Slab Allocator in Linux Kernel

Adrian Huang | Aug, 2022

*** Based on kernel 5.11 (x86_64) – QEMU**
*** 2-socket CPUs (4 cores/socket)**
*** 16GB memory**
*** Kernel parameter: nokaslr norandmaps**
*** KASAN: disabled**
*** Userspace: ASLR is disabled**
*** Legacy BIOS**

Agenda

- Slab & Buddy System

- Slab Concept
  - “Chicken or the egg” problem
  - Will not focus the differences about slab/slub/slob: Lots of info about them in Internet.
    - Note: SLUB is the default allocator since 2.6.23.

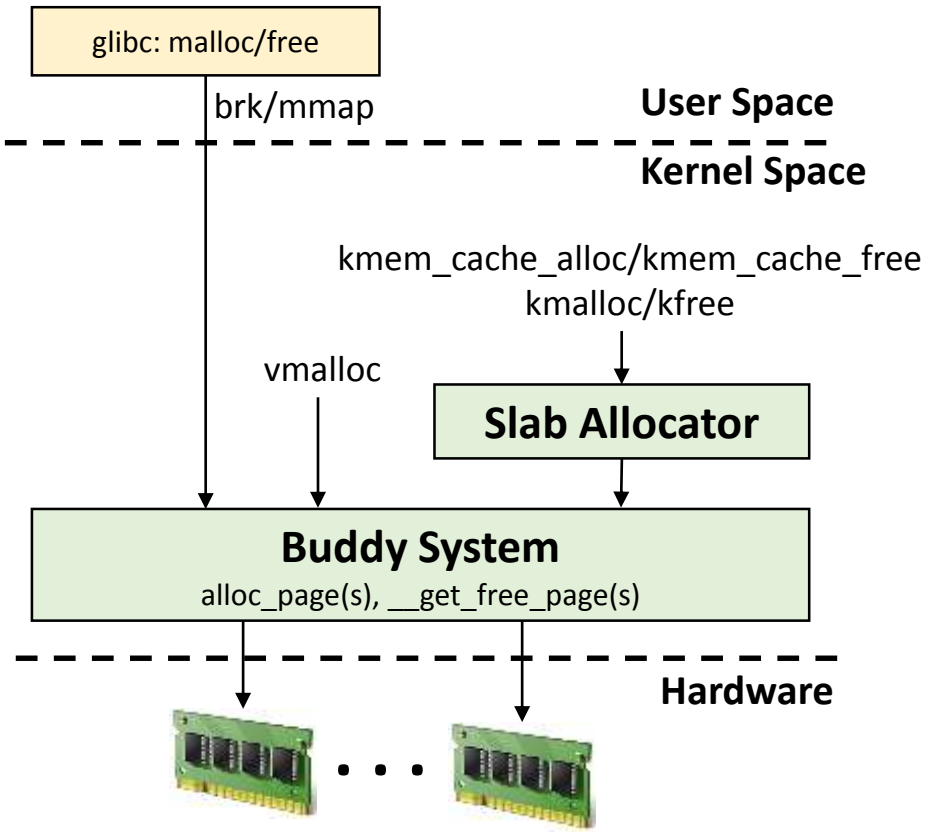
- [Init & Creation] kmem_cache_init(): Let’s check the initialization detail
  - How to initialize ‘boot_kmem_cache’ and ‘boot_kmem_cache_node’?
  - What are freepointers?
  - Data structures: kmem_cache & kmem_cache_node
  - [Comparison] Generic kmem_cache_create() & low-level implementation __kmem_cache_create()

- [Cache allocation] slab_alloc_node – Allocate a slab cache with specific node id
  - Fast path & slow path
    - Will follow kmem_cache_init() flow

