【OS】Day24(2)

Class	Operating System: Three Easy Pieces
Date	@January 24, 2022

【Ch20】Paging: Smaller Tables(2)

20.3 Multi-level Page Tables

A different approach doesn't rely on segmentation but attacks the same problem: *how to get rid of all those invalid regions in the page table instead of keeping them all in memory?*

We call this approach a multi-level page table, as it turns the linear page table into something like a tree.

The basic idea behind a multi-level page table is simple.

First, chop up the page table into page-sized units; then, if an entire page of page-table entries(PTEs) is invalid, don't allocate that page of the page table at all.

To track whether a page of the page table is valid(and if valid, where it is in memory), use a new structure, called the page directory.

The page directory thus either can be used to tell you where a page of the page table is, or that the entire page of the page table contains no valid pages.

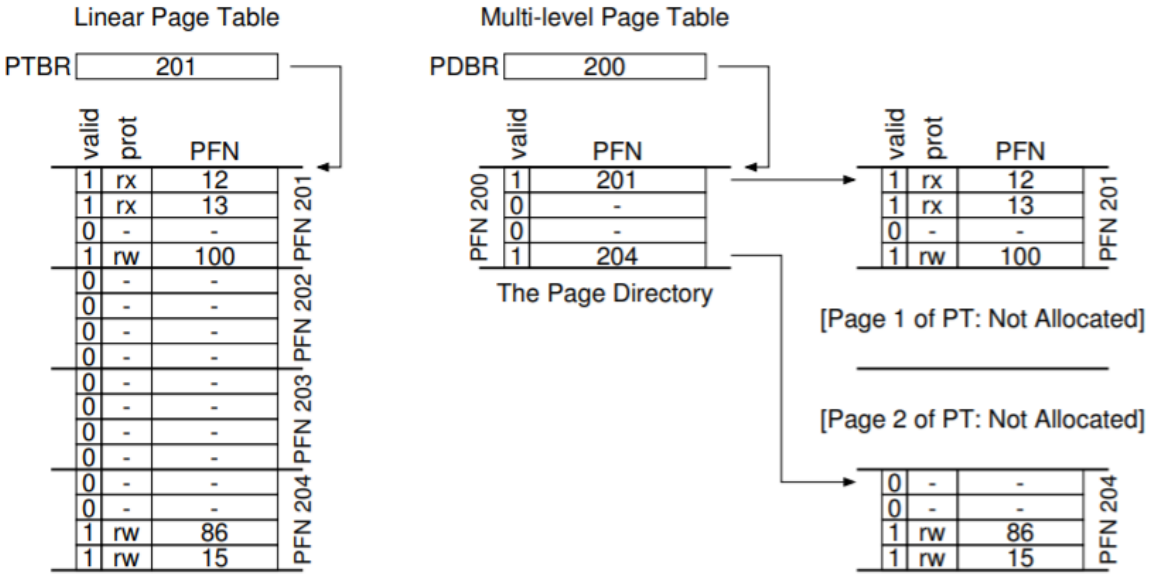

Figure 20.3: **Linear (Left) And Multi-Level (Right) Page Tables**

On the left of the figure is the classic linear page table; even though most of the middle regions of the address space are not valid, we still require page-table space allocated for those regions.

On the right is a multi-level page table. The page directory marks just two pages of the page table as valid(the first and last); thus, just those two pages of the page table reside in memory.

The page directory, in a simple two-level table, contains one entry per page of the page table. It consists of a number of page directory entries(PDE). A PDE(minimally) has a valid bit and a page frame number.

Multi-level page tables have some obvious advantages over approaches we've seen thus far:

1. First, the multi-level table only allocates page-table space in proportion to the amount of address space we are using; and thus it generally compact and supports sparse address spaces.

2. Second, if carefully constructed, each portion of the page table fits neatly within a page, making it easier to manage memory; the OS can simply grab the next free page when it needs to allocate or grow a page table.

It should be noted that there is a cost to multi-level tables; on a TLB miss, two loads from memory will be required to get the right translation information from the page table(one for the page directory, and one for the PTE itself), in contrast to just one load with a linear page table.

Thus, the multi-level table is a small example of a time-space trade-off.

Another obvious negative is complexity. Whether it is the hardware or OS handling the page-table lookup, doing so is undoubtedly more involved than a simple linear page-table lookup.