CS5460/6460: Operating Systems

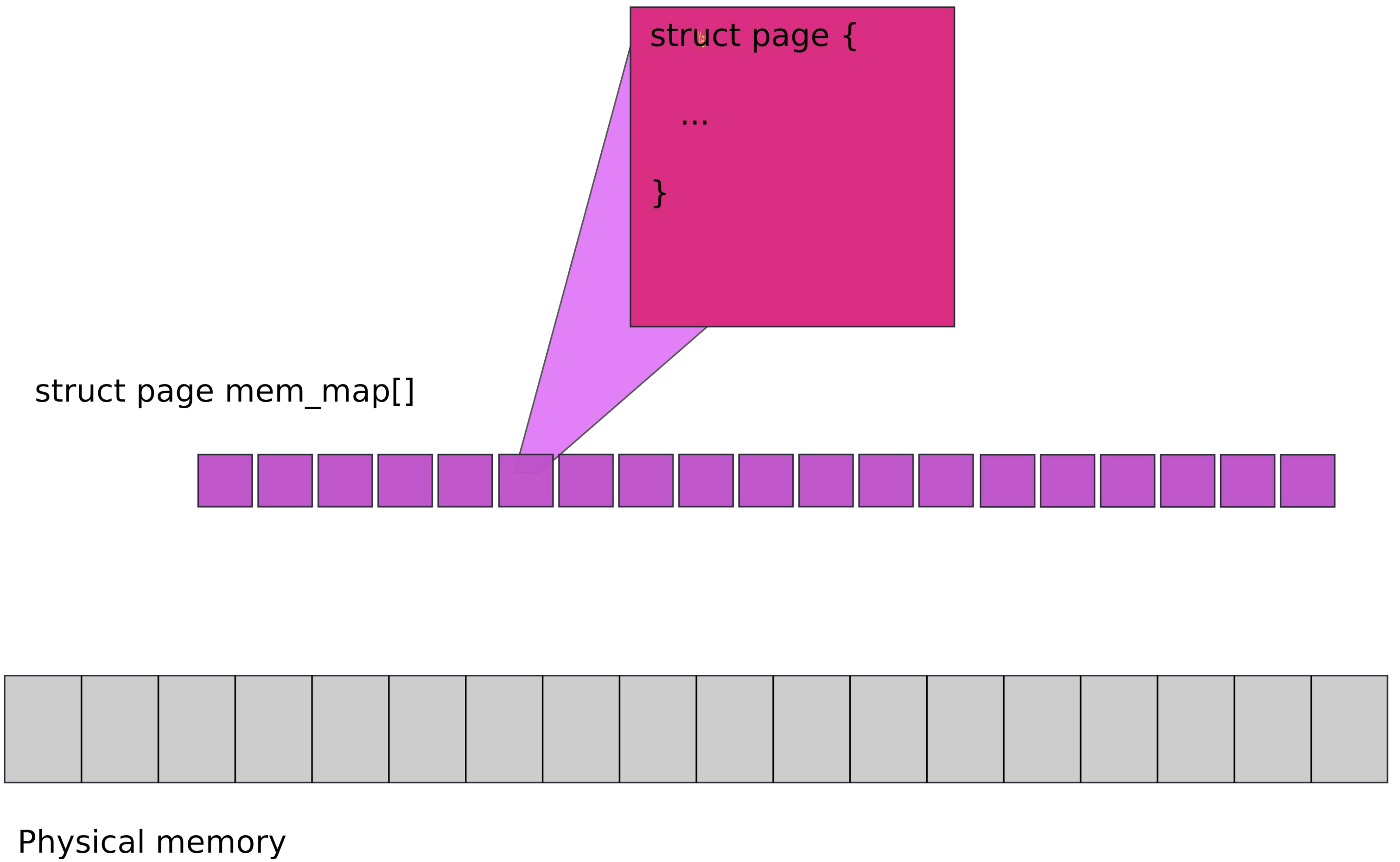
Lecture 19: Memory management

Anton Burtsev
March, 2014

Physical memory

Physical memory

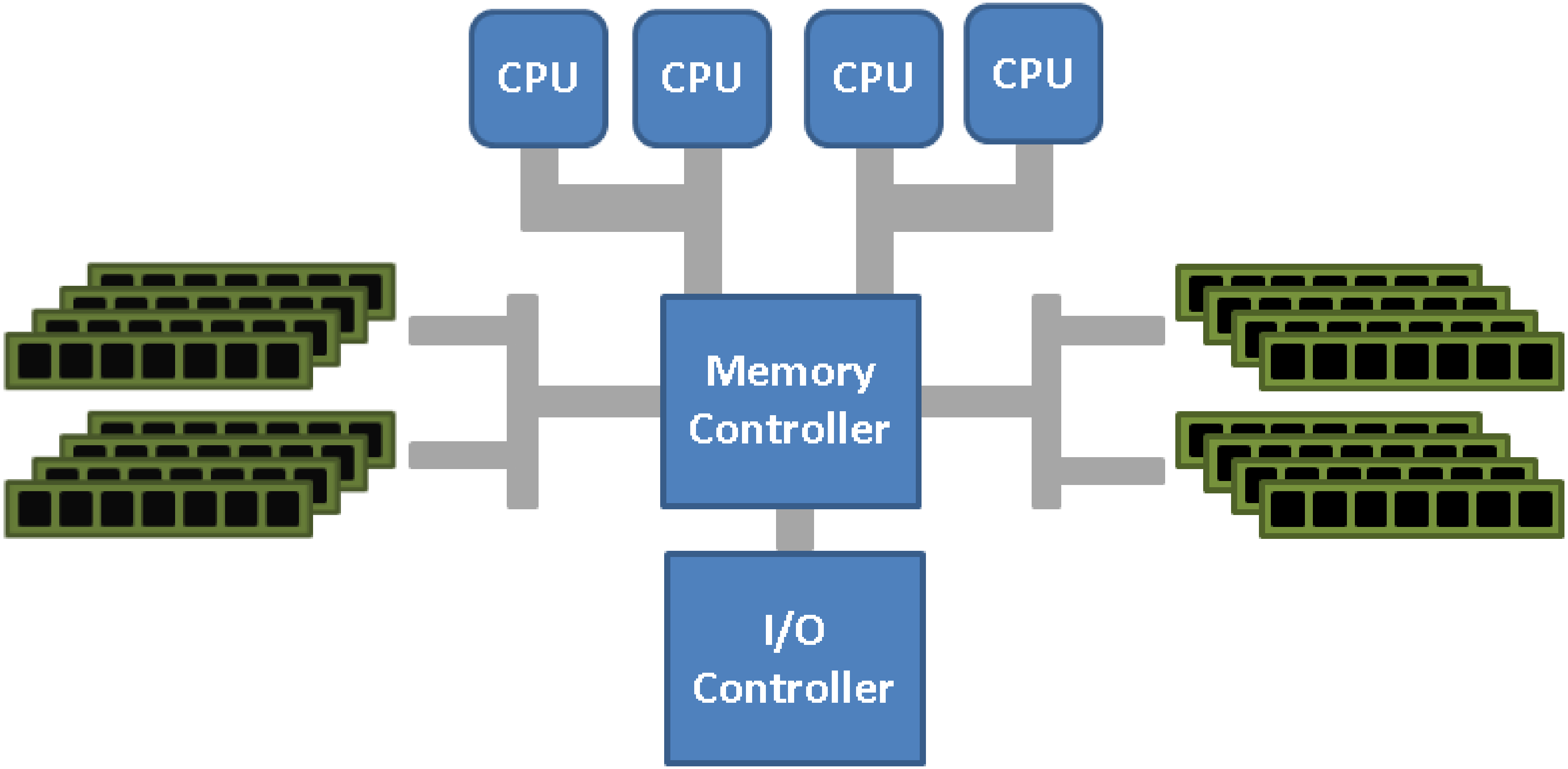
Pages (mem_map[])



