How does
the SLUB allocator work

Joonsoo Kim
LGE CTO SWP Lab.
iamjoonsoo.kim@lge.com
js1304@gmail.com

Contents

- Memory allocation hierarchy
- Implementation – the SLUB
- Difference between the SLAB and the SLUB
- Current status

MEMORY ALLOCATION HIERARCHY

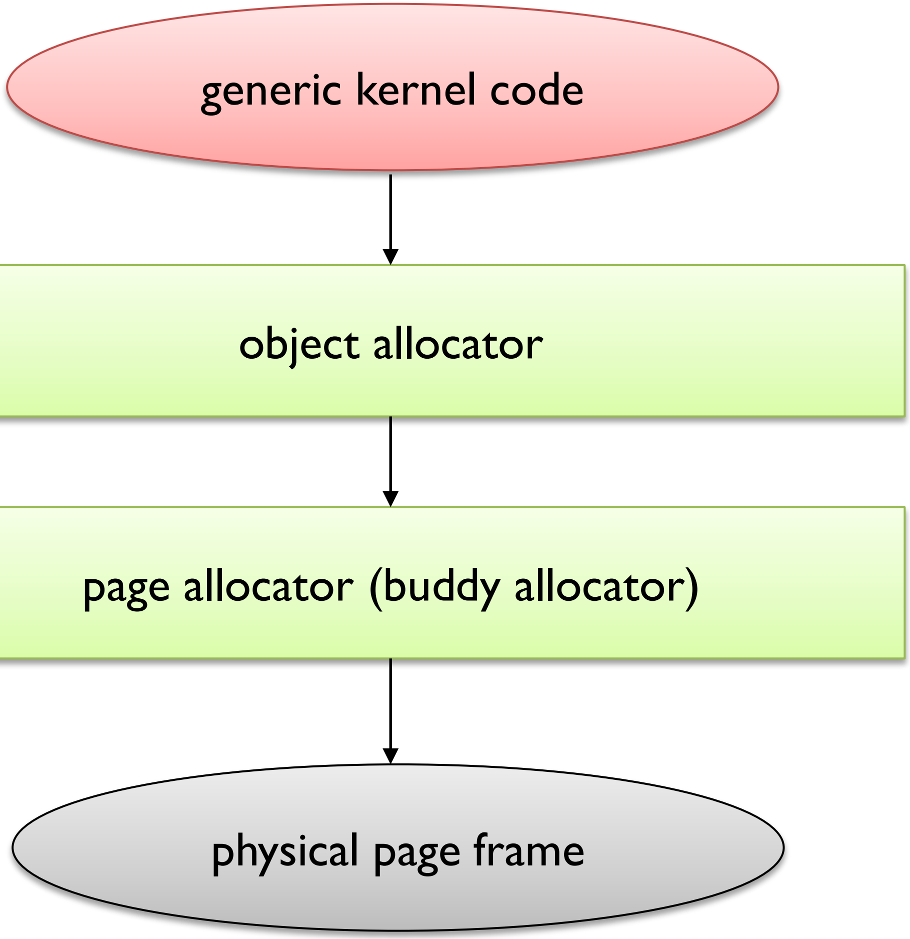
Page allocator

- page allocator
  - fundamental memory allocator
  - manage all physical memory in system
  - page size = 4096 bytes
  - allocate 2^{order} pages at once
- limit
  - size less than page size

What is the SLAB allocator?

- The SLAB allocator
  - in-kernel library like in-userspace library malloc()
  - kmalloc() = malloc()
  - kmem_cache_create(), kmem_cache_alloc(), …
- The object allocator ➔ providing same API
  - The SLAB allocator: traditional allocator
  - The SLOB allocator: for embedded system
  - The SLUB allocator: default, for large system

Allocation hierarchy



generic kernel code

object allocator

page allocator (buddy allocator)

