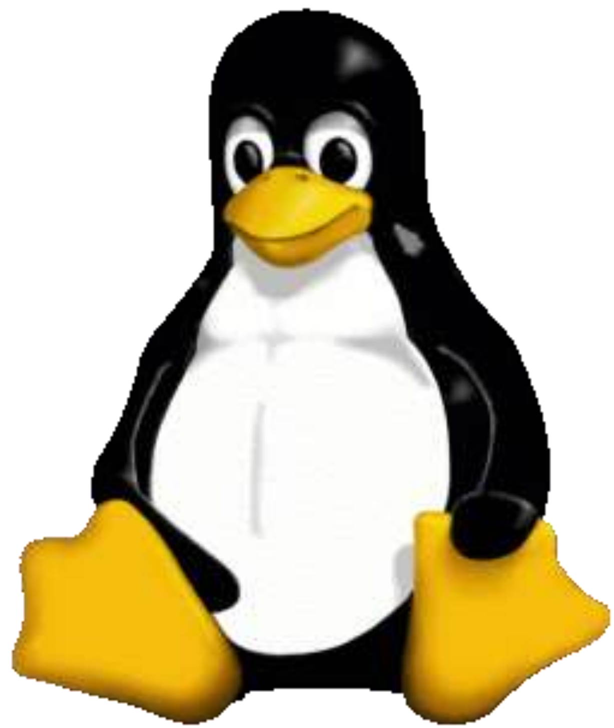
CS353 Linux Kernel

Chentao Wu吴晨涛
Associate Professor
Dept. of CSE, SJTU
wuct@cs.sjtu.edu.cn

上海交通大學

6B. Memory Management -- Methods



Chentao Wu
Associate Professor
Dept. of CSE, SJTU
wuct@cs.sjtu.edu.cn

Outline

- Page Frame Management
- Memory Area Management
- Noncontiguous Memory Area Management

Page Frame Management

- Two different page frame sizes
  - 4KB: standard memory allocation unit
  - 4MB
- Page Descriptors
  - Of type *page*
  - Stored in the *mem_map* array

The Fields of the Page Descriptor

Type	Name	Description
unsigned long	flags	Array of flags
atomic_t	_count	Page frame's reference counter
atomic_t	_mapcount	Number of Page Table entries that refer to the page frame
unsigned long	private	Available to the kernel component that is using the page
struct address_space *	mapping	Used when the page is inserted into the page cache
unsigned long	index	Used by several kernel components with different meanings
Struct list_head	lru	Contains pointers to the least recently used doubly linked list of pages.

Flags Describing the Status of a Page Frame

- PG_locked, PG_error, PG_referenced, PG_uptodate, PG_dirty, PG_lru, PG_active, PG_slab, PG_skip, PG_highmem, PG_checked, PG_arch_1, PG_reserved, PG_private, PG_writeback, PG_nosave, PG_compound, PG_swapcache, PG_mappedtodisk, PG_reclaim, PG_nosave_free

NUMA (Non-Uniform Memory Access)

- The physical memory of the system is partitioned in several nodes
  - Each node has a descriptor (Table 8-3)
  - The physical memory inside each node can be split into several zones
    - ZONE_DMA: < 16MB
    - ZONE_NORMAL: 16MB-896MB
    - ZONE_HIGHMEM: > 896MB
    - Each zone has its descriptor (Table 8-4)