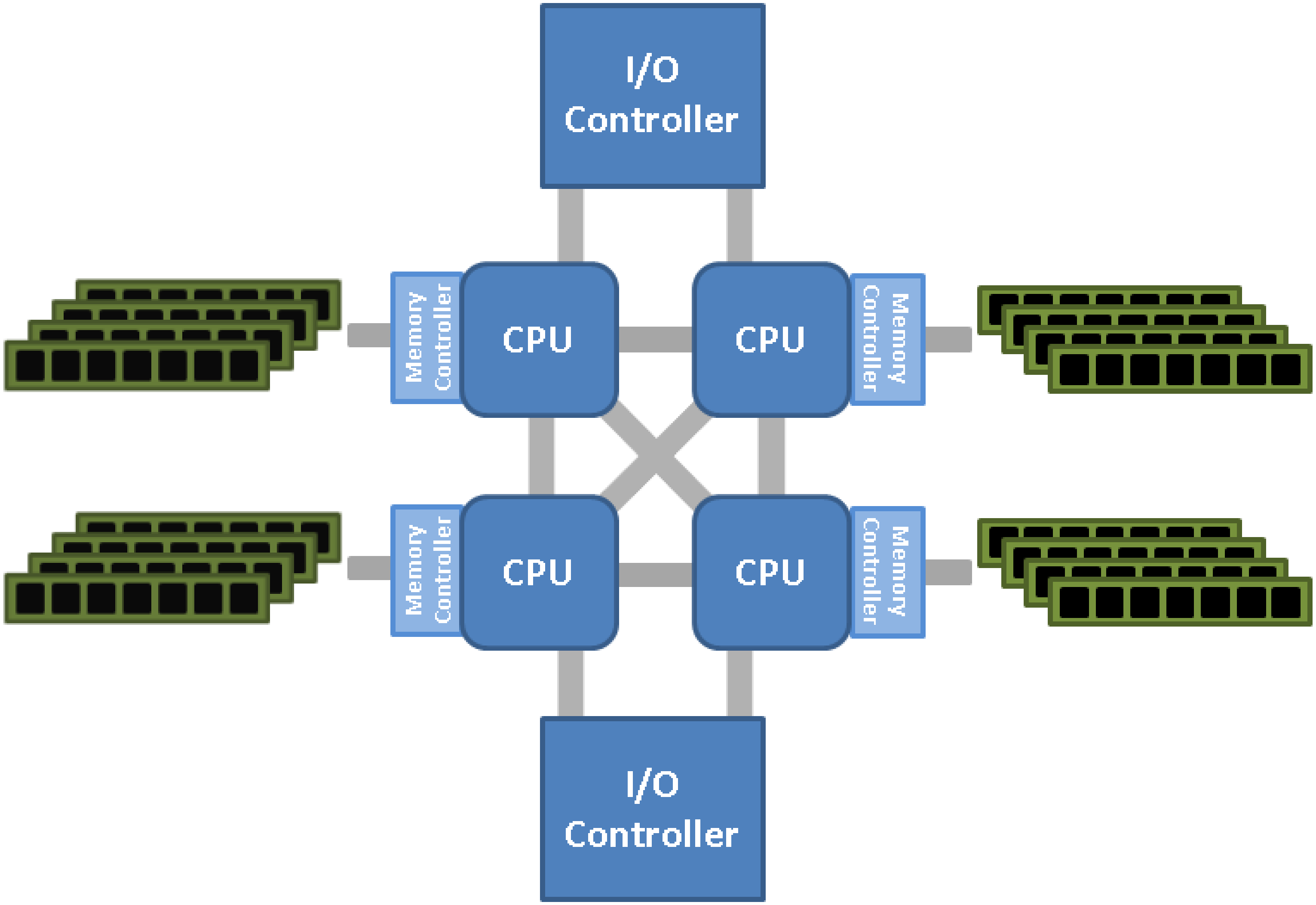
NUMA

- Parts of memory can be faster than others

Uniform memory access (UMA)