physical page frame

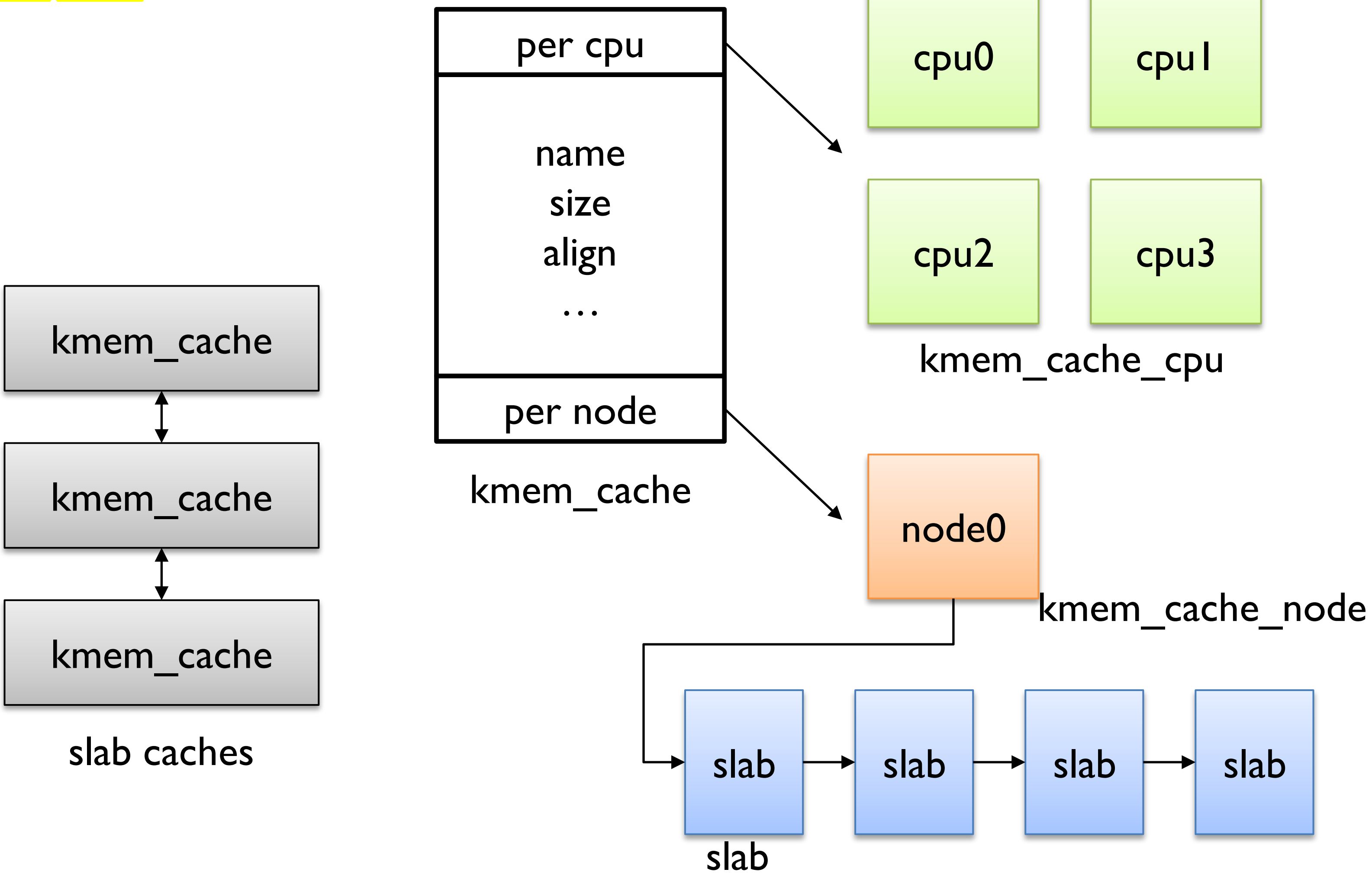
Warning: term

- “the SLAB allocator” vs “the slab”
  - the SLAB allocator
    - one of the object allocator
  - the slab
    - just data structure
    - used by the slab allocator and the slub allocator