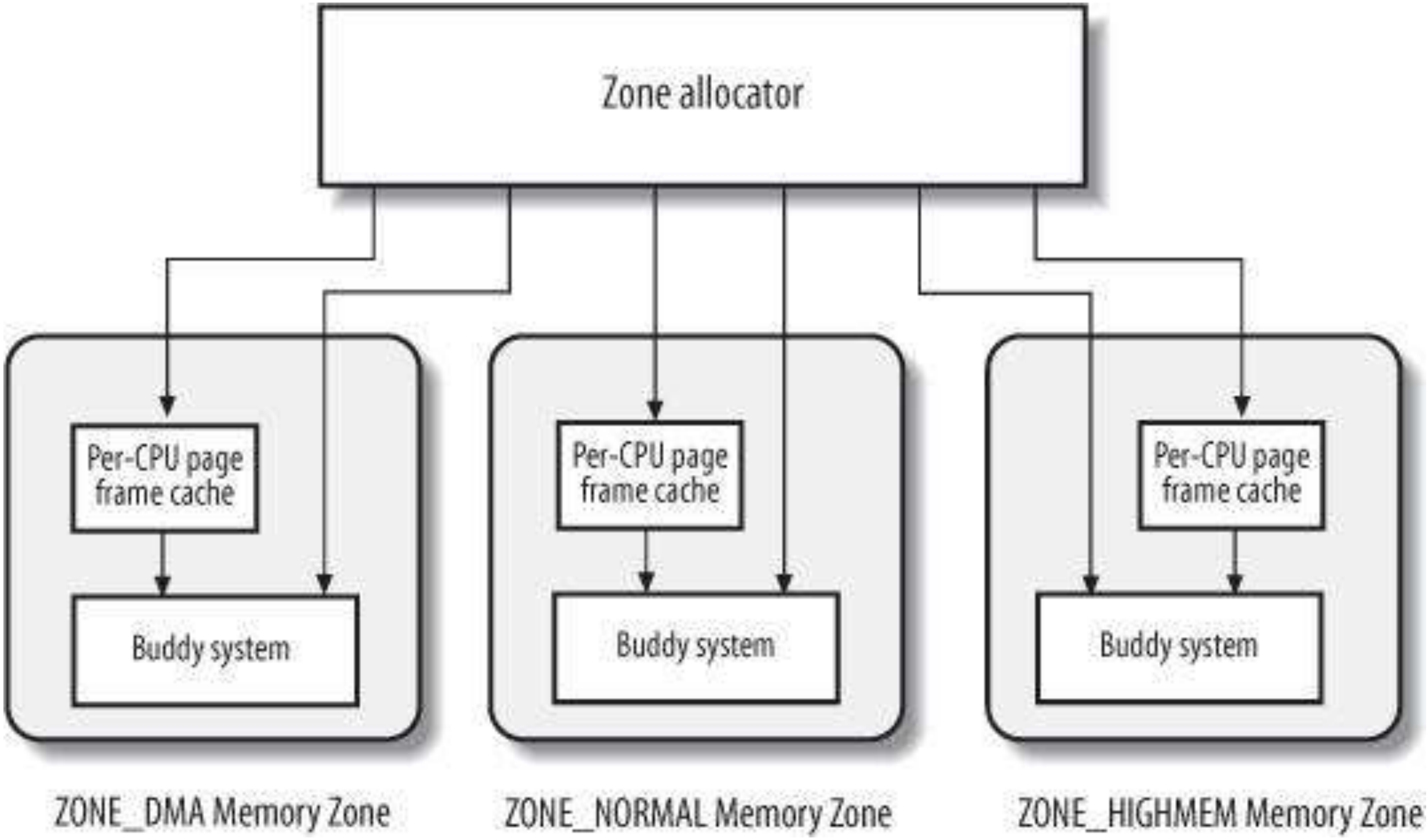
The Pool of Reserved Page Frames

- The amount of the reserved memory (in kilobytes) is stored in the *min_free_kbytes* variable

$$\text{reserved pool size} = \left\lfloor \sqrt{16 \times \text{directly mapped memory}} \right\rfloor \quad \text{(kilobytes)}$$

上海交通大學

Zoned Page Frame Allocator



Requesting and Releasing Page Frames

- Request:
  - alloc_pages(), alloc_page(), __get_free_pages(), __get_free_page(), get_zoned_page(), __get_dma_pages()
- Release:
  - __free_pages(), free_pages(), __free_page(), free_page(),

Buddy System Algorithm (1)

- To avoid external fragmentation without paging
  - Contiguous page frames are sometimes necessary
  - Advantage of leaving kernel page tables unchanged
  - Large chunks of contiguous physical memory can be accessed by the kernel through 4MB pages

上海交通大學

Buddy System Algorithm (2)

- The Buddy System
  - 11 lists of blocks: groups of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 contiguous page frames
  - 3 buddy systems in Linux 2.6
    - For DMA, normal page frames, and high-memory page frames
  - Allocating a block: __rmqueue()
  - Freeing a block: __free_pages_bulk()

