at the page. It also clears the dirty bit, which is a special bit in the page table entry.

When you write to memory, the CPU sets the dirty bit in the page table entry for the page you wrote to. This bookkeeping is done automatically by the CPU and requires no effort from the operating system. When it comes time to page out the memory, the memory manager can do a quick check of the dirty bit, and if it's clear, then it knows that the memory was not modified since it was originally faulted in, which means that there are no changes that need to be written to disk. The next time the page faults in, it can be initialized the same way it was last time (filled with zeroes or loaded from disk).

If you manually zero out the page, then you set the dirty bit, and the memory manager will say, "Well, it looks like the program modified the memory, so I'll have to write it out to the page file so I don't lose it."

Now, in theory, the memory manager could add an extra step: Check if the page consists entirely of zeroes, and if so, then mark it as a zero-initialized page and discard it. The memory manager doesn't do this because it's such a low probability shot. The savings in the rare cases where a page being paged out happens to be a dirty page full of zeroes are outweighed by the cost of checking the page in all the cases where the page is not filled with zeroes.

In a sense, this is a self-fulfilling prophecy. The memory manager doesn't perform this check because it pays off so rarely as not to be worth the effort of checking. But since the memory manager doesn't perform the check, programs don't bother zeroing out pages when they are done with them. This creates a feedback loop and the net result is that nobody zeroes out pages because it doesn't help.

You can imagine an alternate universe where a positive feedback loop exists: The memory manager performs this check because it pays off, and the fact that the memory manager performs the check induces more programs to zero out their pages, which increases the payoff. But that's not the world we live in today.

And we're likely never to enter that world: Programs which want to tell the memory manager, "Don't bother paging this memory back out because I don't care what's in it" can convey this message by passing the `MEM_RESET` flag to the `VirtualAlloc` function.

Raymond Chen

Follow