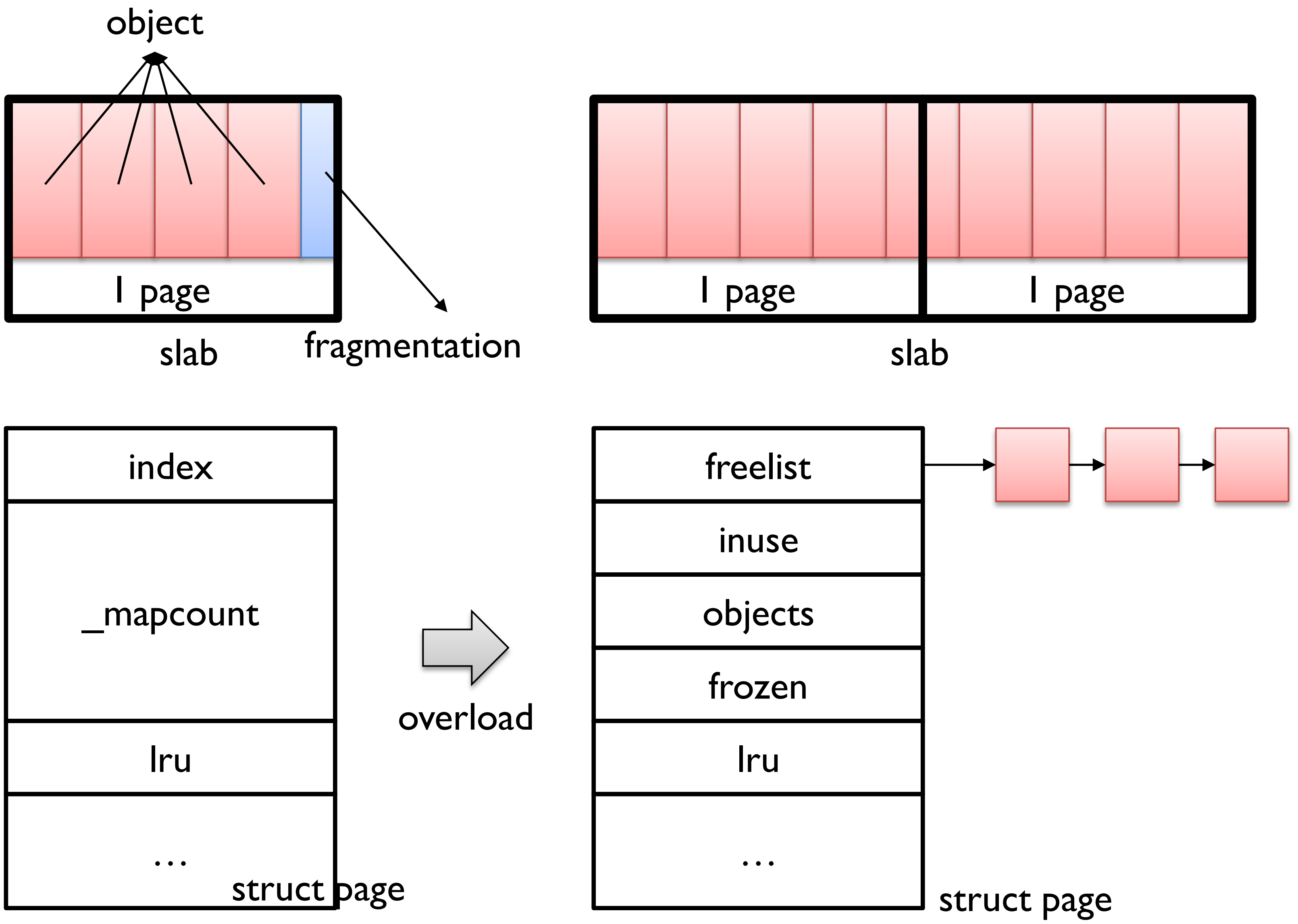
IMPLEMENTATION
- SLUB

Overall structure

UMA, SMP



Slab



object

1 page

slab

fragmentation

1 page

1 page

slab

index

_mapcount

lru

…

struct page