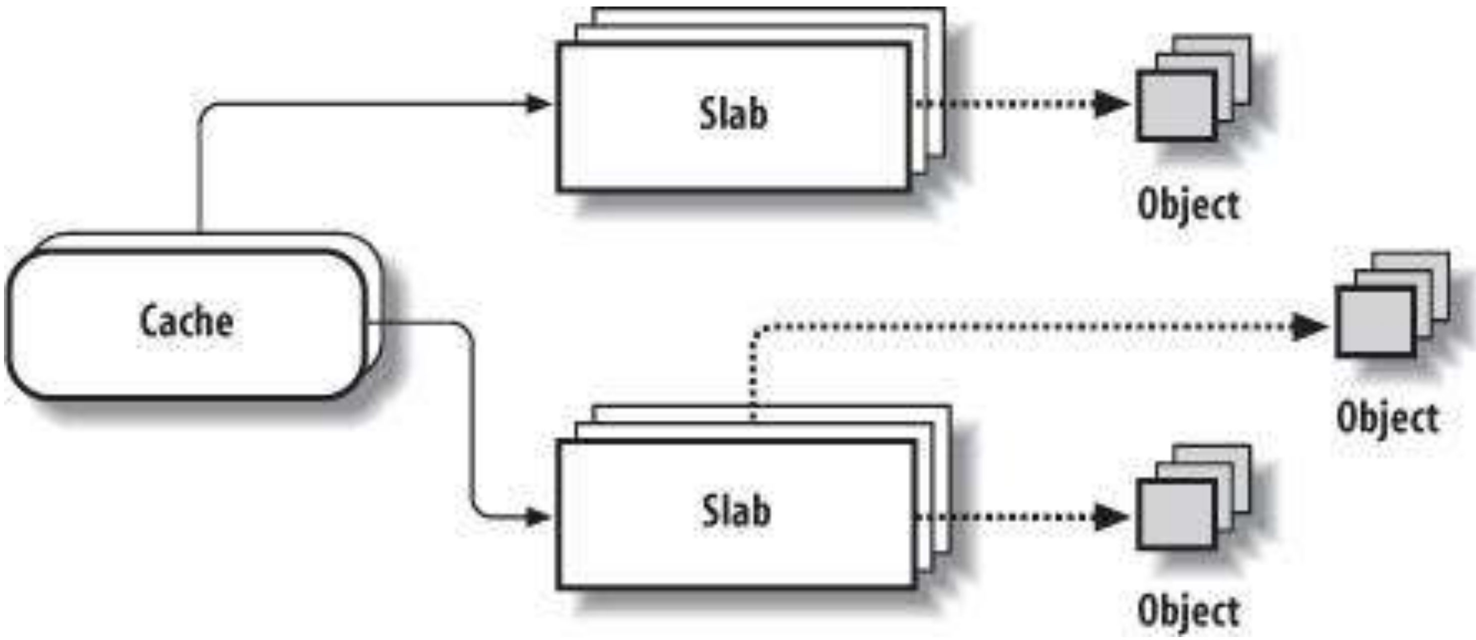
Per-CPU Page Frame Cache

- Each per-CPU cache includes some pre-allocated page frames
  - Hot cache
  - Cold cache
  - The fields of per_cpu_pages descriptor (Table 8-7)
  - Allocating page frames: buffered_rmqueue()
  - Releasing page frames: free_hot_page(), free_cold_page()

Memory Area Management

- To avoid internal fragmentation
  - Early Linux version adopt *buddy system*
    - Not efficient
  - A better algorithm is derived from *slab allocator*
    - Adopted in Solaris 2.4

