Myth: You need /3GB if you have more than 2GB of physical memory

devblogs.microsoft.com/oldnewthing/20040811-00

August 11, 2004

Raymond Chen

Physical memory is not virtual address space.

In my opinion, this is another *non sequitur*. I'm not sure what logical process led to this myth. It can't be a misapprehension of a 1-1 mapping between physical memory and virtual memory, because that mapping is blatantly not one-to-one. You typically have far more virtual memory than physical memory. Free physical memory doesn't have any manifestation in any virtual address space. And shared memory has manifestations in multiple virtual address spaces yet correspond to the same physical page.

Though this brings up a historical note.

In Windows/386, the kernel just so happened to map all physical memory into the kernel-mode virtual address space. There was a function `_MapPhysToLinear`. You gave it a physical memory range and it returned the base of a range of linear addresses that could be used to access that physical memory. Some driver developers discovered that the kernel mapped all of physical memory and just handed out pointers into that single mapping. As a result, they called `_MapPhysToLinear(0, 0x1000)` and whenever they wanted to access physical memory in the future, they just added the address to the return value from that single call. In other words, they assumed that

```
_MapPhysToLinear(p, x) = _MapPhysToLinear(0, x) + p
```

In Windows 95, the memory manager was completely rewritten and the above coincidence was no longer true. To conserve kernel-mode virtual address space, physical memory was now mapped linearly only as necessary.

Of course, the drivers that relied on the old behavior were now broken because the undocumented behavior they relied upon was no longer present.

As a result, when it starts up, Windows 95 looks around to see if any drivers known to rely on this undocumented behavior are loaded. (Windows 3.1 didn't support dynamically-loaded kernel drivers so looking at boot time was sufficient.) If so, then it went ahead and mapped

all of physical memory into the kernel-mode virtual address space to keep those driver happy. This wasted virtual address space but kept your machine running.

I can already hear people saying, “Microsoft shouldn’t have made those buggy drivers work. They should have just let the computer crash in order to put pressure on the authors of those drivers to fix their bugs.” This assumes, of course, that the cause of the crash could be traced back to the buggy driver in the first place. A very common manifestation of a stray pointer in kernel mode is memory corruption, which means that the component that crashes is rarely the one that caused the problem in the first place.

For example, nearly all Windows 95 bluescreen crashes in `VMM(01)` are caused by memory corruption. `VMM(01)` is the non-swappable part of the Windows 95 kernel which is where the memory manager lives. If a driver corrupts the kernel-mode heap, a bluescreen in the memory manager is typically how the corruption manifests itself.

Raymond Chen

Follow