- [Cache release/free] kmem_cache_free()
  - Fast path & slow path

- kmalloc & slab

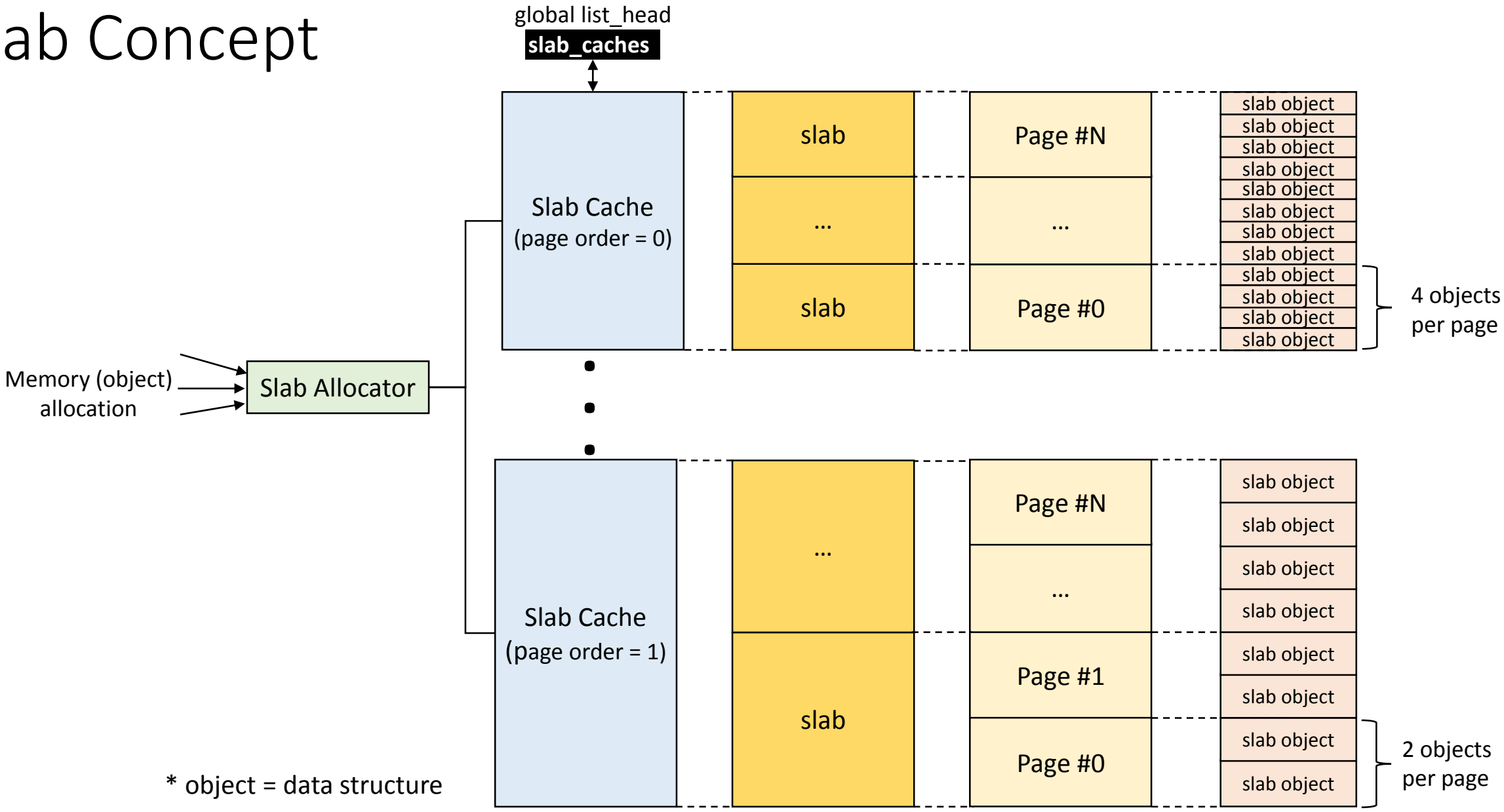
Slab & Buddy System



[glibc] malloc

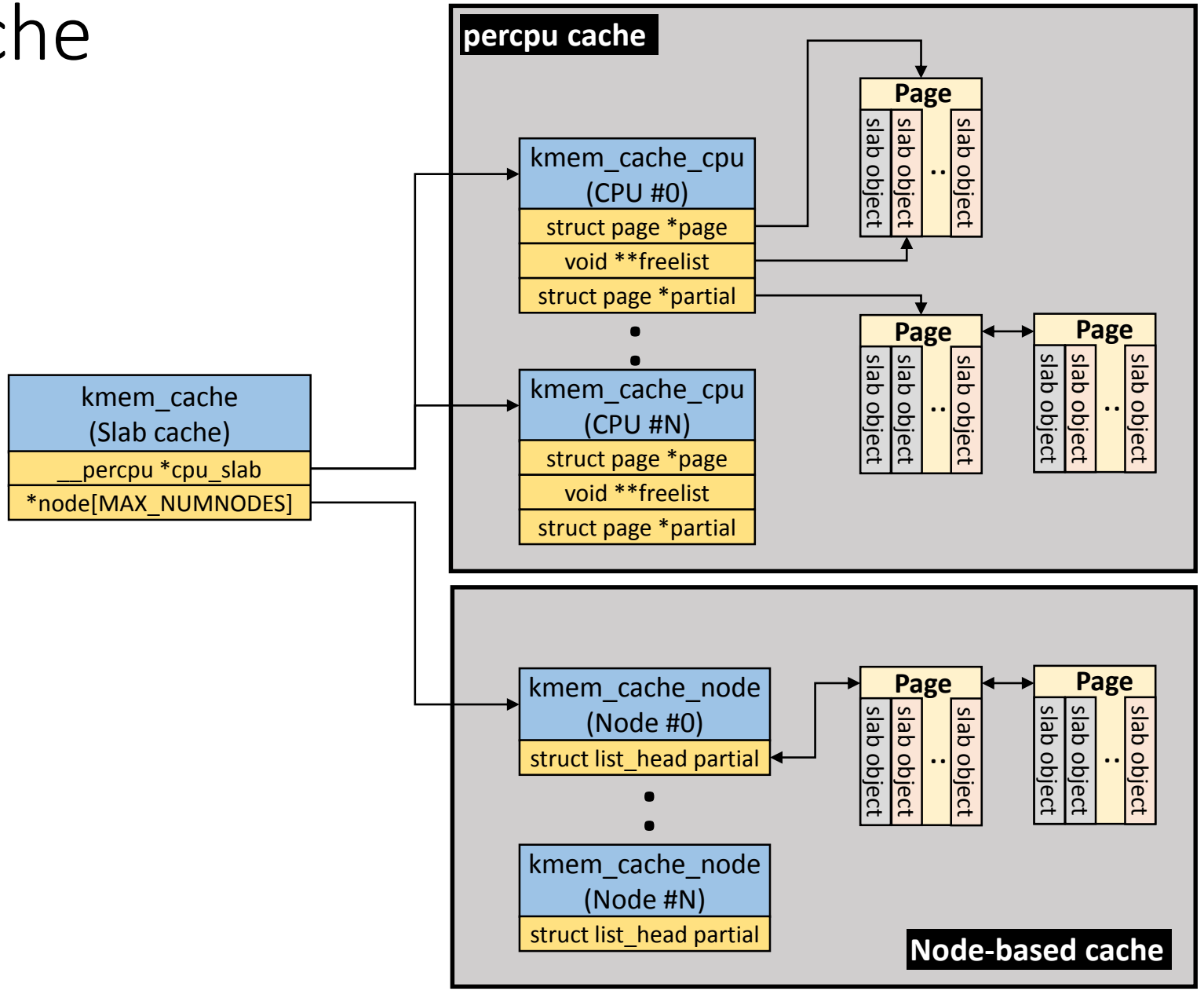
- Balance between **brk()** and **mmap()**
- Use brk() if request size < DEFAULT_MMAP_THRESHOLD_MIN (128 KB)
  - The heap can be trimmed only if memory is freed at the top end.
  - sbrk() is implemented as a library function that uses the brk() system call.
- Use mmap() if request size >= DEFAULT_MMAP_THRESHOLD_MIN (128 KB)
  - The allocated memory blocks can be independently released back to the system.
  - Deallocated space is not placed on the free list for reuse by later allocations.
  - Memory may be wasted because mmap allocations must be page-aligned; and the kernel must perform the expensive task of zeroing out memory allocated.
  - Note: glibc uses the dynamic mmap threshold
  - Detail: `man mallopt`

Slab Concept



1. **Allocating/freeing data structures is the common operations in Linux kernel**
2. **Slab is a generic data structure-caching layer**

Slab Cache



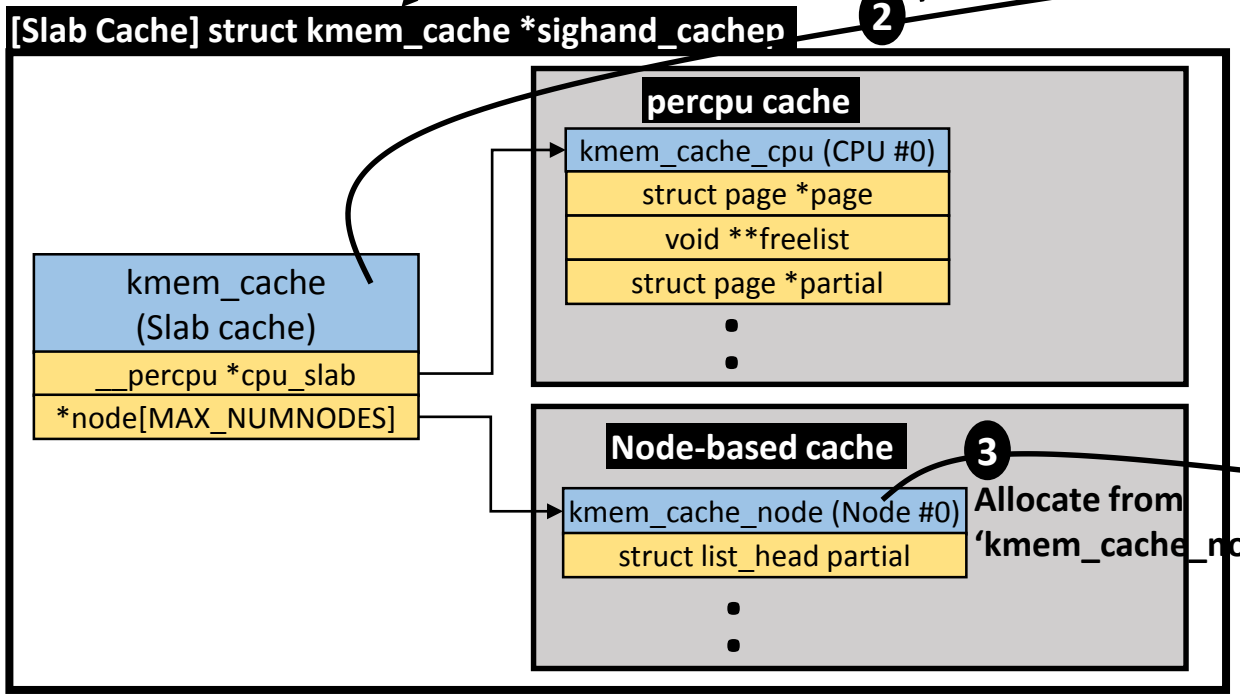
Node-based cache: freelist concept is implemented in page struct (not shown)