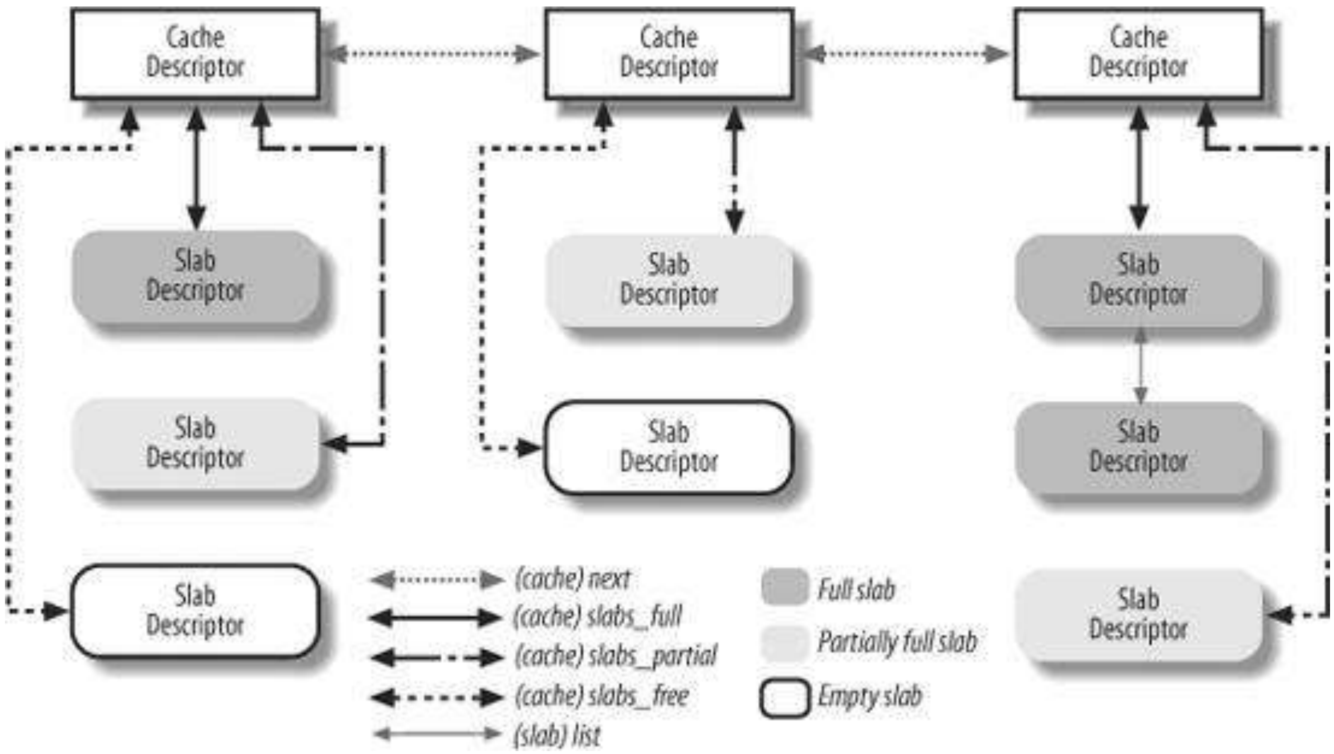
The Slab Allocator



The Slab Allocator

- The slab allocator groups objects into *caches*
  - Each cache is a store of objects of the same type
  - A *cache* is divided into *slabs*
  - Each slab consists of one or more contiguous page frames that contain both allocated and free *objects*
  - Cache descriptor of type *kmem_cache_t* (Table 8-8)
  - Slab descriptor of type *slab* (Table 8-10)
  - Object descriptor of type *kmem_bufctl_t*

Relationship between Cache and Slab Descriptors



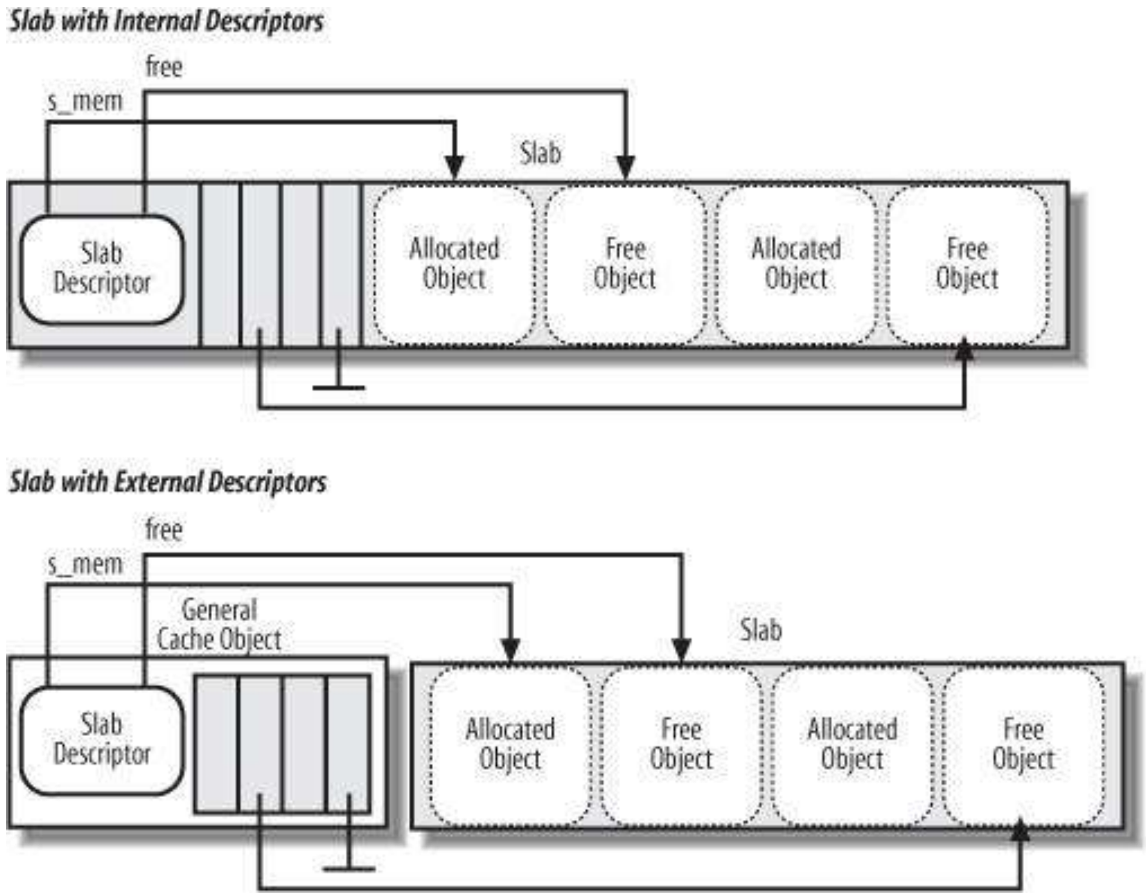
General vs. Specific Caches (1)

