Is there a way to change the minimum size for large pages?

devblogs.microsoft.com/oldnewthing/20160614-00

June 14, 2016

Raymond Chen

We discussed Large Page Support a few years ago. In practice, the large page minimum size is 2MB. A customer wanted to know if there was a configuration setting to change the minimum size for a large page.

The large page minimum size isn't chosen by Windows. It's chosen by the processor.

Recall that the paging structure is hierarchical. The x86 and x64 use a three-level page table;[1] ARM uses a two-level page table. For concreteness, we'll talk about the x86 page table structure, and let's assume that all pages are present.

When the processor tries to access a linear address, the address is broken down into four pieces:

- Bits 30 and 31 form a two-bit value that selects a page directory from the page directory pointer table.
- Bits 21 through 29 form a nine-bit value that selects a page table within the page directory.
- Bits 12 through 20 form a nine-bit value that selects a page within the page table.
- Bits 0 through 11 form a twelve-bit value that selects a byte within the page.

In pseudo-C++, it goes something like this:

```
byte& FindPhysicalAddress(intptr_t linearAddress)
{
  // assuming all pages are present and there are no funny games.
  return cr3[(linearAddress >> 30) & 0x003]
            [(linearAddress >> 21) & 0x1FF]
            [(linearAddress >> 12) & 0x1FF]
            [(linearAddress      ) & 0xFFF];
}
```

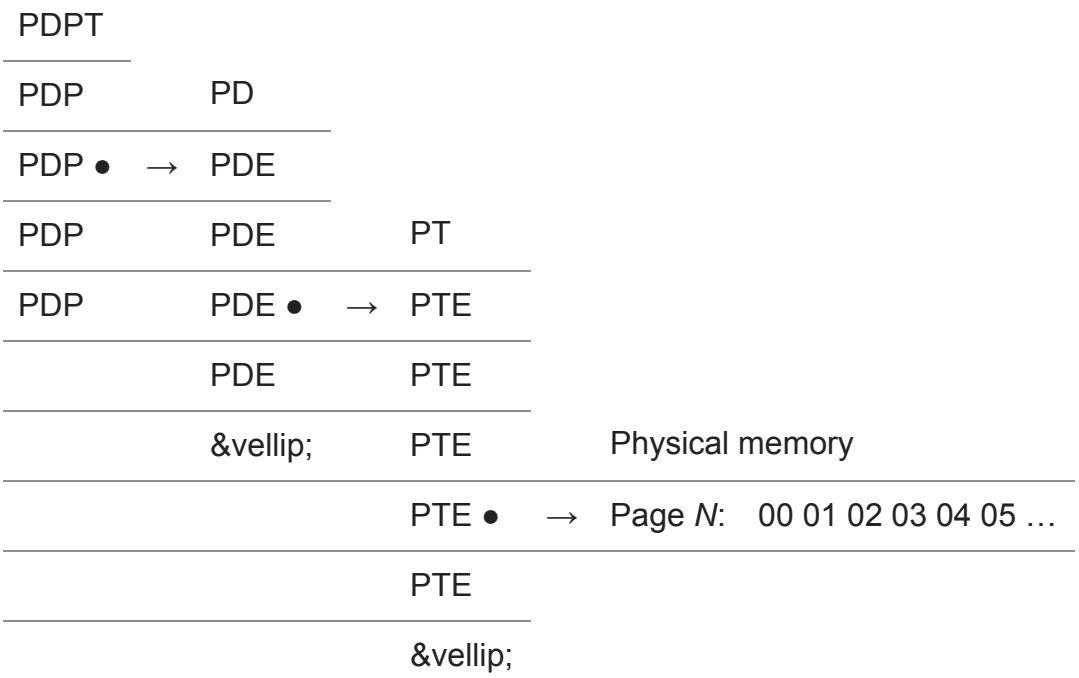
At the bottom of the hierarchy are pages. Each page is 4KB of data, and the bottom 12 bits of the address are an offset into the page.

The next level up is a *page table*. Each page table is one 4KB page in size, broken down into entries called *page table entries* which are each 8 bytes in size. A little math tells you, therefore, that each page table contains 512 page table entries. Since each page table entry describes one 4KB page, an entire page table describes 512 × 4KB = 2MB.

The next level up is a *page directory*, which is filled with 512 *page directory entries*. Each page directory entry describes one page table.

The top of the hierarchy is a *page directory pointer table*, which holds four *page directory pointers*.

Here's a diagram.



You can see how the hierarchy can be extended further if necessary. Basically, each level of the hierarchy describes some number of objects one level down, until you finally reach pages, which contain the actual data.

The trick with large pages is that instead of having a page table filled with 512 entries that say "Entry 0 is page *N*; entry 1 is page *N* + 1; entry 2 is page *N* + 2; … and entry 511 is page *N* + 511," you say "Hey, let's just save everybody some trouble and I'll just say "Pretend you have a full page table of contiguous pages starting at page *N*."

In other words, we start with this:

PDP		PD					
PDP ●	→	PDE					
PDP		PDE		PT		Physical memory (contiguous)	
PDP		PDE ●	→	PTE ●	→	Page *N*:	00 01 02 03 04 05 …
		PDE		PTE ●	→	Page *N*+1:	00 01 02 03 04 05 …
		⋮		PTE ●	→	Page *N*+2:	00 01 02 03 04 05 …
				PTE ●	→	Page *N*+3:	00 01 02 03 04 05 …
				PTE ●	→	Page *N*+4:	00 01 02 03 04 05 …
				⋮			

and we create a special page directory entry that says “Hey, let’s just pretend there’s a full page table of contiguous pages starting at page *N*“:

PDPT							
PDP		PD					
PDP ●	→	PDE					
PDP		PDE		Imaginary PT		Physical memory (contiguous)	
PDP		Page *N* through *N*+511 ●	→	Imaginary PTE ●	→	Page *N*:	00 01 02 03 04 05 …
		PDE		Imaginary PTE ●	→	Page *N*+1:	00 01 02 03 04 05 …
		⋮		Imaginary PTE ●	→	Page *N*+2:	00 01 02 03 04 05 …
				Imaginary PTE ●	→	Page *N*+3:	00 01 02 03 04 05 …
				Imaginary PTE ●	→	Page *N*+4:	00 01 02 03 04 05 …
				⋮			

This special “understanding between friends” is known as a *large page*. Instead of filling out an entire page table and referencing it from a page directory entry, we just put a shorthand entry in the page directory entry that says “Pretend there’s a page table full of contiguous pages.” This saves memory because you don’t have to spend 4KB of memory on a page table, and it saves the processor some work because it doesn’t have to walk a page table and therefore doesn’t need to spend TLB slots to remember the contents of that page table.[2]

Since a page table describes 512 × 4KB = 2MB of memory, each large page is 2MB in size.

There isn’t anything that can be done to customize this value. It’s a consequence of the way the processor is designed. So if you want to have custom sizes for large pages, you need to start by asking the processor manufacturers.

[1]In the dark ages before PAE, the x86 had a two-level page table. Windows uses PAE even on machines with less than 4GB of physical memory because NX requires PAE. If you don’t understand what that means, don’t worry.

[2] TLB slots are another one of those hidden variables in the processor. TLB stands for *translation lookaside buffer*, which is a cache that remembers mappings between linear addresses and pages.

Raymond Chen

Follow