overload

freelist

inuse

objects

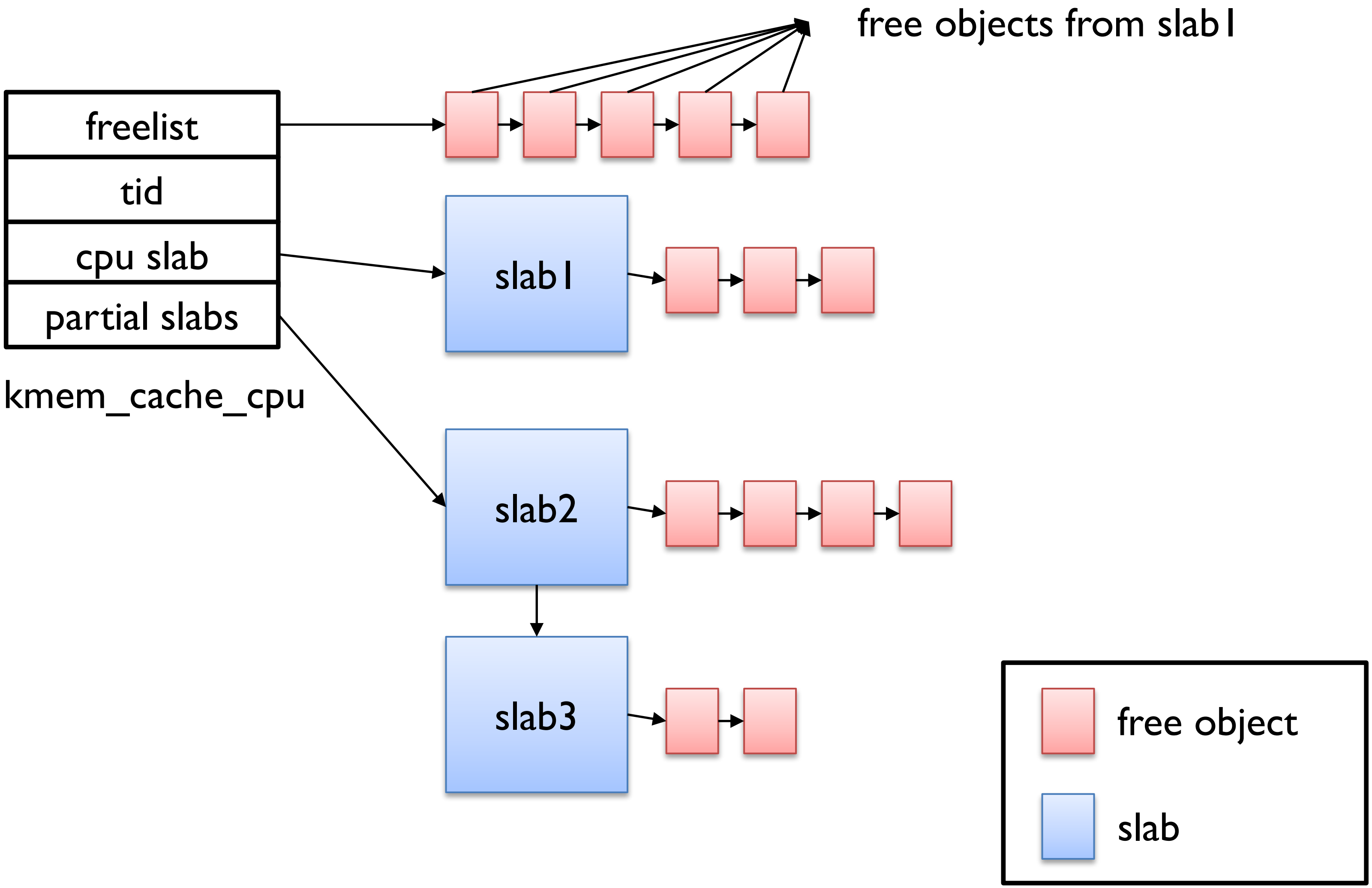
frozen

lru

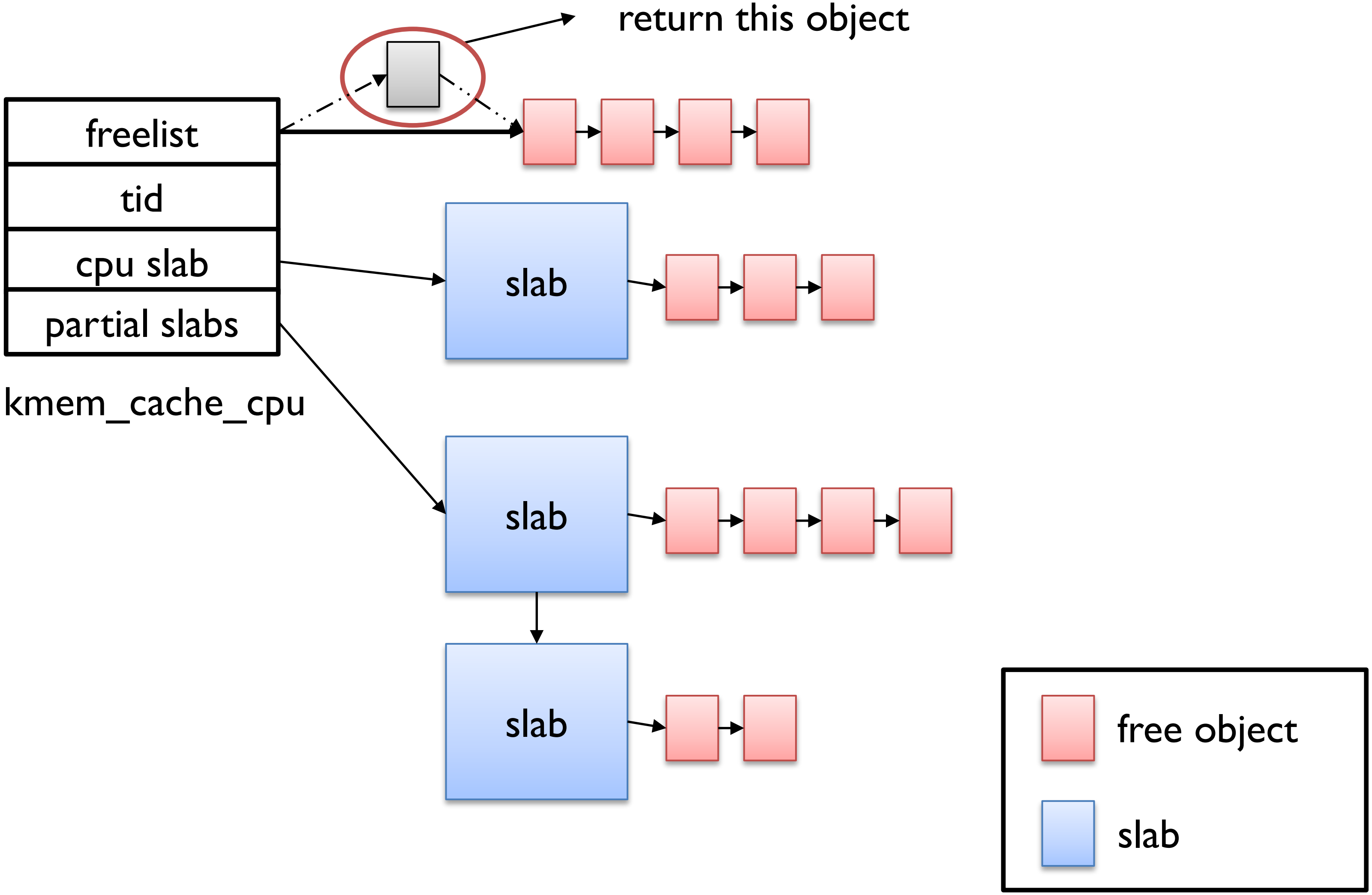
…

struct page

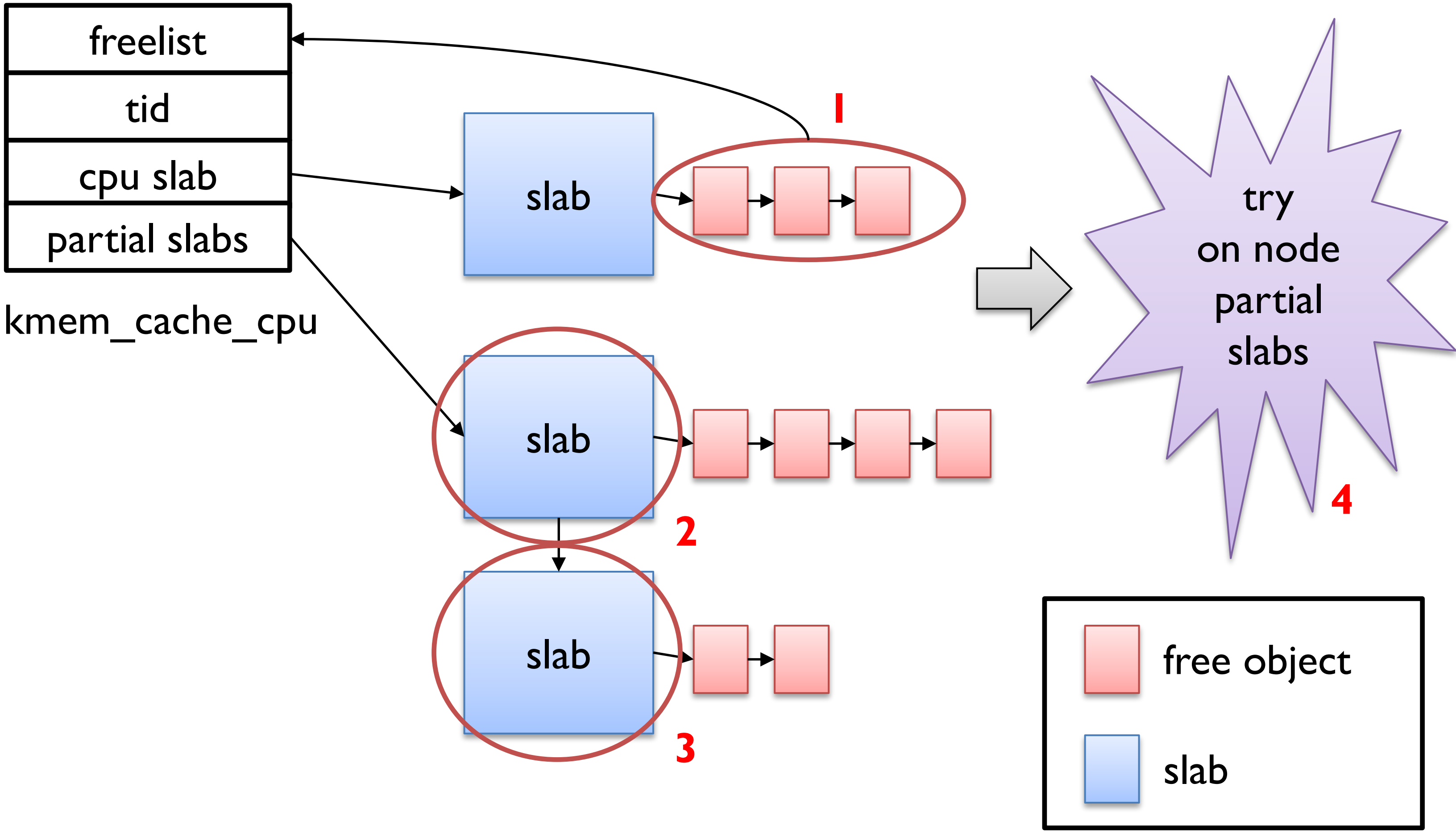
Per cpu structure