- General cache: kmem_cache_init()
  - kmem_cache
  - 26 caches: two caches for each of the 13 sizes
- Specific cache: kmem_cache_create()

General vs. Specific Caches (2)

- Allocating a slab to a cache
  - cache_grow()
- Releasing a slab from a cache
  - slab_destroy()

Relationships between Slab and Object Descriptors (1)



Relationships between Slab and Object Descriptors (2)

- Allocating a slab object
  - kmem_cache_alloc()
- Freeing a slab object
  - kmem_cache_free()
- General purpose objects
  - kmalloc()
  - kfree()

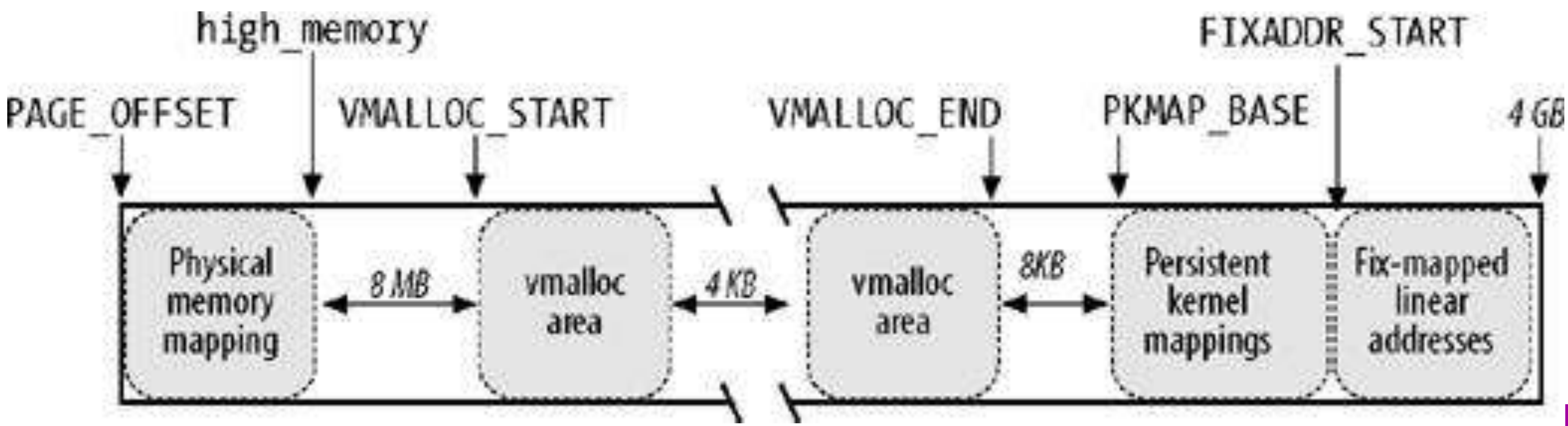
Memory Pools

- New in Linux 2.6
- Type *mempool_t*
- mempool_alloc()
- mempool_free()

Noncontiguous Memory Area Management

- To avoid external fragmentation
- Descriptor of type *vm_struct* (Table 8-13)
- get_vm_area(): look for a free range of linear address
- Allocating a noncontiguous memory area
  - vmalloc()
- Releasing a noncontiguousa memory area
  - vfree()

Linear Address Interval Starting from PAGE_OFFSET



Project 3:
Memory Management

Memory management homework

- Write a module that is called mtest
- When module loaded, module will create a proc fs entry /proc/mtest
- /proc/mtest will accept 3 kind of input
  - "listvma" will print all vma of current process in the format of

    start-addr end-addr permission

    e.g

    0x10000 0x20000 rwx

    0x30000 0x40000 r—
  - "findpage addr" will find va->pa translation of address in current process's mm context and print it. If there is not va->pa translation, prink "tranlsation not found"
  - "writeval addr val" will change an unsigned long size content in current process's virtual address into val. Note module should write to identity mapping address of addr and verify it from userspace address addr.
- All the print can be done with printk and check result with dmesg.

上海交通大學