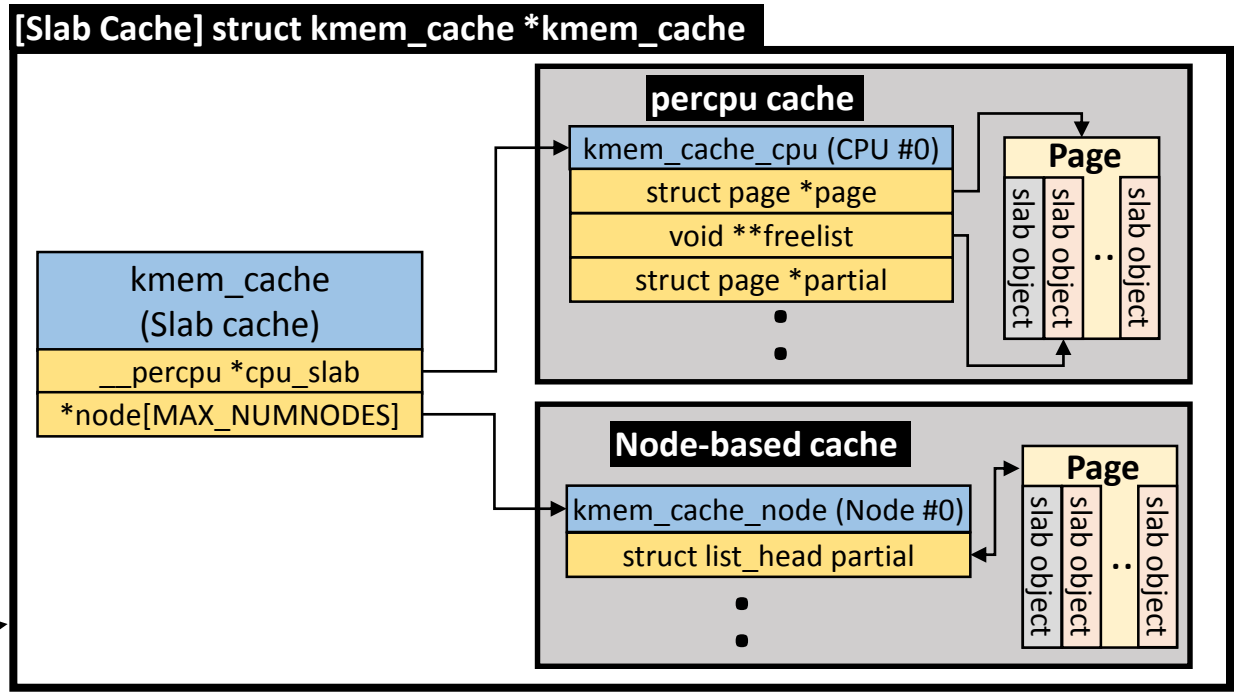
Slab Cache

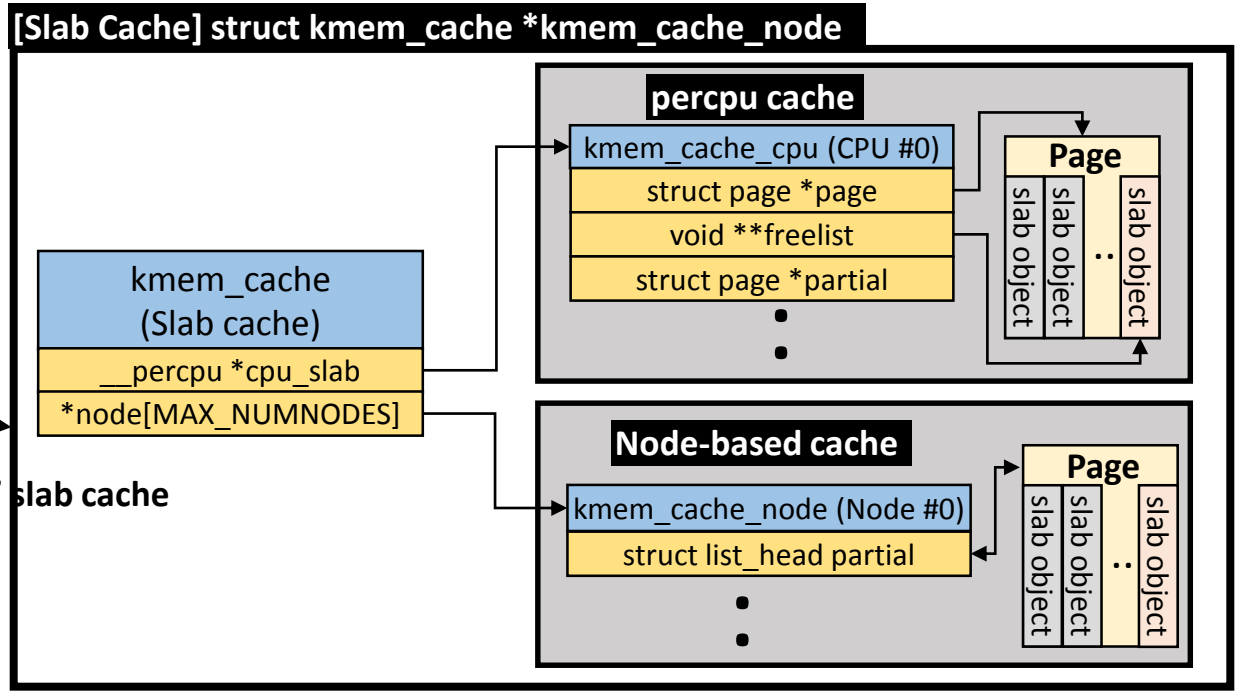
```
void __init proc_caches_init(void)
{
        unsigned int mm_size;

        sighand_cachep = kmem_cache_create("sighand_cache",
                         sizeof(struct sighand_struct), 0,
                         SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_TYPESAFE_BY_RCU|
                         SLAB_ACCOUNT, sighand_ctor);
+-- 29 lines: signal_cachep = kmem_cache_create("signal_cache",-------------
}
kernel/fork.c                                                  2771,1
```