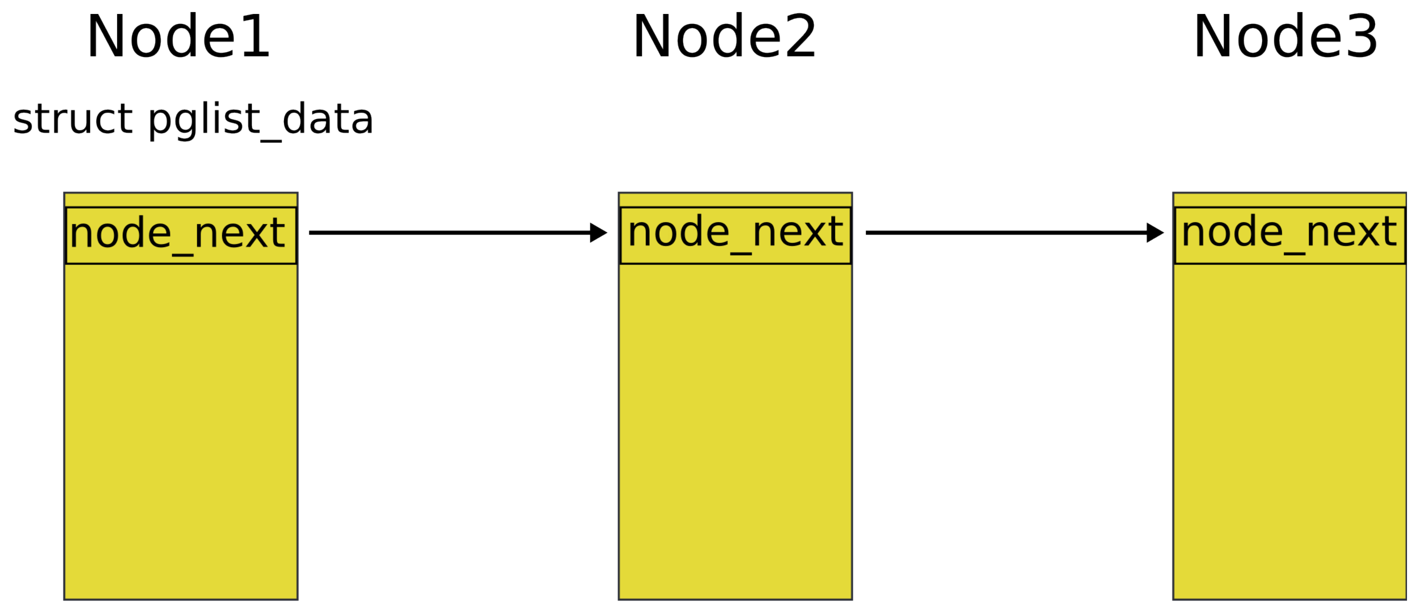


Nonuniform memory access (NUMA)

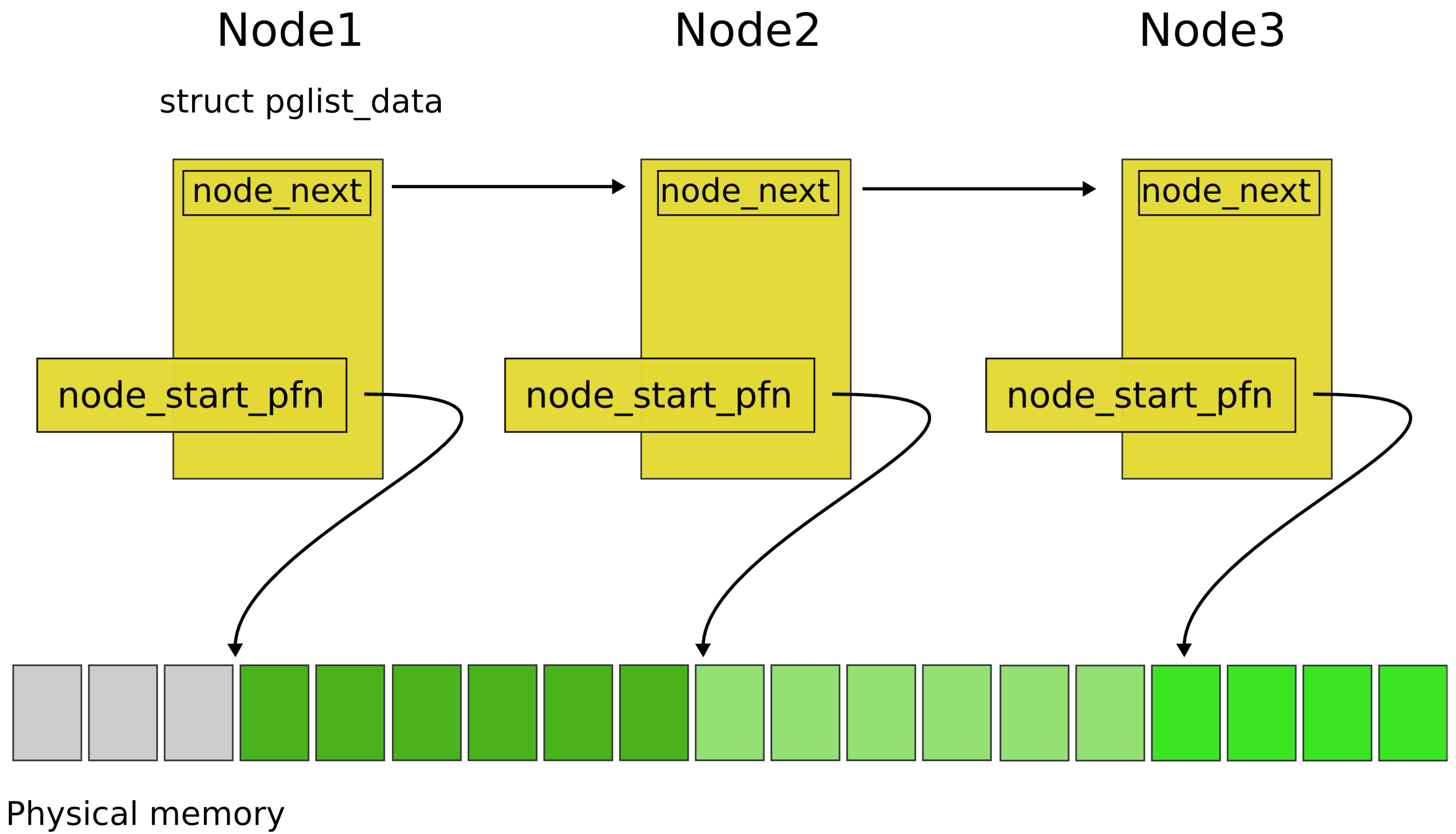


Nodes

- Attempt to allocate memory from the current node
  - Fall back to the next node in list
    - If ran out of local memory