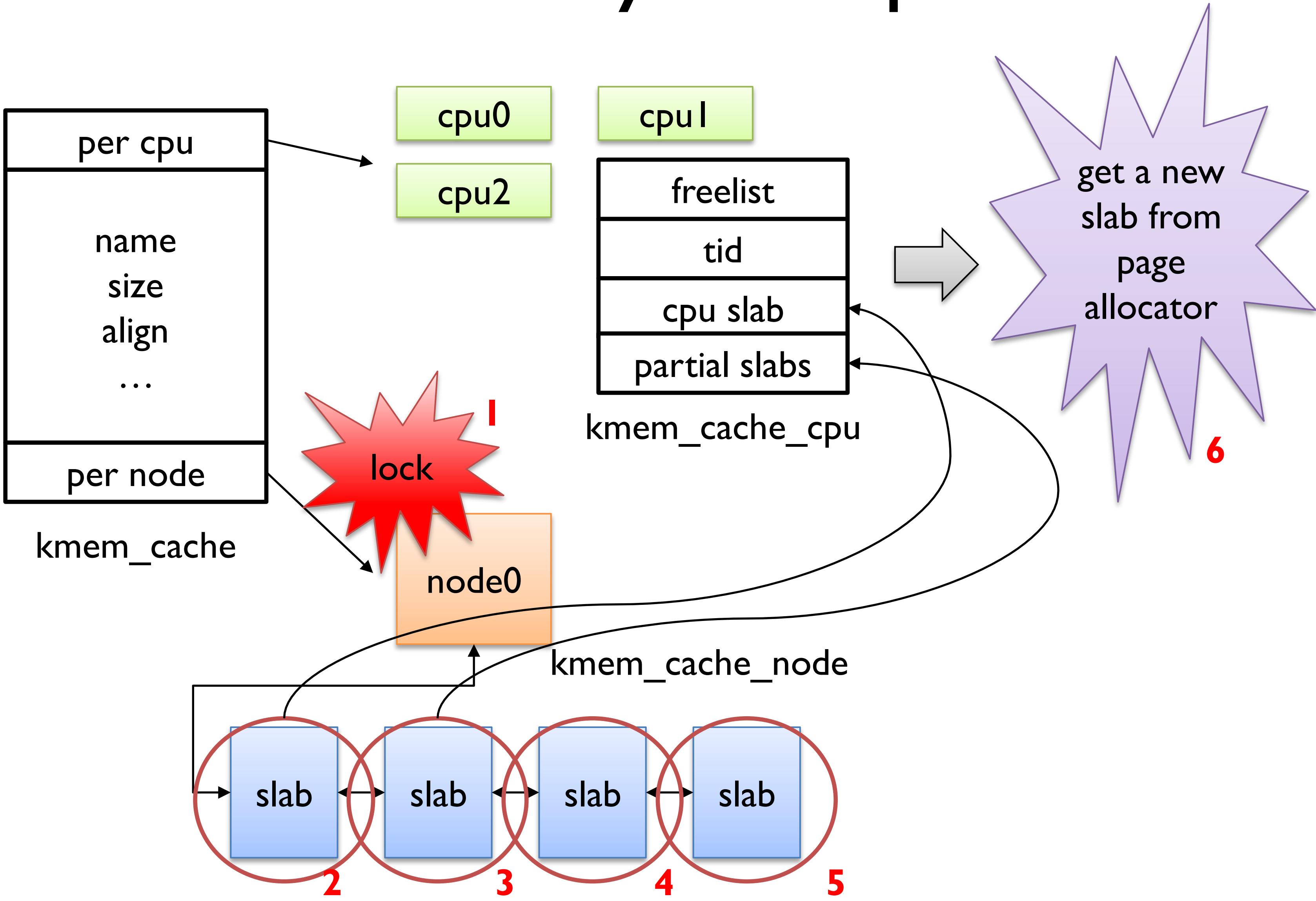
Allocation: fast-path



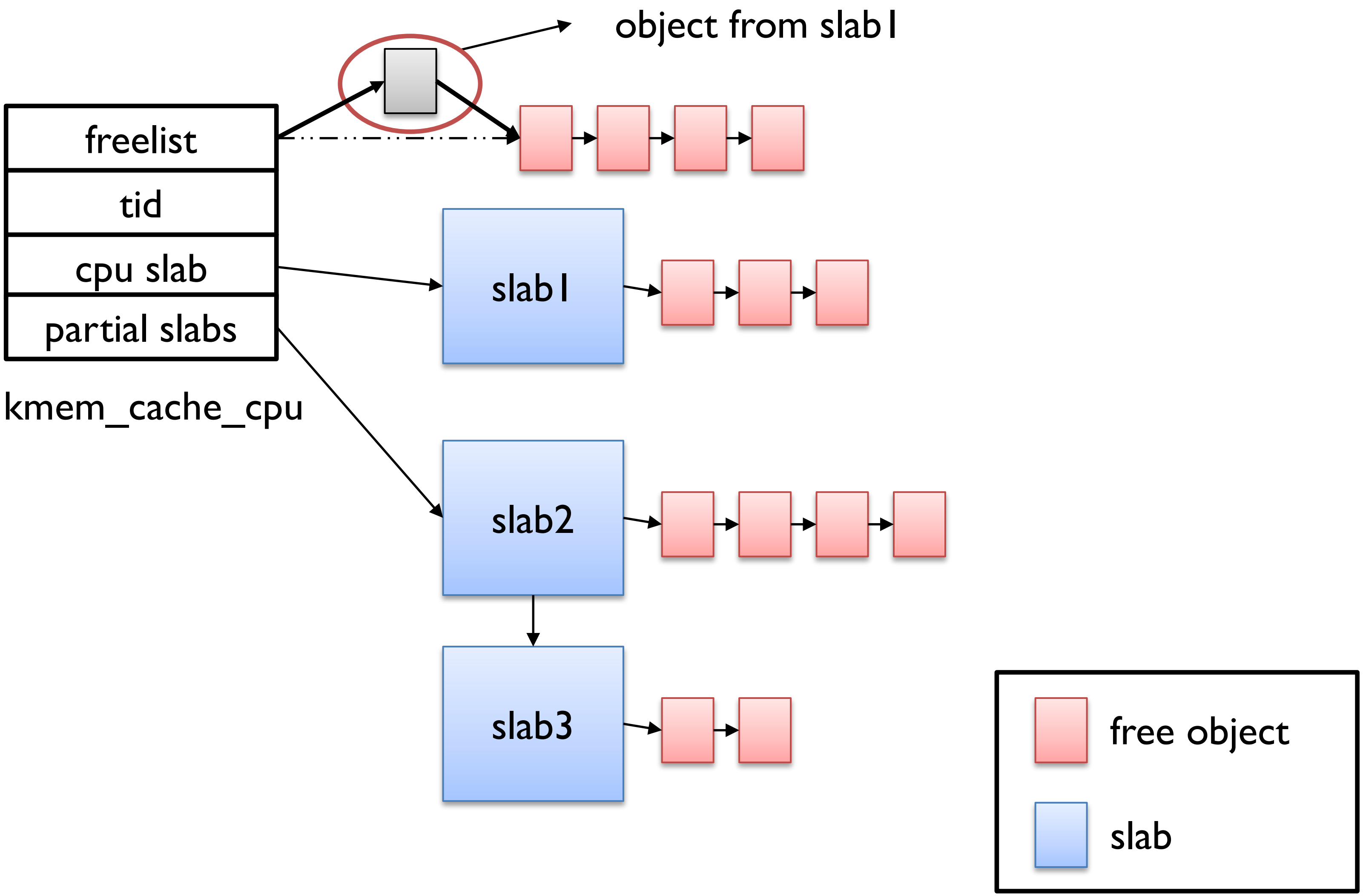
Allocation: slow-path



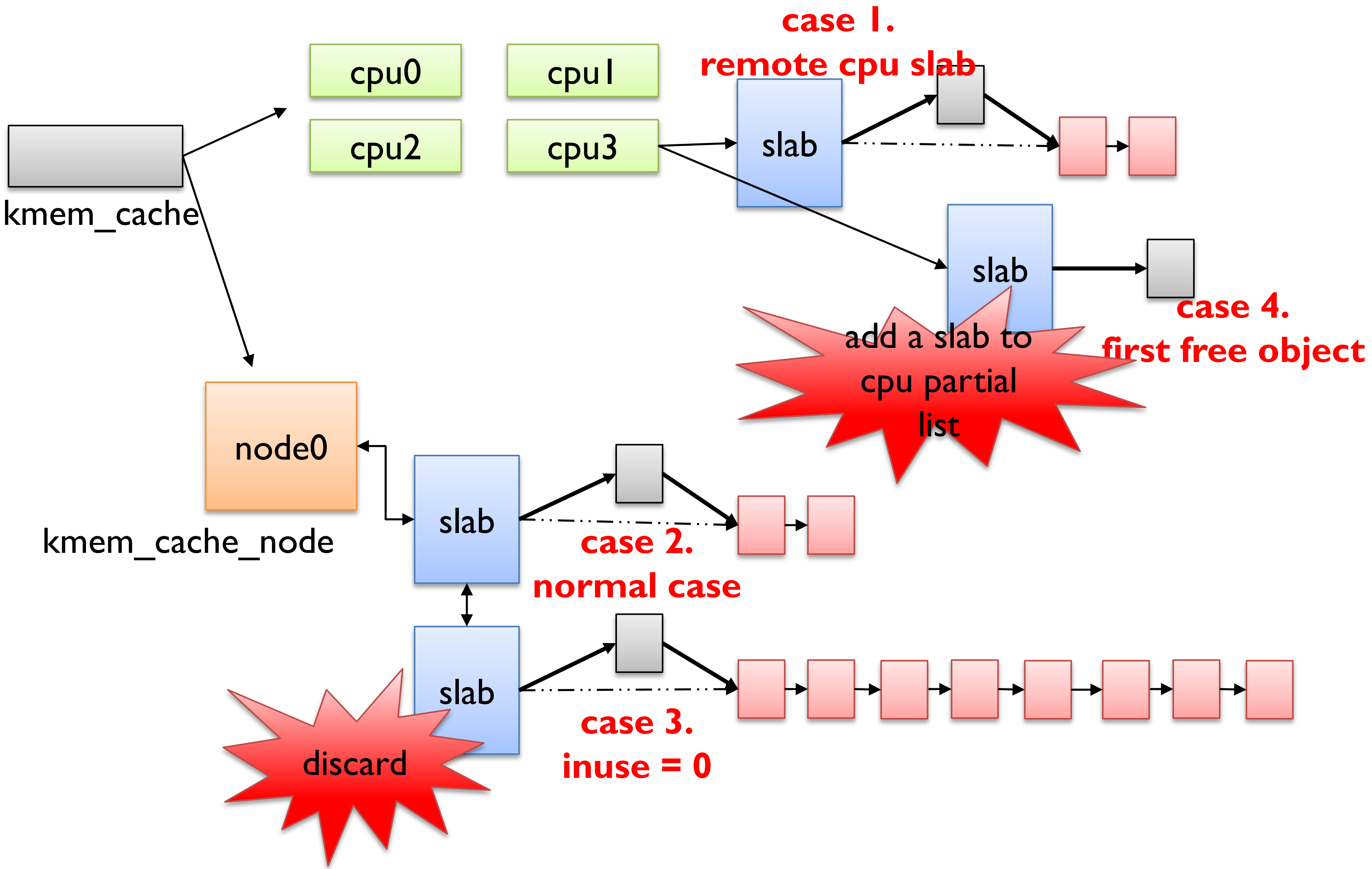
Allocation: very slow-path