1

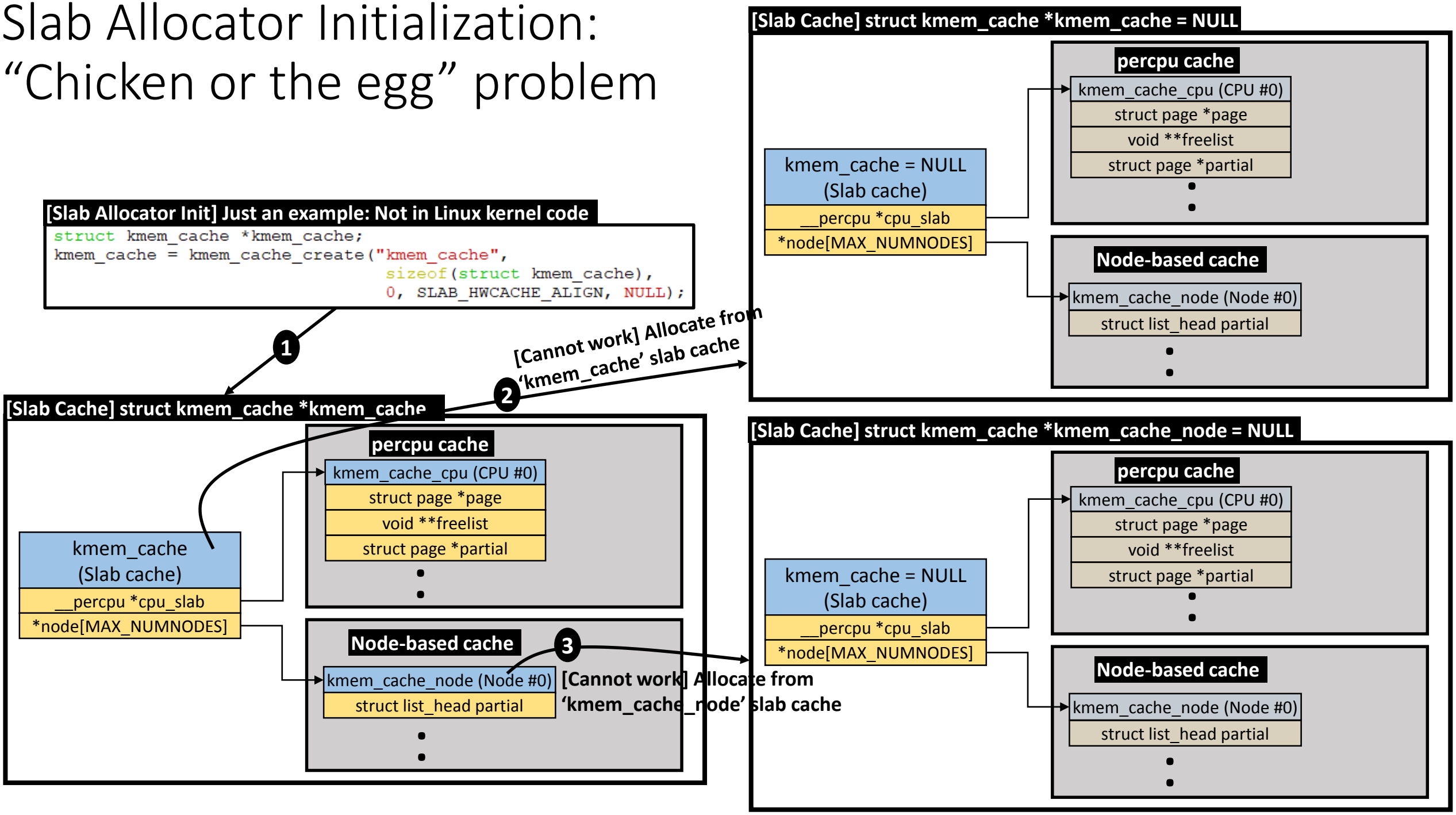
2 Allocate from 'kmem_cache' slab cache







Slab Allocator Initialization: “Chicken or the egg” problem



Slab Allocator Initialization: “Chicken or the egg” problem

[Example] Slab allocator init for kmem_cache

```
struct kmem_cache *kmem_cache;
kmem_cache = kmem_cache_create("kmem_cache",
                               sizeof(struct kmem_cache),
                               0, SLAB_HWCACHE_ALIGN, NULL);
```

1

[Slab Cache] struct kmem_cache *kmem_cache

kmem_cache (Slab cache)
__percpu *cpu_slab
*node[MAX_NUMNODES]

percpu cache
kmem_cache_cpu (CPU #0)
struct page *page
void **freelist
struct page *partial

Node-based cache
kmem_cache_node (Node #0)
struct list_head partial

2 **[Cannot work] Allocate from ‘kmem_cache’ slab cache**

[Slab Cache] struct kmem_cache *kmem_cache = NULL

kmem_cache = NULL (Slab cache)
__percpu *cpu_slab
*node[MAX_NUMNODES]

percpu cache
kmem_cache_cpu (CPU #0)
struct page *page
void **freelist
struct page *partial

Node-based cache
kmem_cache_node (Node #0)
struct list_head partial

3 **[Cannot work] Allocate from ‘kmem_cache_node’ slab cache**

[Slab Cache] struct kmem_cache *kmem_cache_node = NULL

kmem_cache = NULL (Slab cache)
__percpu *cpu_slab
*node[MAX_NUMNODES]

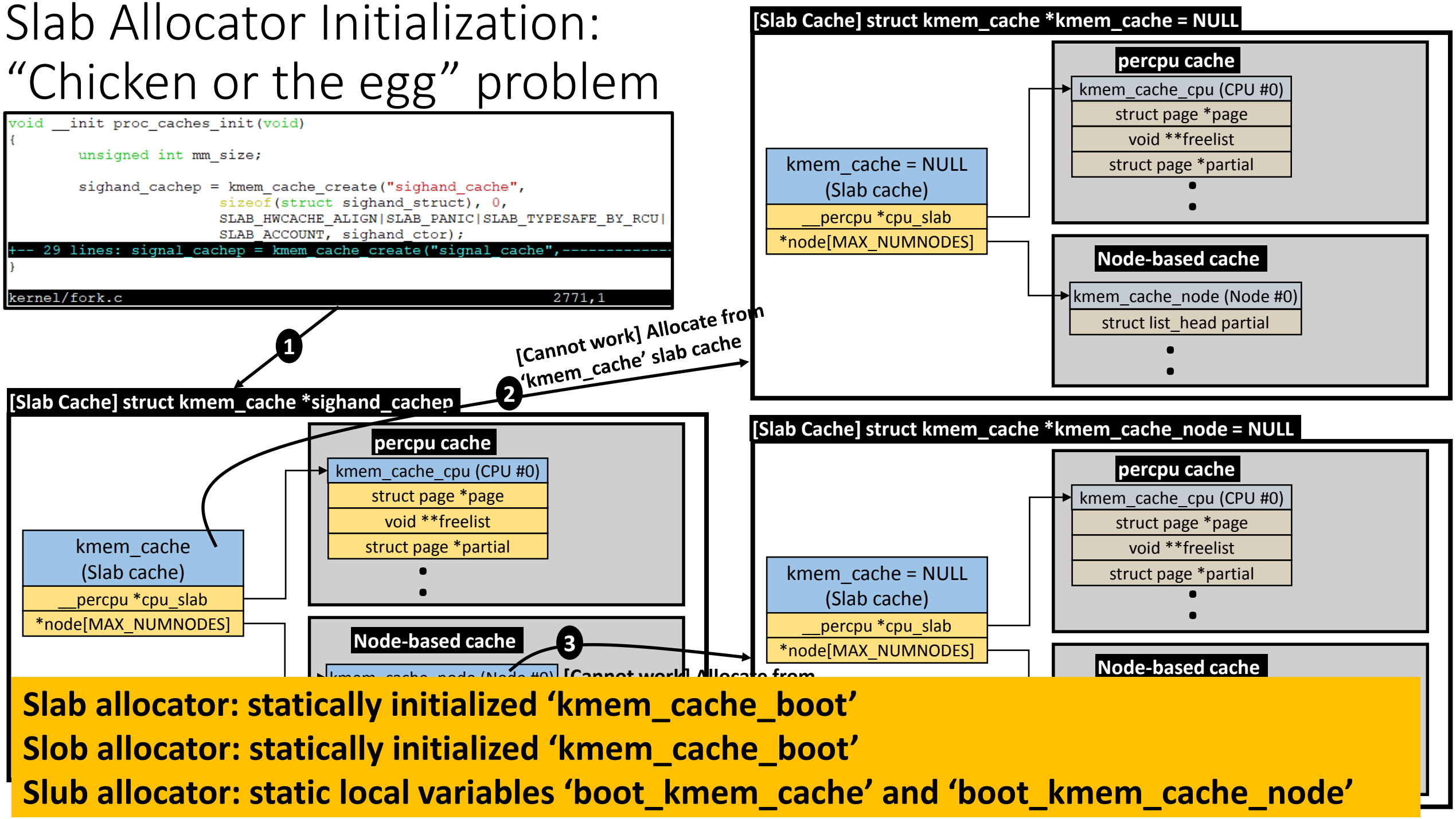
percpu cache
kmem_cache_cpu (CPU #0)
struct page *page
void **freelist
struct page *partial

Node-based cache
kmem_cache_node (Node #0)
struct list_head partial

Who initializes global variables ‘kmem_cache’ and ‘kmem_cache_node’?