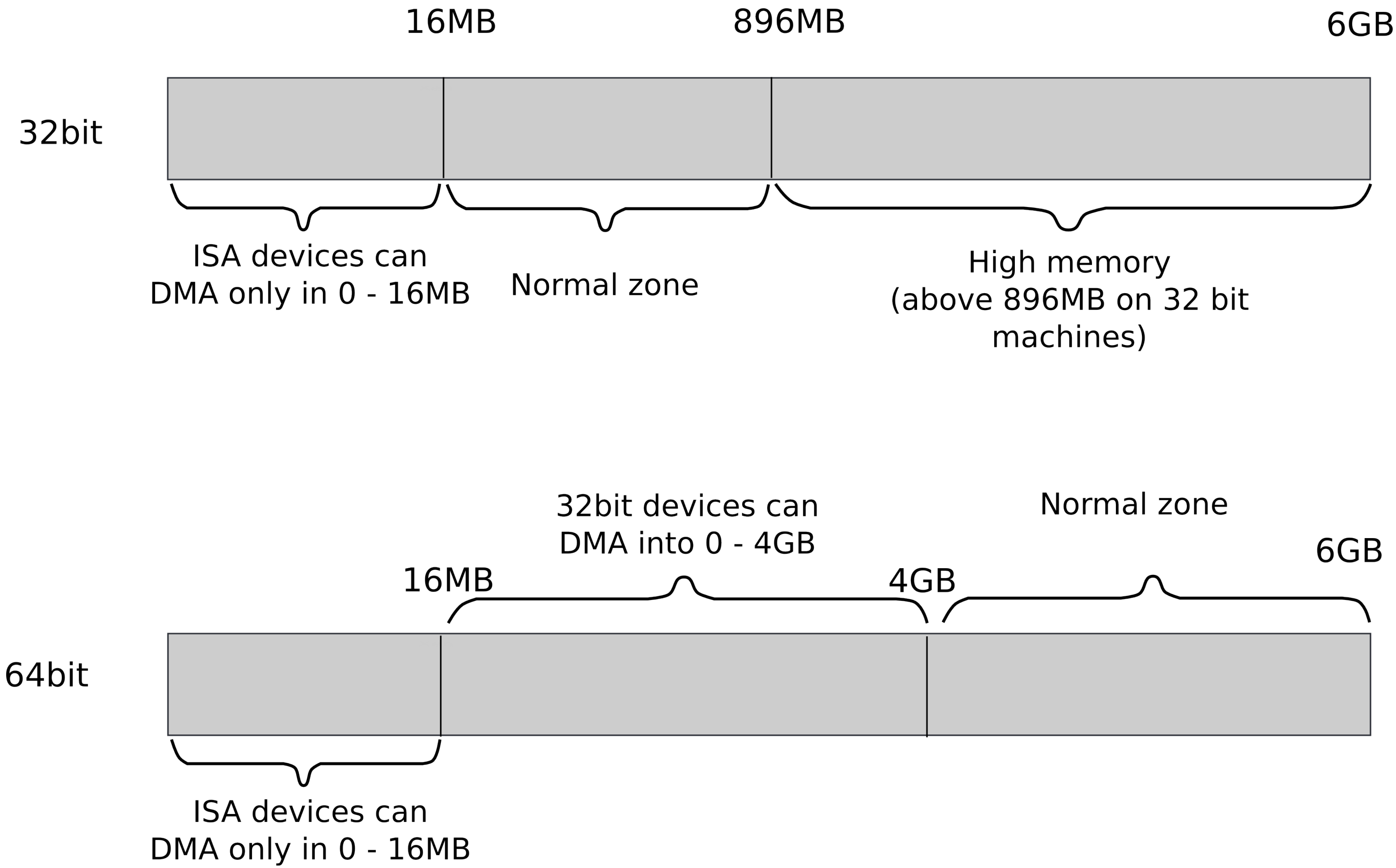


Nodes

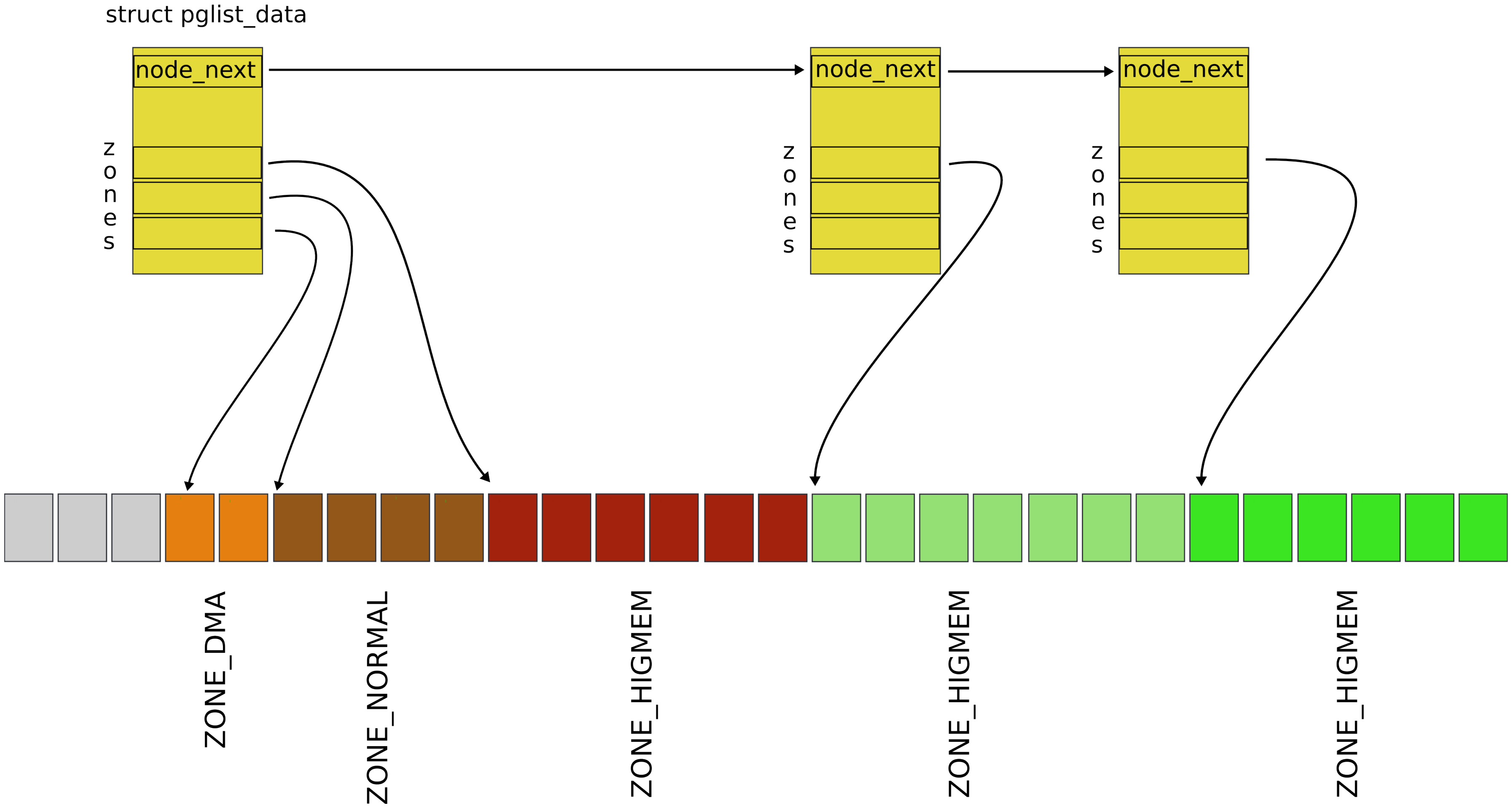


Node1
Node2
Node3
struct pglist_data
node_next
node_next
node_next
node_start_pfn
node_start_pfn
node_start_pfn
Physical memory

Zones


16MB
896MB
6GB
32bit
ISA devices can
DMA only in 0 - 16MB
Normal zone
High memory
(above 896MB on 32 bit
machines)
32bit devices can
DMA into 0 - 4GB
Normal zone
6GB
16MB
4GB
64bit
ISA devices can
DMA only in 0 - 16MB

Zones


struct pglist_data
node_next
zones
node_next
zones
node_next
zones
ZONE_DMA
ZONE_NORMAL
ZONE_HIGMEM
ZONE_HIGMEM
ZONE_HIGMEM

Memory allocation

Boot memory allocator

- Bitmap of all pages
- Allocation searches for an unused page
  - Multiple sub-page allocations can be served from the same page by advancing a pointer
- Works ok, but what is the problem?