Free: fast-path



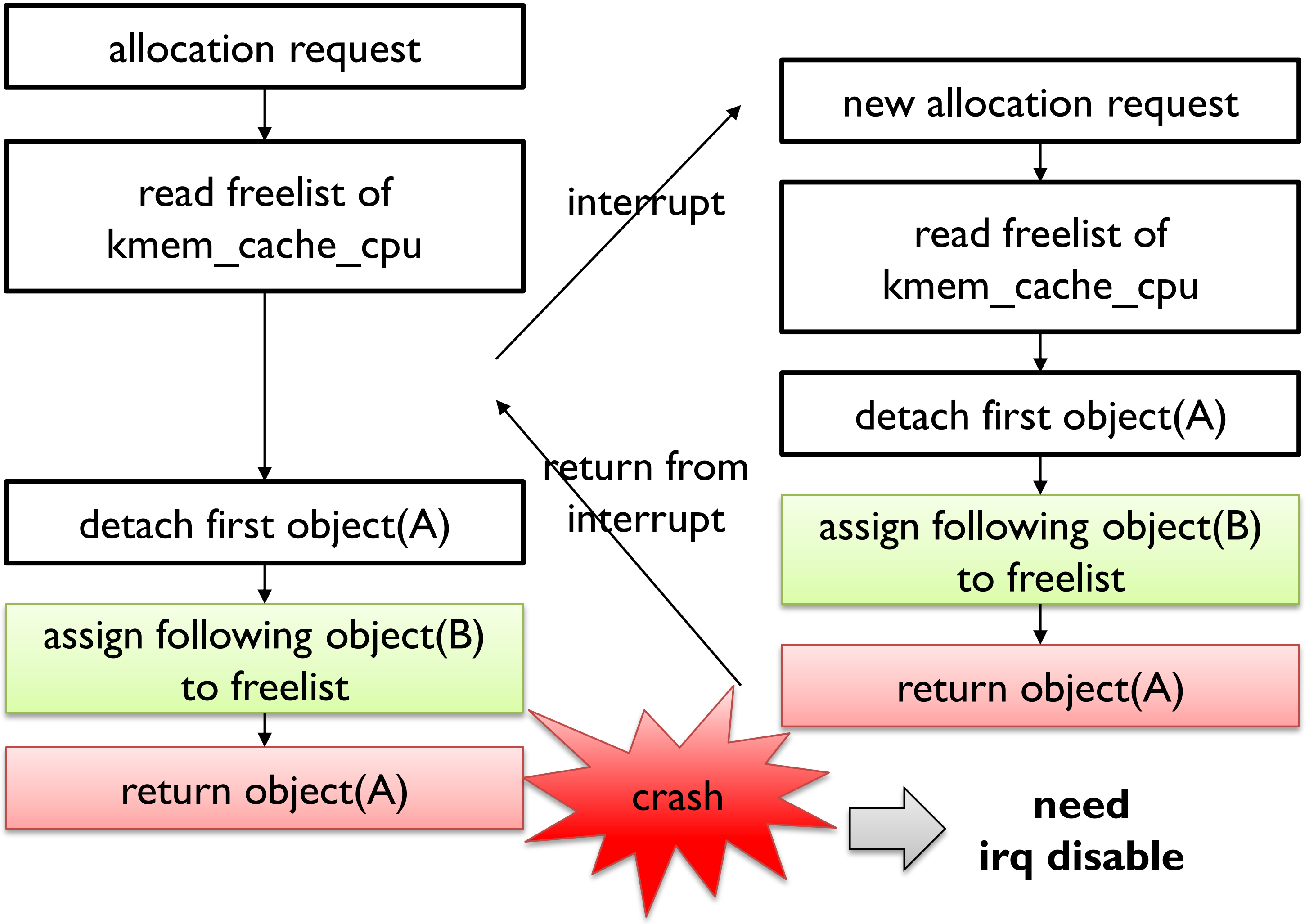
Free: slow-path



Performance optimization

- this_cpu_cmpxchg_double
  - avoid disabling interrupt
- cmpxchg_double
  - avoid taking a lock