Slab Allocator Initialization: “Chicken or the egg” problem

```
void __init proc_caches_init(void)
{
        unsigned int mm_size;

        sighand_cachep = kmem_cache_create("sighand_cache",
                        sizeof(struct sighand_struct), 0,
                        SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_TYPESAFE_BY_RCU|
                        SLAB_ACCOUNT, sighand_ctor);
+-- 29 lines: signal_cachep = kmem_cache_create("signal_cache",-------------
}

kernel/fork.c                                                  2771,1
```



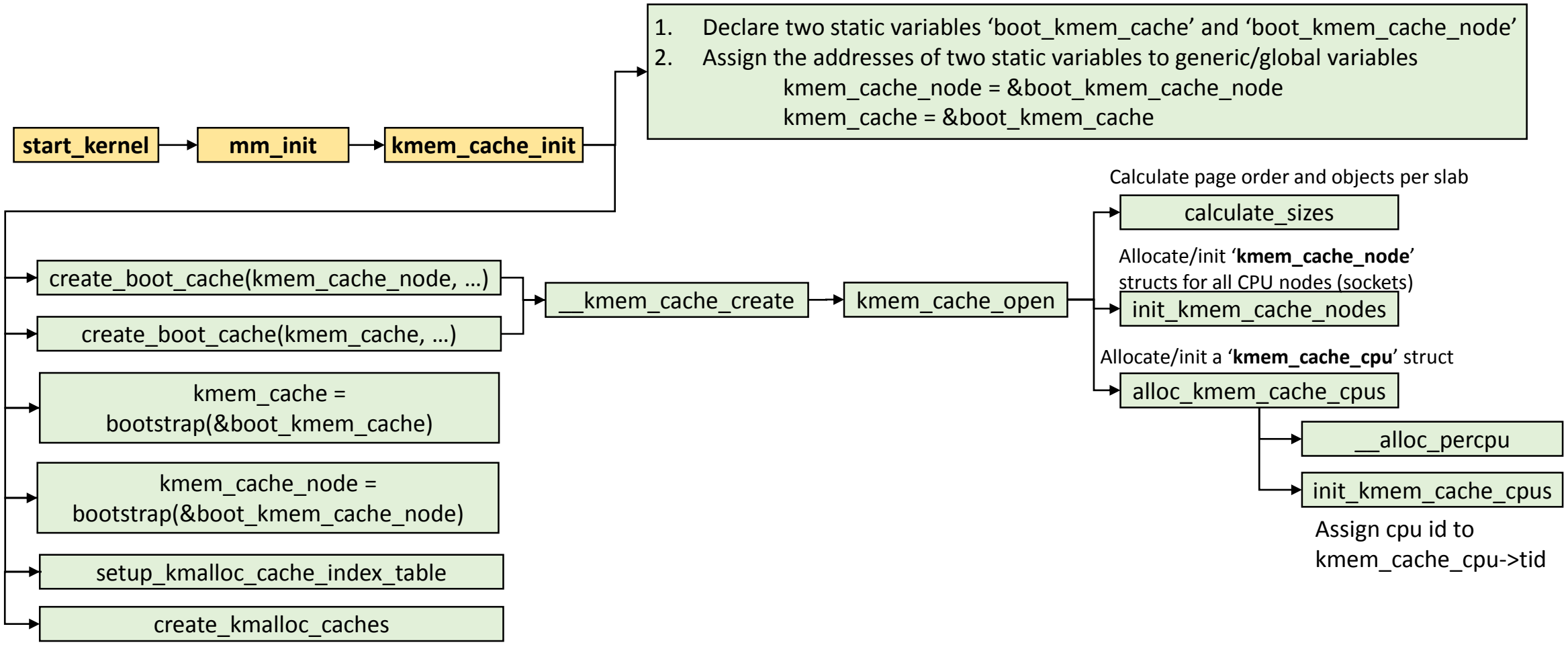
Slab allocator: statically initialized ‘kmem_cache_boot’
Slob allocator: statically initialized ‘kmem_cache_boot’
Slub allocator: static local variables ‘boot_kmem_cache’ and ‘boot_kmem_cache_node’

[Init & Creation] kmem_cache_init(): Let’s check the initialization detail

- How to initialize ‘boot_kmem_cache’ and ‘boot_kmem_cache_node’?
- What are freepointers?
- Data structures: kmem_cache & kmem_cache_node
- [Comparison] Generic kmem_cache_create() & low-level implementation __kmem_cache_create()

*** Call paths are based on SLUB (the unqueued slab allocator)**

kmem_cache_init()

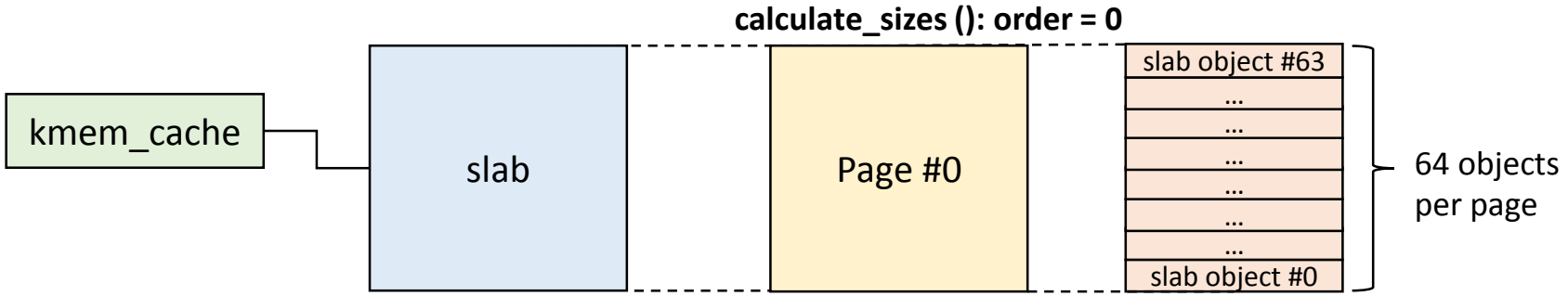


kmem_cache_node/boot_kmem_cache_node init

```
void __init kmem_cache_init(void)
{
        static __initdata struct kmem_cache boot_kmem_cache,
                boot_kmem_cache_node;

+-----  3 lines: if (debug_guardpage_minorder())----------------------------
        kmem_cache_node = &boot_kmem_cache_node;
        kmem_cache = &boot_kmem_cache;

        create_boot_cache(kmem_cache_node, "kmem_cache_node",
                sizeof(struct kmem_cache_node), SLAB_HWCACHE_ALIGN, 0, 0);

        register_hotmemory_notifier(&slab_memory_callback_nb);

+---------------- 25 lines: Able to allocate the per node structures -------
}
```

object size = sizeof(struct kmem_cache_node) = 64

object size with alignment (SLAB_HWCACHE_ALIGN: 64) = 64



* object = data structure

Slub allocator: A generic data structure-cache layer

kmem_cache_node/boot_kmem_cache_node init

object size = sizeof(struct kmem_cache_node) = 64
object size with alignment (SLAB_HWCACHE_ALIGN: 64) = 64

```
(gdb) bt
#0  init_kmem_cache_nodes (s=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3589
#1  kmem_cache_open (flags=<optimized out>,
    s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3845
#2  __kmem_cache_create (s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>,
    flags=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:4451
#3  0xffffffff81b1b84b in create_boot_cache (
    s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>,
    name=name@entry=0xffffffff819186dc "kmem_cache_node", size=size@entry=64,
    flags=flags@entry=8192, useroffset=useroffset@entry=0,
    usersize=usersize@entry=0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slab_common.c
:563
#4  0xffffffff81b1f44d in kmem_cache_init ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:4385
#5  0xffffffff81b00d99 in mm_init ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/init/main.c:832
#6  start_kernel ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/init/main.c:905
#7  0xffffffff81b00496 in x86_64_start_reservations (
    real_mode_data=real_mode_data@entry=0x13a10 <bts_ctx+2576> <error: Cannot ac
cess memory at address 0x13a10>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/kernel/
head64.c:525
#8  0xffffffff81b00519 in x86_64_start_kernel (
    real_mode_data=0x13a10 <bts_ctx+2576> <error: Cannot access memory at addres
s 0x13a10>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/kernel/
head64.c:506
#9  0xffffffff81000107 in secondary_startup_64 ()
```