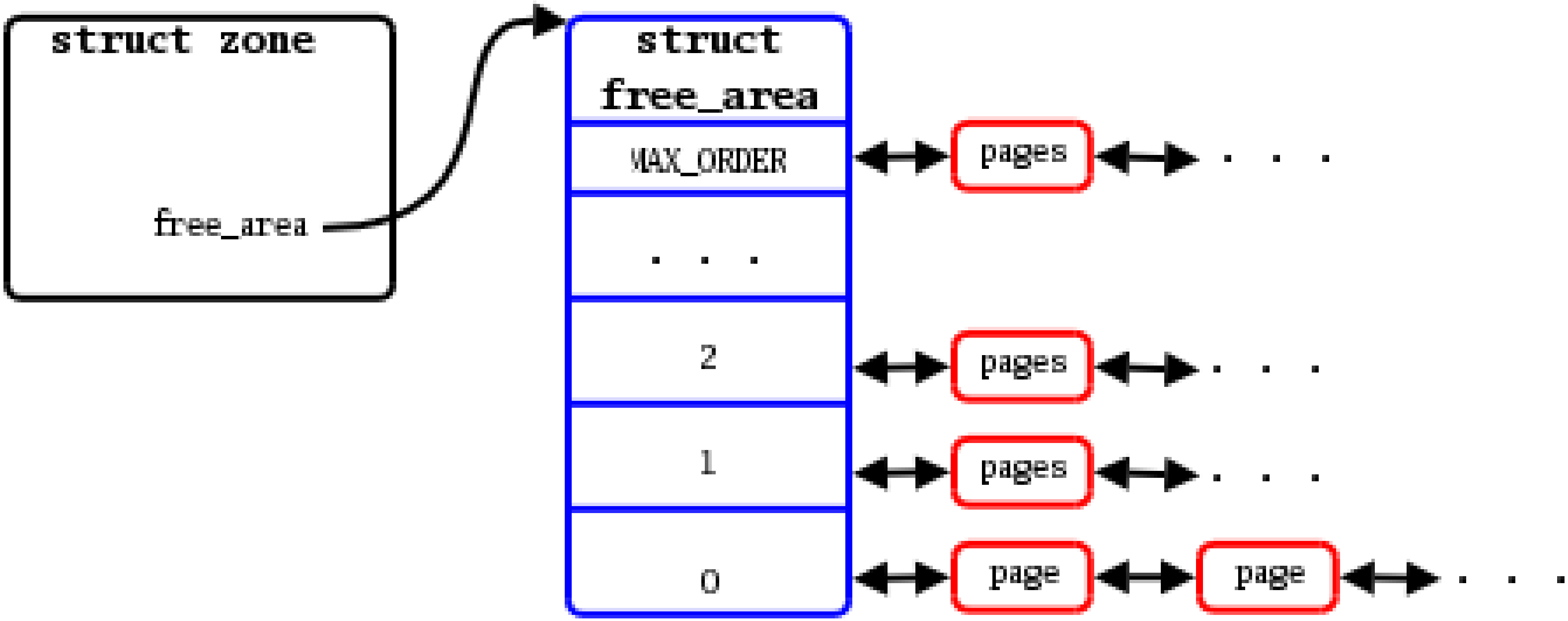
Boot memory allocator

- Bitmap of all pages
- Allocation searches for an unused page
  - Multiple sub-page allocations can be served from the same page by advancing a pointer

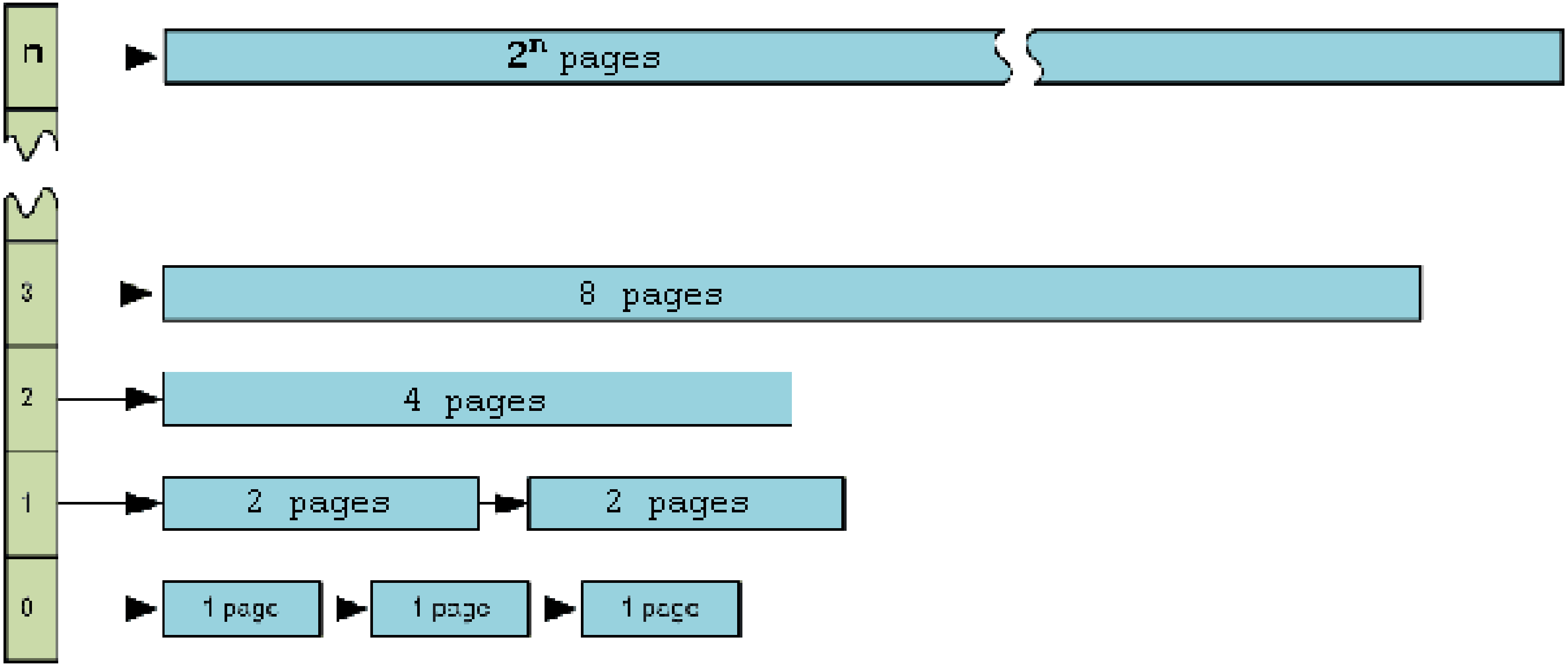
- Works ok, but what is the problem?
  - Linear scan of the bitmap
    - Too long

