Why is the virtual address space 4GB anyway?

devblogs.microsoft.com/oldnewthing/20040817-00

August 17, 2004

Raymond Chen

The size of the address space is capped by the number of unique pointer values. For a 32-bit processor, a 32-bit value can represent 2^{32} distinct values. If you allow each such value to address a different byte of memory, you get 2^{32} bytes, which equals four gigabytes. If you were willing to forego the flat memory model and deal with selectors, then you can combine a 16-bit selectors value with a 32-bit offset for a combined 48-bit pointer value. This creates a theoretical maximum of 2^{48} distinct pointer values, which if you allowed each such to address a different byte of memory, yields 256TB of memory. This theoretical maximum cannot be achieved on the Pentium class of processors, however. On reason is that the lower bits of the segment value encode information about the type of selector. As a result, of the 65536 possible selector values, only 8191 of them are usable to access user-mode data. This drops you to 32TB. The real limitation on the address space using the selector:offset model is that each selector merely describes a subset of a flat 32-bit address space. So even if you could get to use all 8191 selectors, they would all just be views on the same underlying 32-bit address space. (Besides, I seriously doubt people would be willing to return the the exciting days of segmented programming.) In 64-bit Windows, the 2GB limit is gone; the user-mode virtual address space is now a stunning 8 terabytes. Even if you allocated a megabyte of address space per second, it would take you three months to run out. (Notice however that you can set `/LARGEADDRESSAWARE:NO` on your 64-bit program to tell the operating system to force the program to live below the 2GB boundary. It's unclear why you would ever want to do this, though, since you're missing out on the 64-bit address space while still paying for it in pointer size. It's like paying extra for cable television and then not watching.)

Armed with what you have learned so far, maybe you can respond to this request that came in from a customer:

> Oen of our boot.ini files has a /7GB switch. Our consultant told us that we should set it to 1GB less than the system memory. Since we have 8GB, 8GB – 1GB = 7GB. The consultant said that setting this value allows an application to allocate more than 2GB of memory. We would like Microsoft to comment on this analysis.

Raymond Chen

Follow