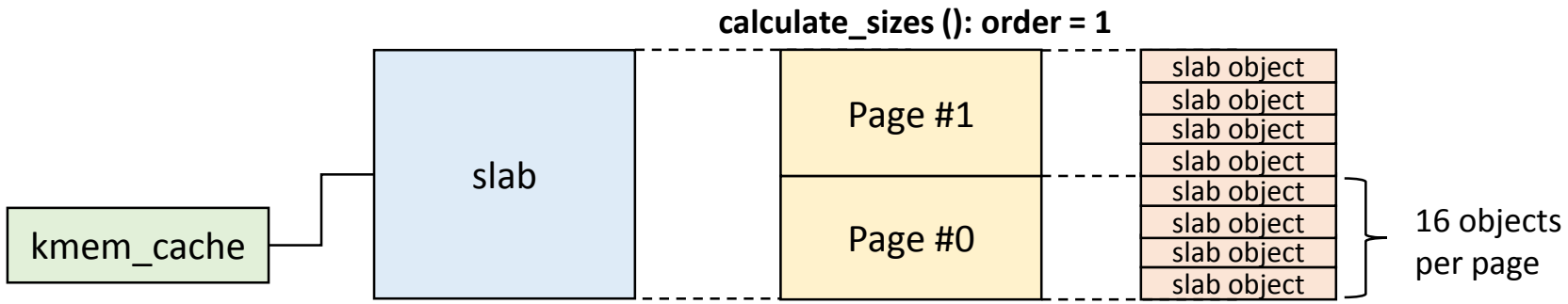
kmem_cache/boot_kmem_cache init

```
void __init kmem_cache_init(void)
{
        static __initdata struct kmem_cache boot_kmem_cache,
                boot_kmem_cache_node;
+--  3 lines: if (debug_guardpage_minorder())-------------------------------
        kmem_cache_node = &boot_kmem_cache_node;
        kmem_cache = &boot_kmem_cache;
+--  4 lines: create_boot_cache(kmem_cache_node, "kmem_cache_node",---------
        /* Able to allocate the per node structures */
        slab_state = PARTIAL;

        create_boot_cache(kmem_cache, "kmem_cache",
                        offsetof(struct kmem_cache, node) +
                                nr_node_ids * sizeof(struct kmem_cache_node *),
                        SLAB_HWCACHE_ALIGN, 0, 0);
+-- 17 lines: kmem_cache = bootstrap(&boot_kmem_cache);-------------------------
}
```

object size = offsetof(struct kmem_cache, node) +
nr_node_ids * sizeof(struct kmem_cache_node *)
nr_node_ids = 2 (2-socket guest OS)

object size = 200 + 2 * 8 = 216

object size with alignment (SLAB_HWCACHE_ALIGN: 64) = 256



* object = data structure

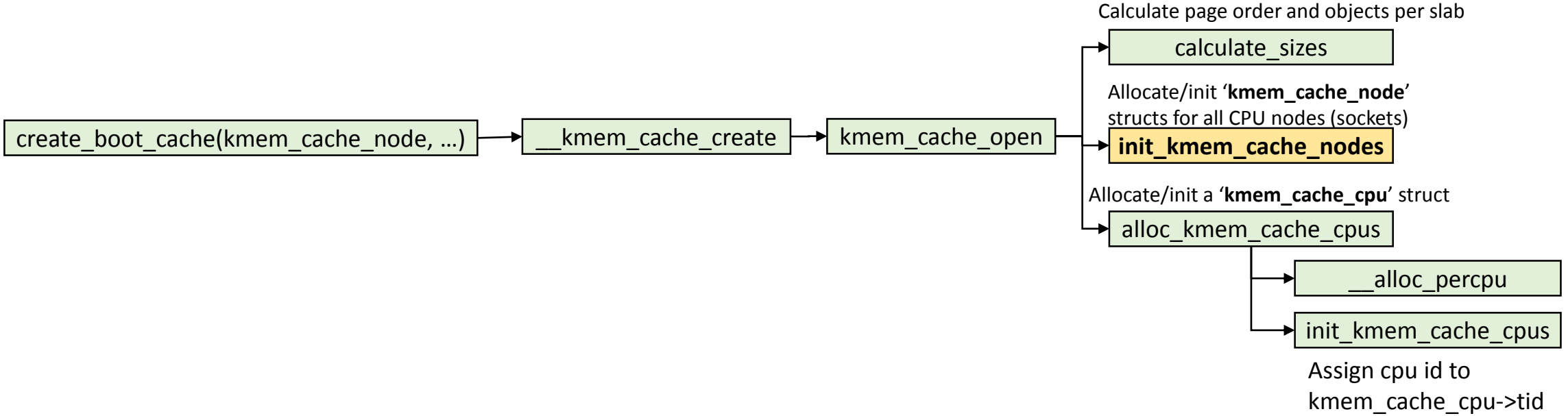
Slab allocator: A generic data structure-cache layer

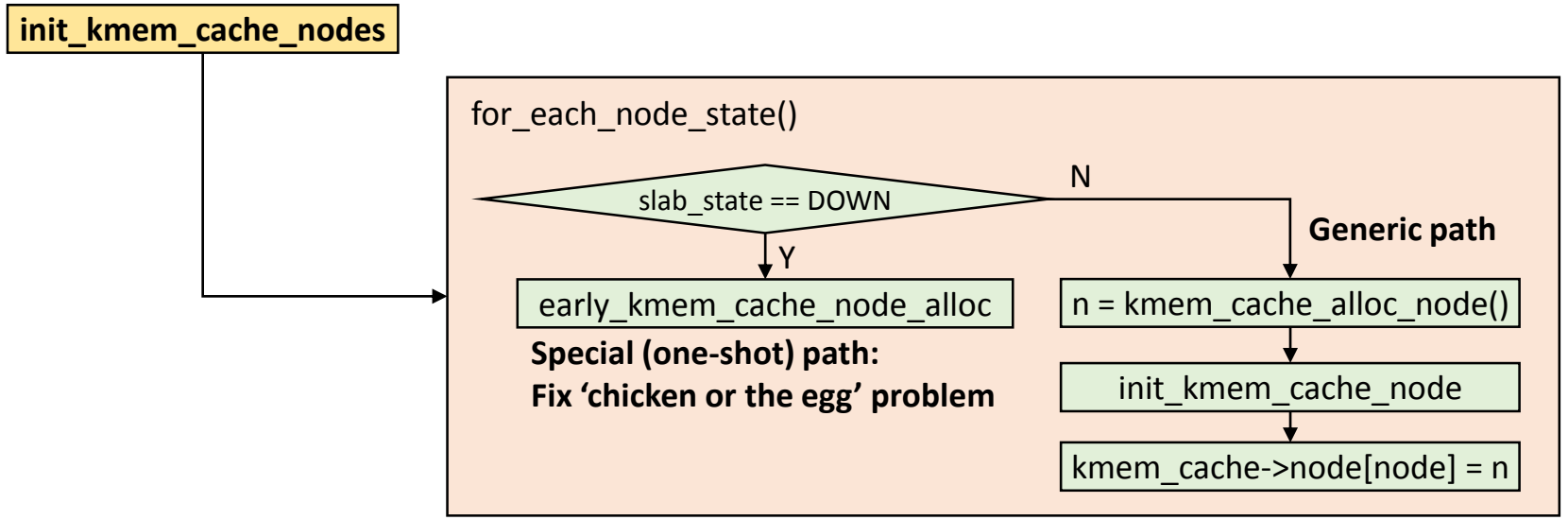
kmem_cache/boot_kmem_cache init

```
Thread 1 hit Breakpoint 2, init_kmem_cache_nodes (s=<optimized out>)
--Type <RET> for more, q to quit, c to continue without paging--
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3589
3589            for_each_node_state(node, N_NORMAL_MEMORY) {
(gdb) bt
#0  init_kmem_cache_nodes (s=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3589
#1  kmem_cache_open (flags=<optimized out>,
    s=s@entry=0xffffffff81bed380 <boot_kmem_cache>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3845
#2  __kmem_cache_create (s=s@entry=0xffffffff81bed380 <boot_kmem_cache>,
    flags=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:4451
#3  0xffffffff81b1b84b in create_boot_cache (
    s=0xffffffff81bed380 <boot_kmem_cache>,
    name=name@entry=0xffffffff819186ec "kmem_cache", size=216,
    flags=flags@entry=8192, useroffset=useroffset@entry=0,
    usersize=usersize@entry=0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slab_common.c
:563
#4  0xffffffff81b1f48e in kmem_cache_init ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:4393
#5  0xffffffff81b00d99 in mm_init ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/init/main.c:832
#6  start_kernel ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/init/main.c:905
#7  0xffffffff81b00496 in x86_64_start_reservations (
    real_mode_data=real_mode_data@entry=0x13a10 <bts_ctx+2576> <error: Cannot ac
cess memory at address 0x13a10>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/kernel/
head64.c:525
#8  0xffffffff81b00519 in x86_64_start_kernel (
    real_mode_data=0x13a10 <bts_ctx+2576> <error: Cannot access memory at addres
s 0x13a10>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/kernel/
head64.c:506
#9  0xffffffff81000107 in secondary_startup_64 ()
```