Buddy memory allocator

- Each zone has a buddy allocator



Buddy allocator



Per-CPU page caches

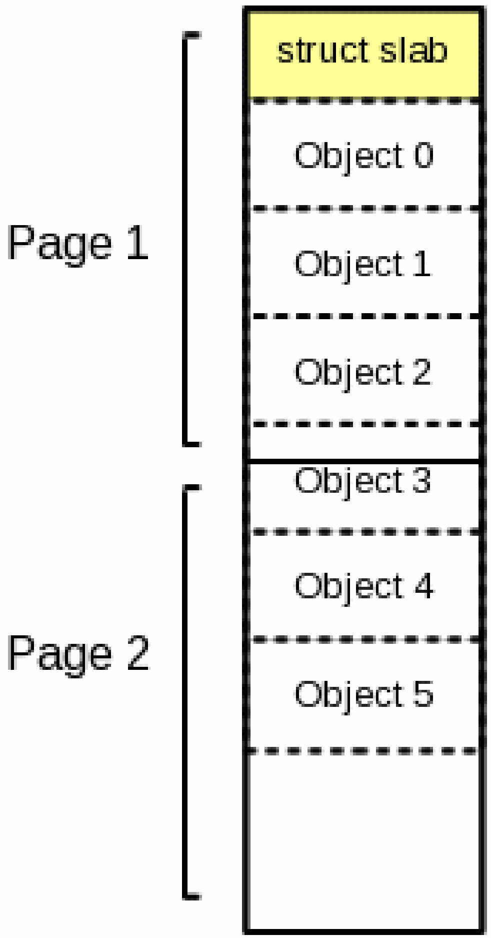
- Each memory zone defines a per-CPU page cache
  - Actually two caches:
    - Hot – pages likely accessed by CPU
    - Cold – pages used for I/O operations
- This works for serving single-page allocations

Slab allocator

- Buddy allocator is ok for large allocations
  - E.g. 1 page or more
- But what about small allocations?
  - Buddy uses the whole page for a 4 bytes allocation
    - Wasteful
  - Buddy is still slow for short-lived objects