Performance optimization: freelist of kmem_cache_cpu


allocation request

read freelist of kmem_cache_cpu

detach first object(A)

assign following object(B) to freelist

return object(A)

interrupt

new allocation request

read freelist of kmem_cache_cpu

detach first object(A)

assign following object(B) to freelist

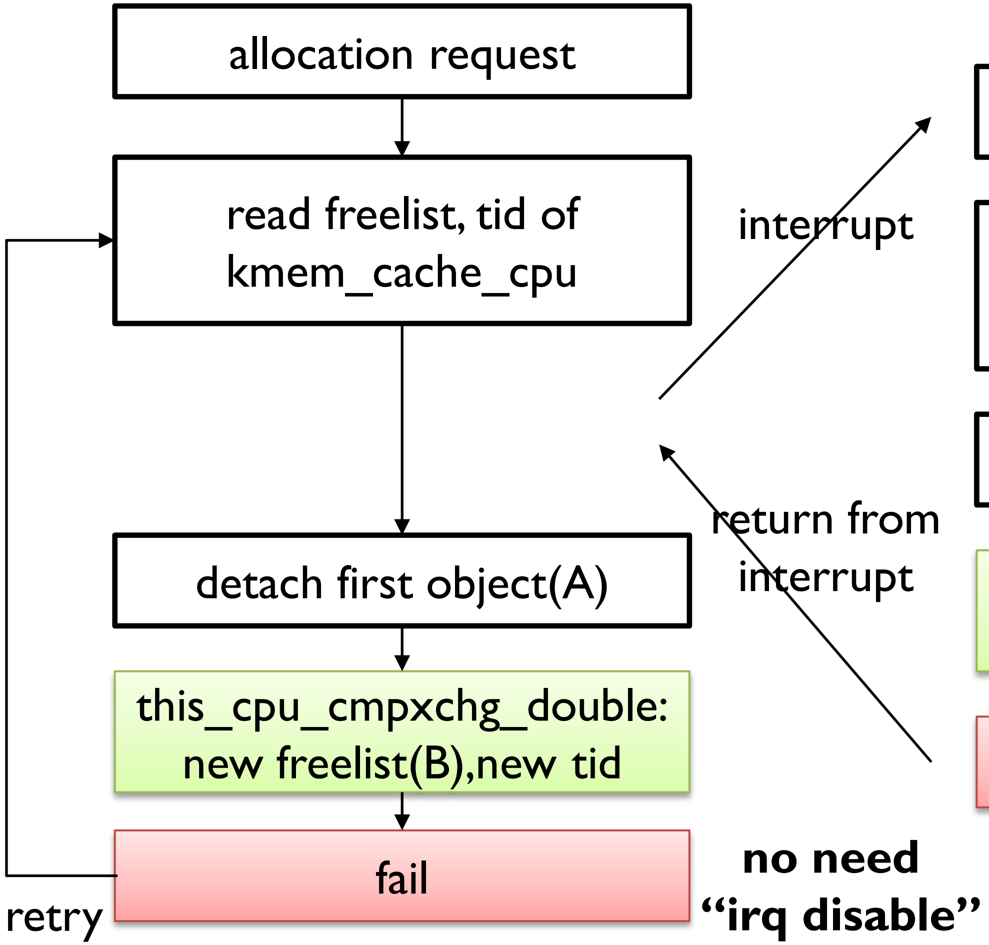
return object(A)

return from interrupt

crash

need irq disable

Performance optimization: freelist of kmem_cache_cpu



allocation request

read freelist, tid of kmem_cache_cpu

detach first object(A)

this_cpu_cmpxchg_double: new freelist(B),new tid

fail

retry

interrupt

new allocation request

read freelist, tid of kmem_cache_cpu

detach first object(A)

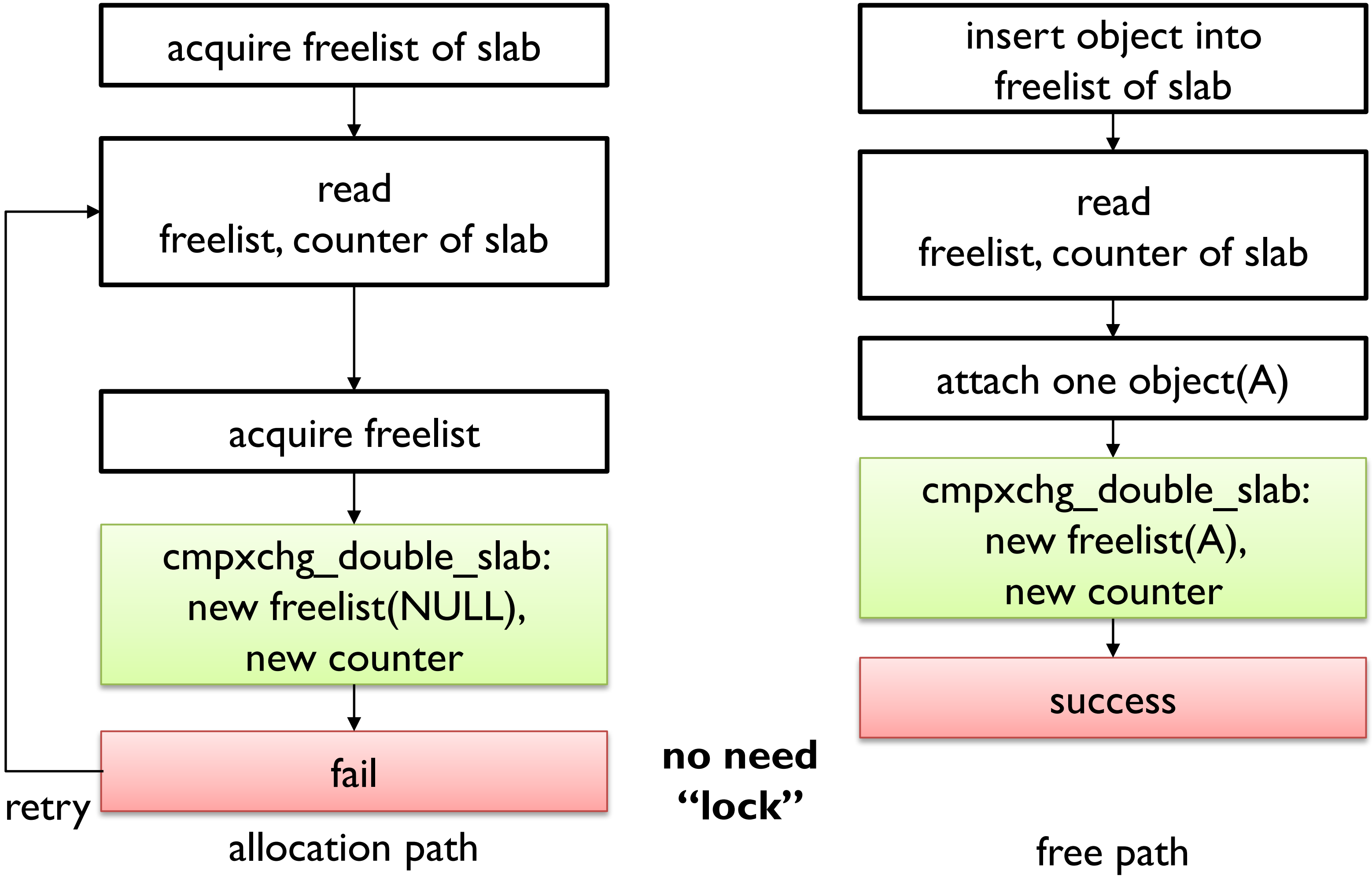
this_cpu_cmpxchg_double: new freelist(B),new tid