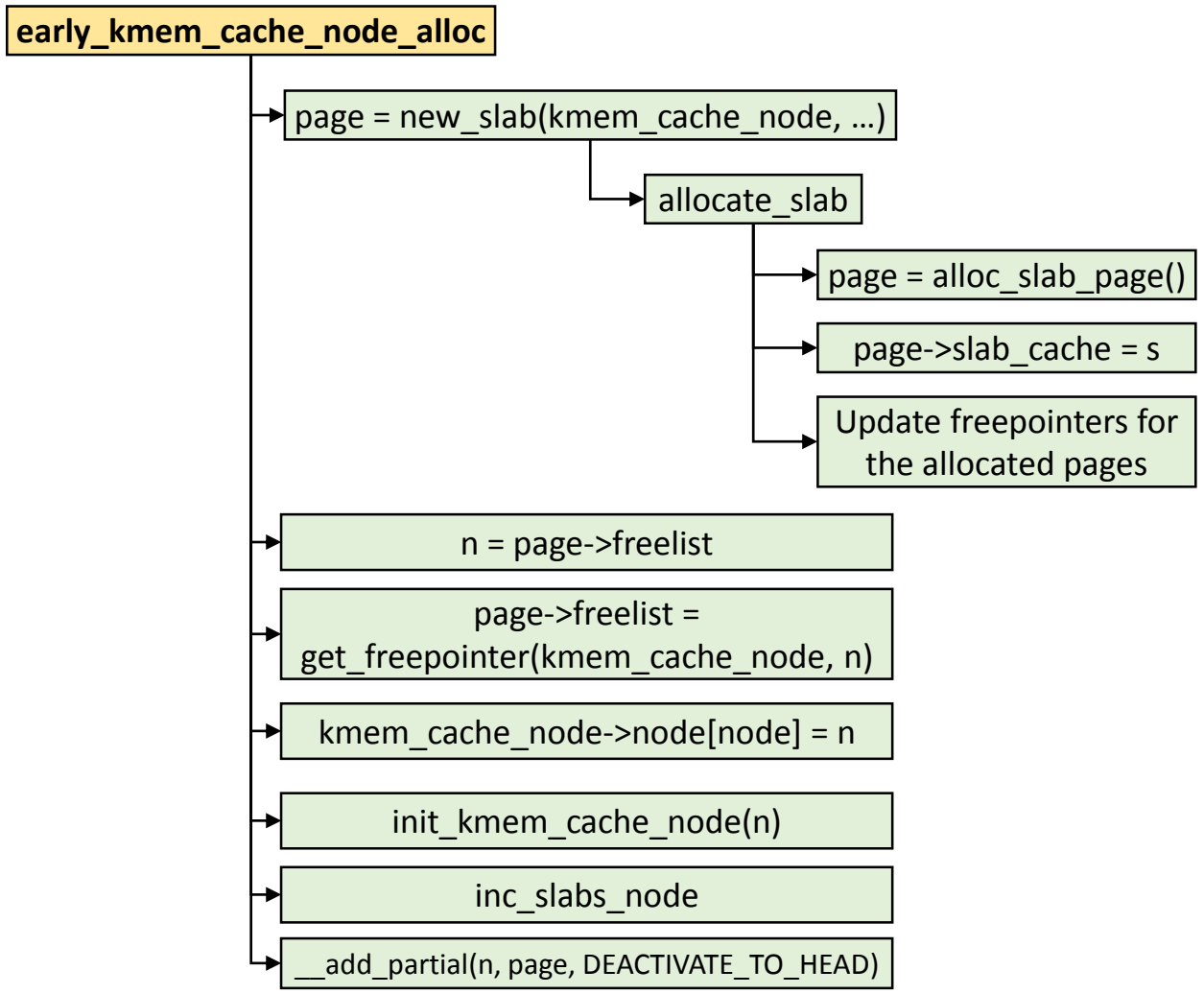
object size = 200 + 2 * 8 = 216

init_kmem_cache_nodes()