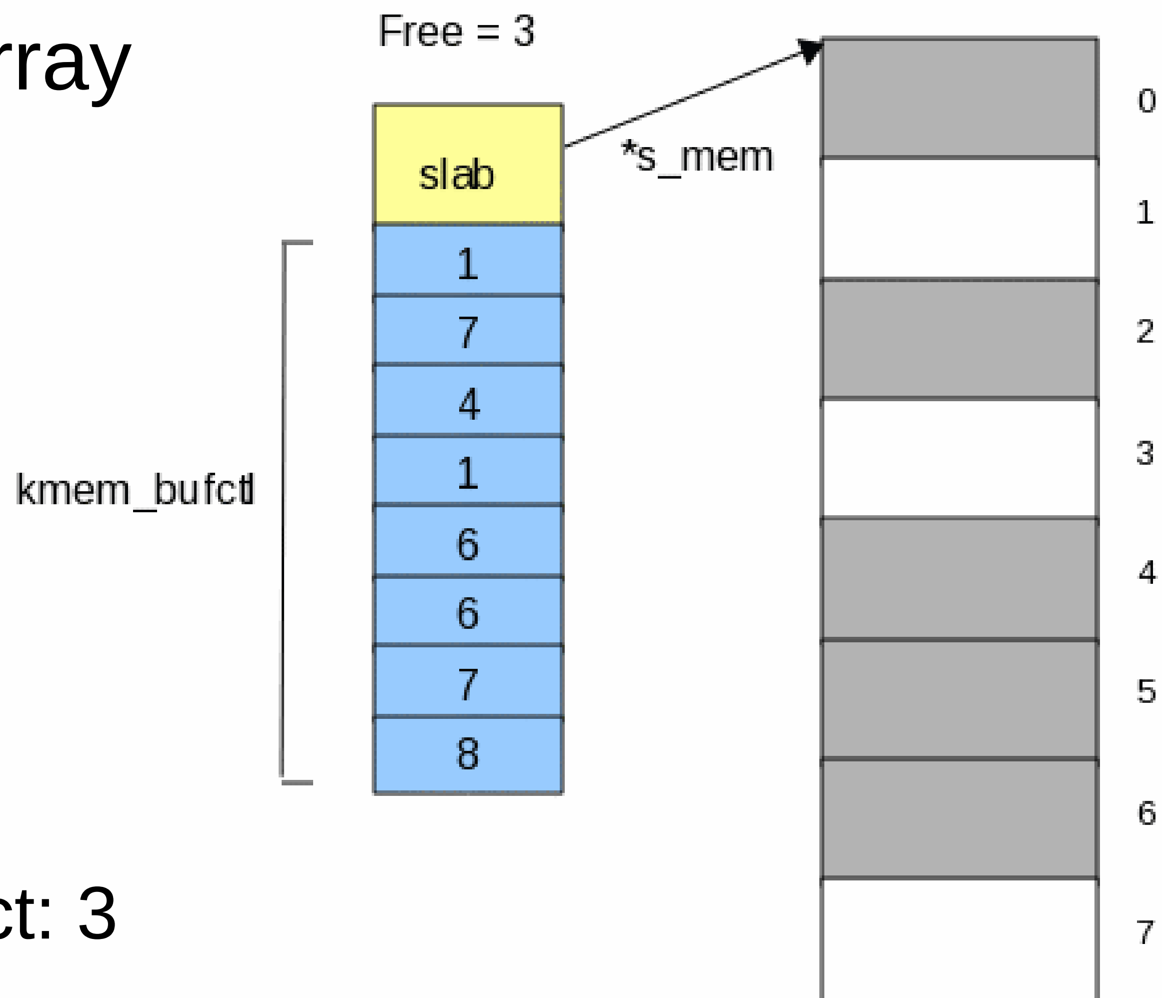
Slab

- A 2 page slab with 6 objects



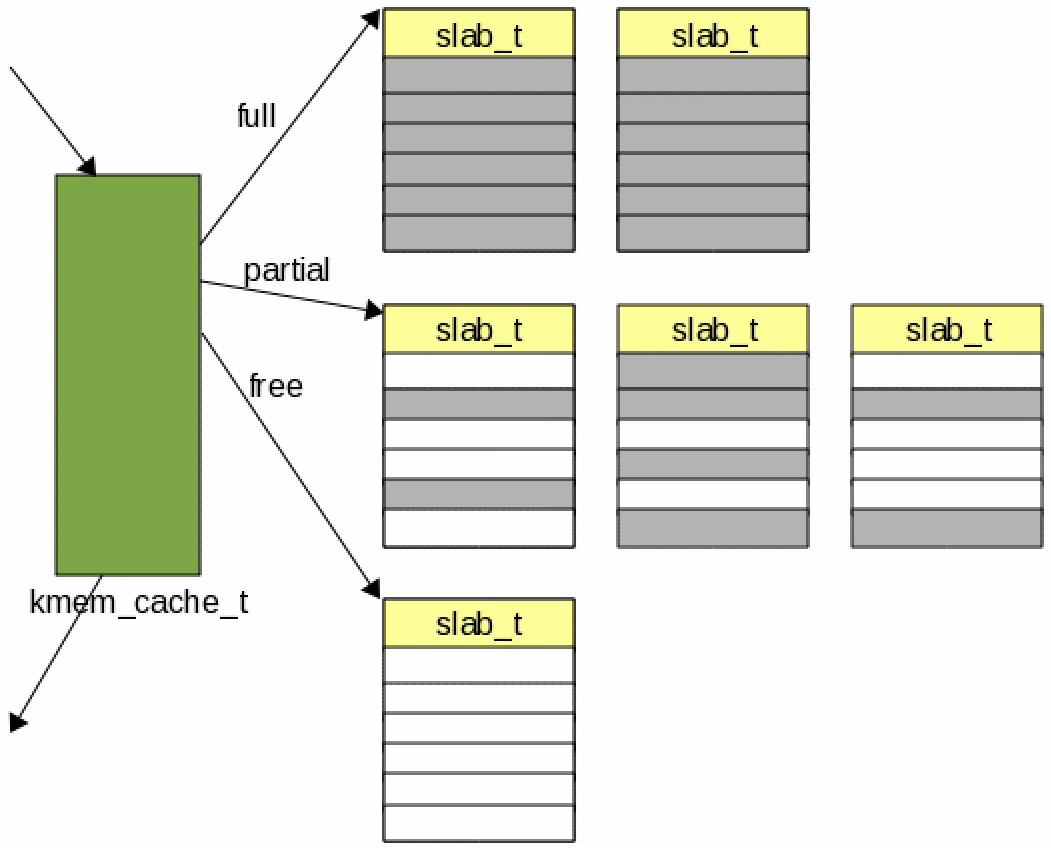
Keeping track of free objects

- kmem_bufctl array is effectively a linked list



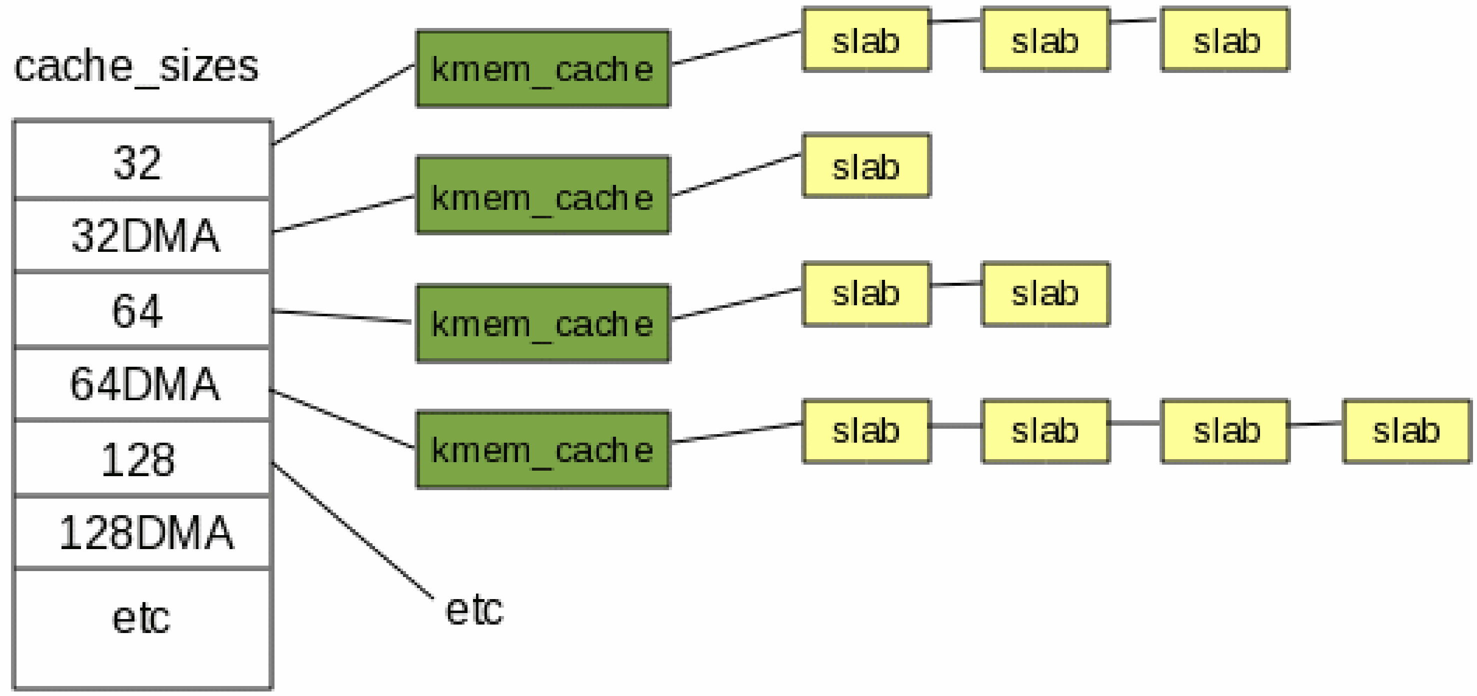
- First free object: 3
- Next free object: 1