return object(A)

return from interrupt

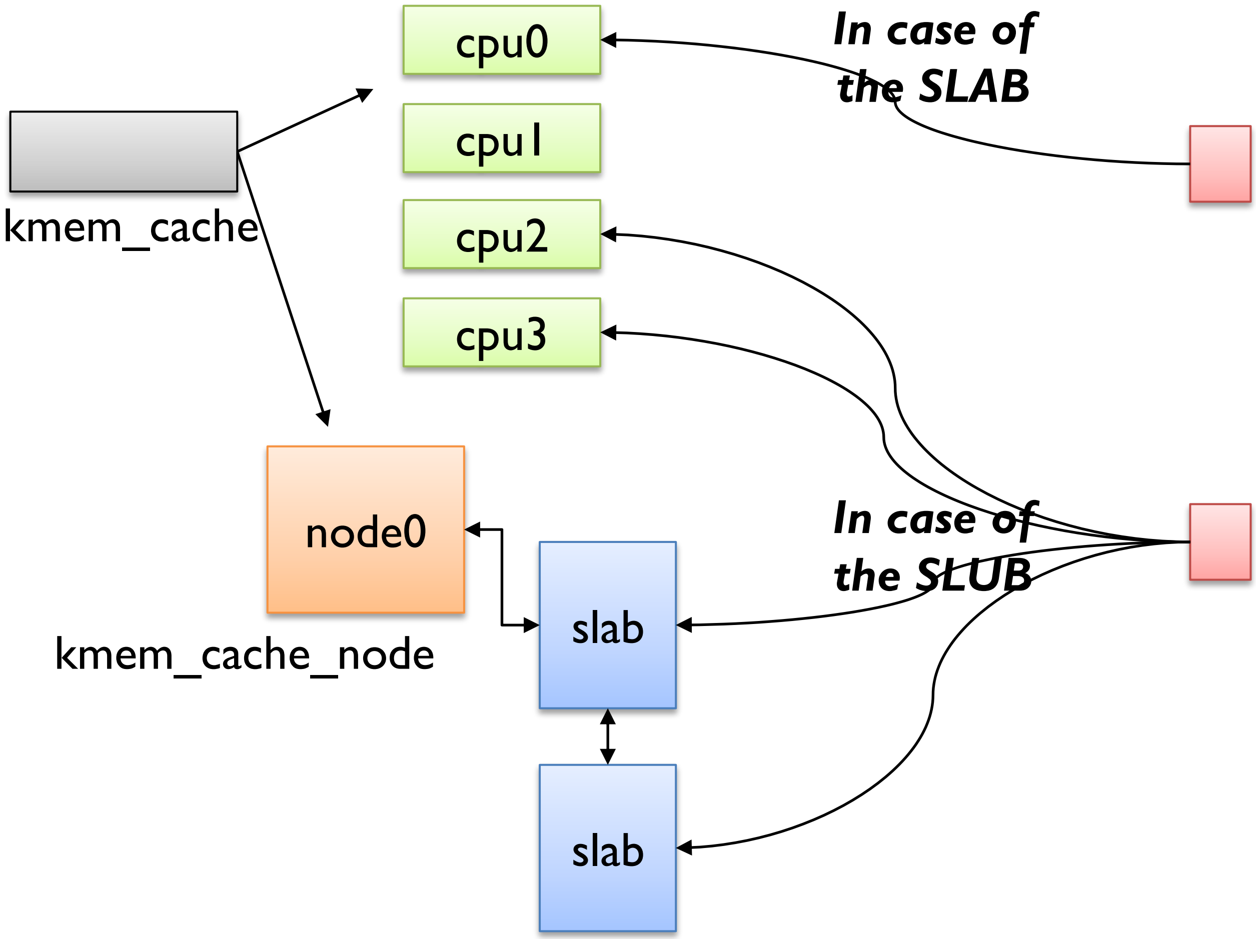
no need “irq disable”

Performance optimization: freelist of a slab



A DIFFERENCE BETWEEN THE SLAB AND THE SLUB

Caching policy



Free object management

cpu object cache	The SLAB	The SLUB
data structure	array	list
max number of objects	120	don’t care
size (64bits)	120 * 8 bytes	8 byte

slab	The SLAB	The SLUB
data structure	array	list
max number of objects	202	don’t care
size (64bits)	202 * 4 bytes	8 byte (overload “struct page”)

Miscellaneous

- kmalloc alignment
- fallback order slab
- kmem_cache alignment
- debugging feature
- NUMA

CURRENT STATUS

Trends

- per cpu partial lists
- common sl[aou]b
- slab accounting for memcg

Any questions?