A cache is formed out of slabs

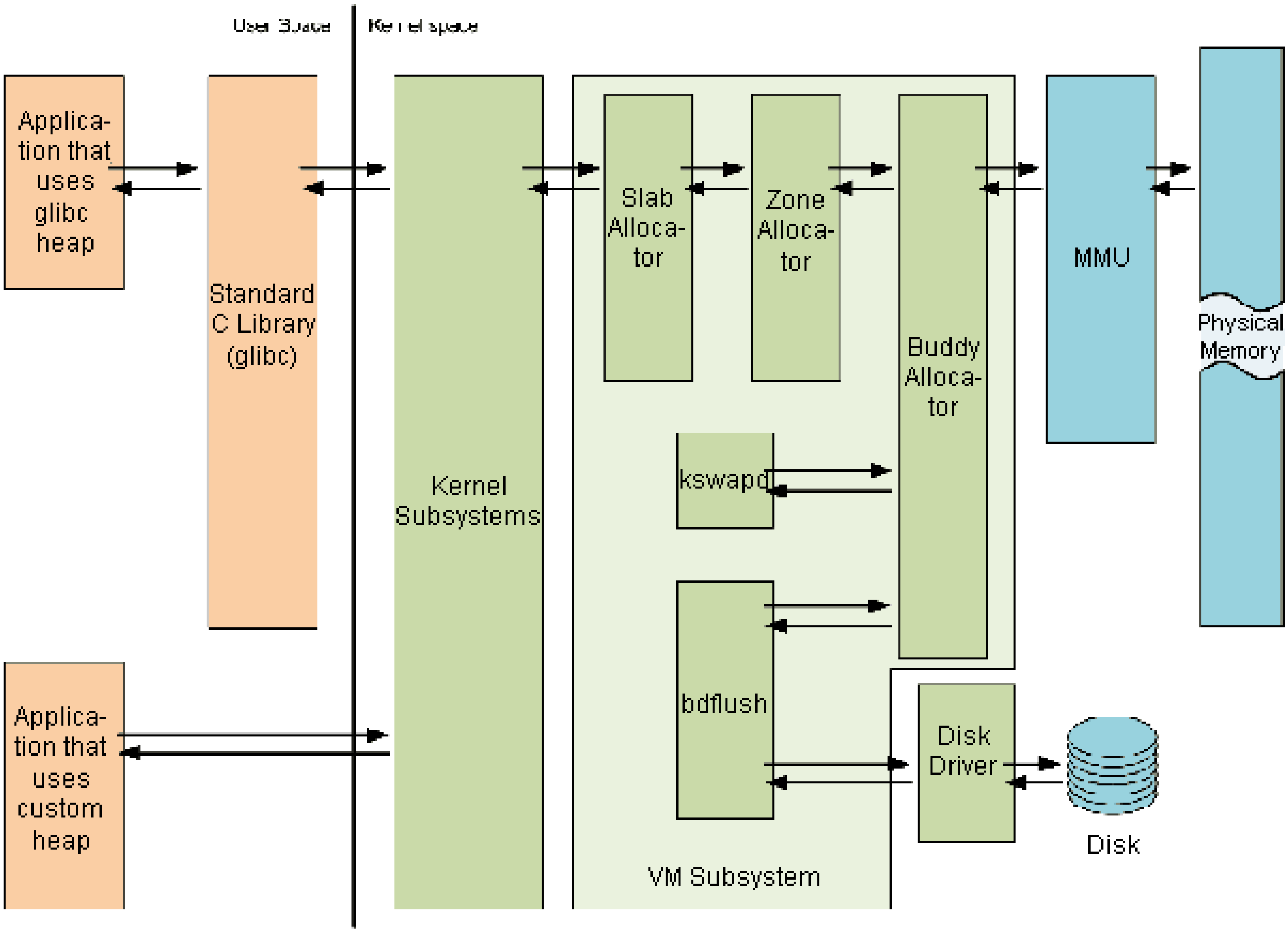


Kmalloc(): variable size objects

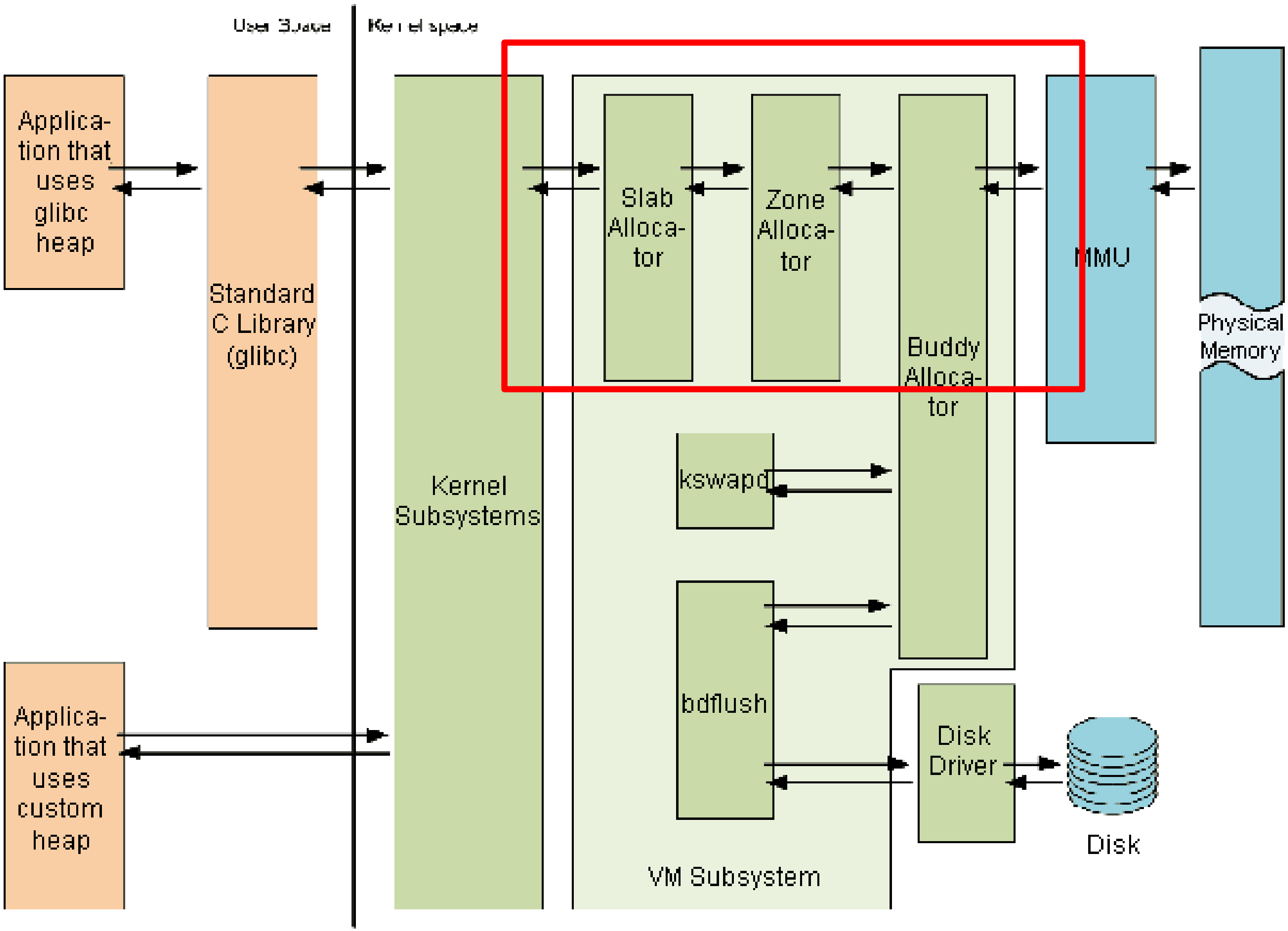
- A table of caches
  - Size: 32, 64, 128, etc.



Linux memory management



Linux memory management



Thank you!