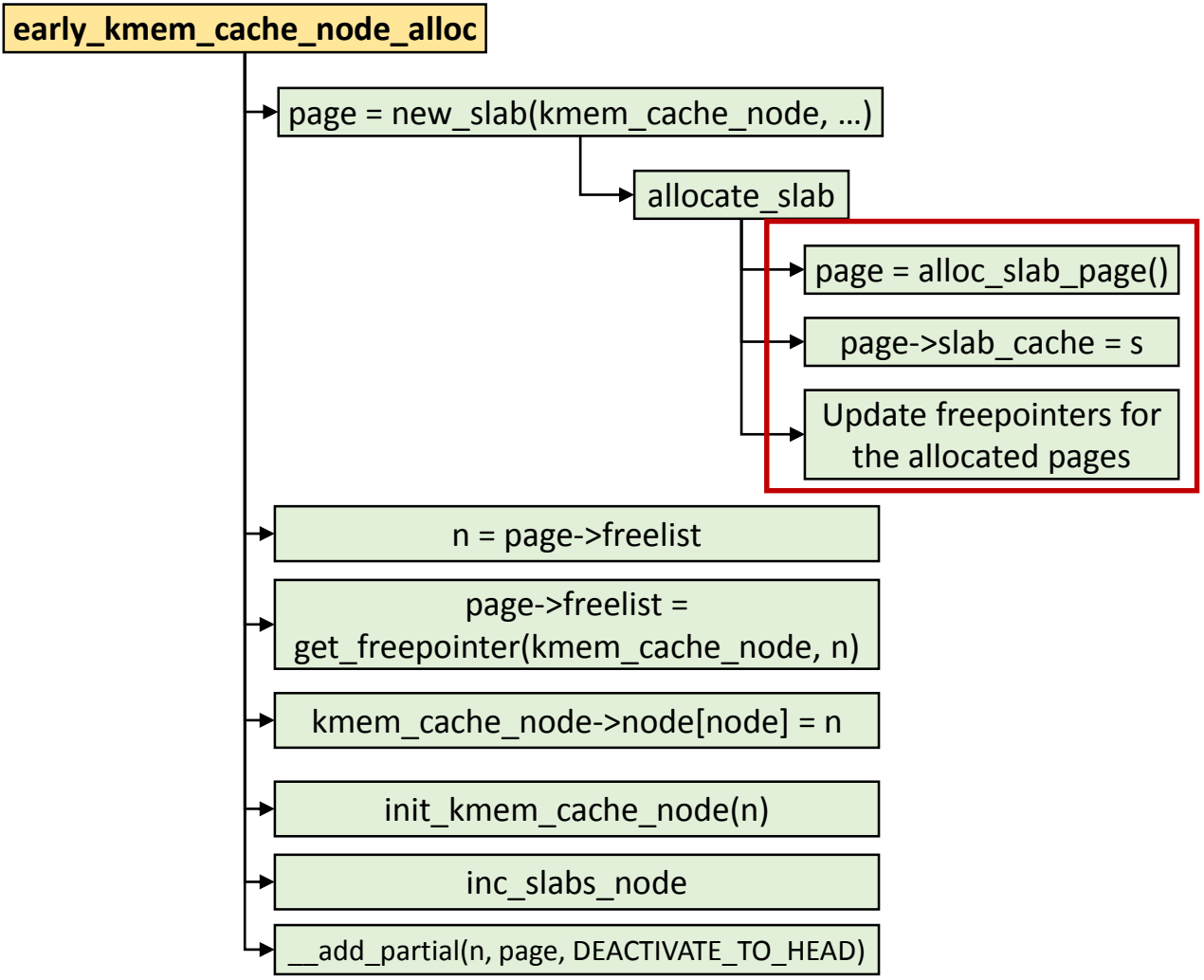


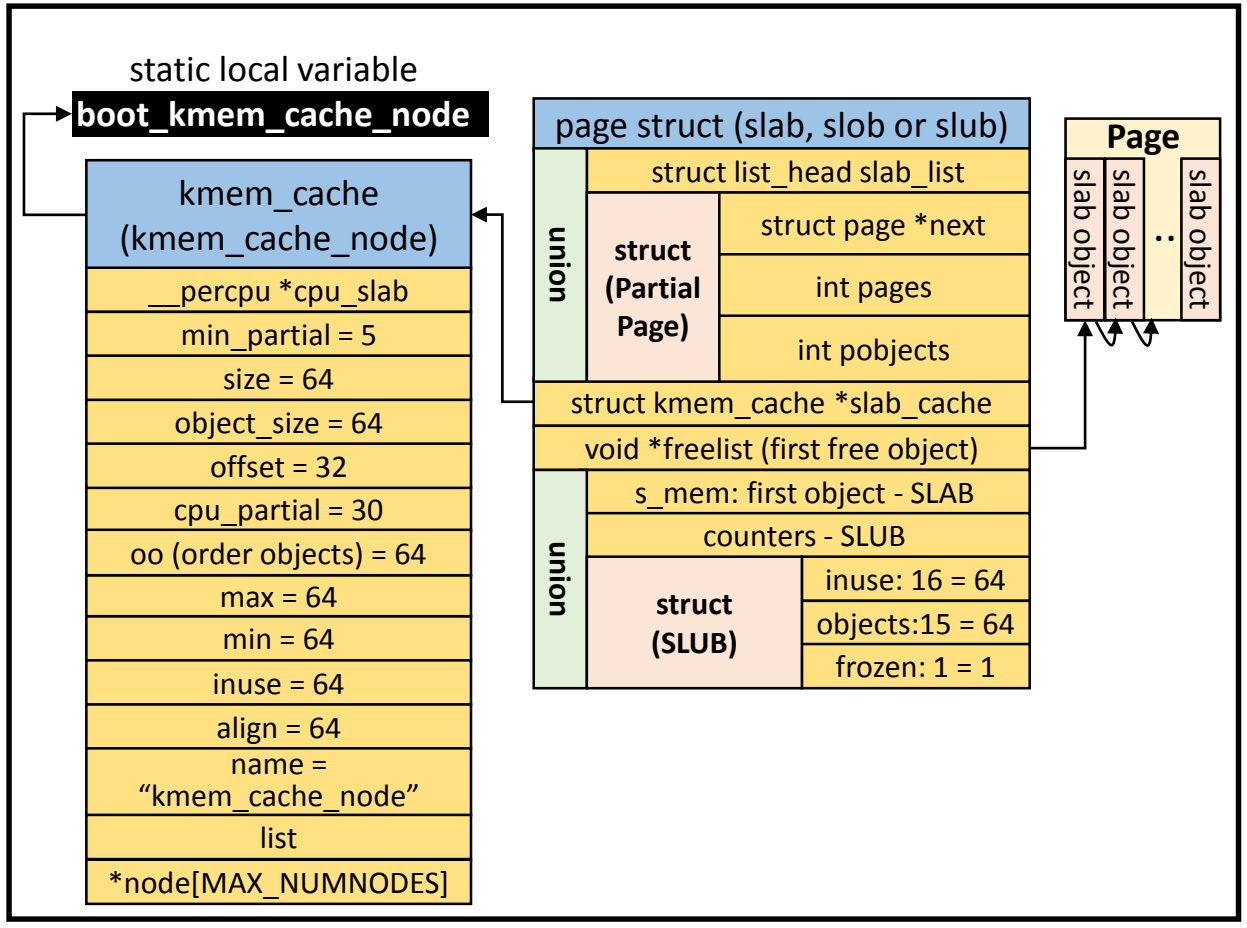


early_kmem_cache_node_alloc for ‘kmem_cache_node’



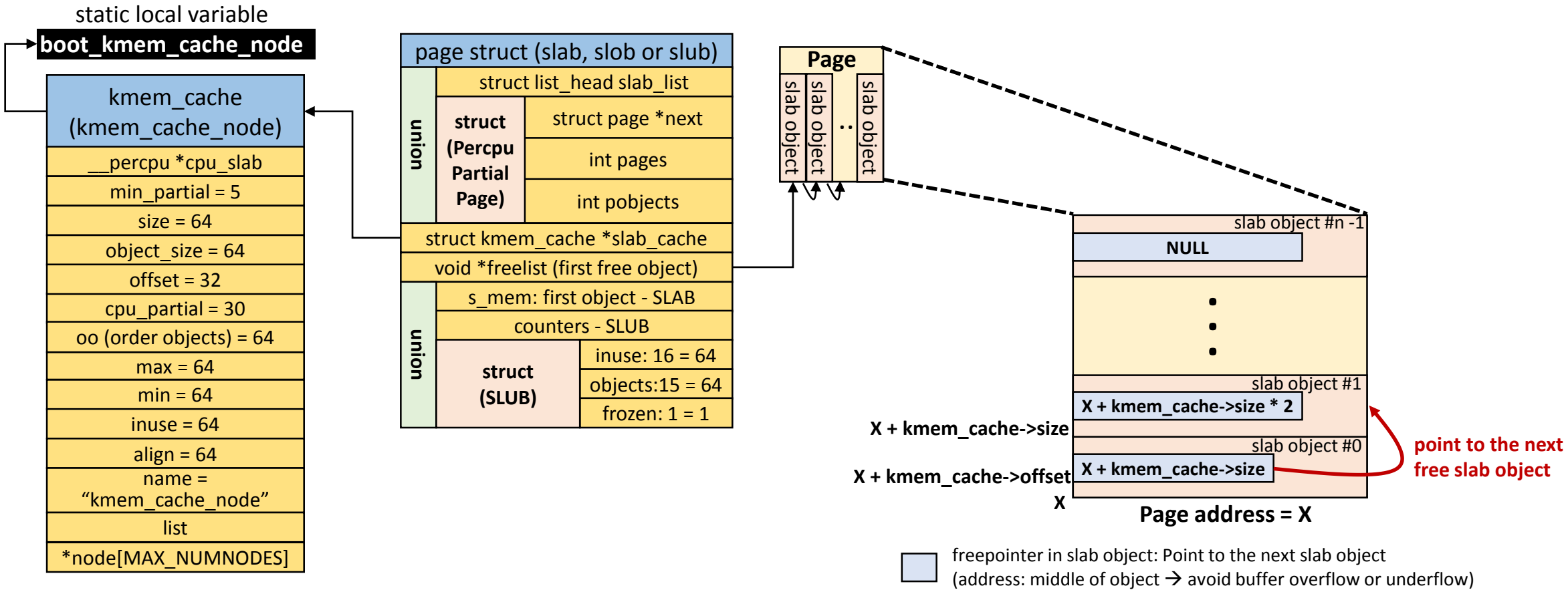
allocate_slab() – freelist/freepointer configuration





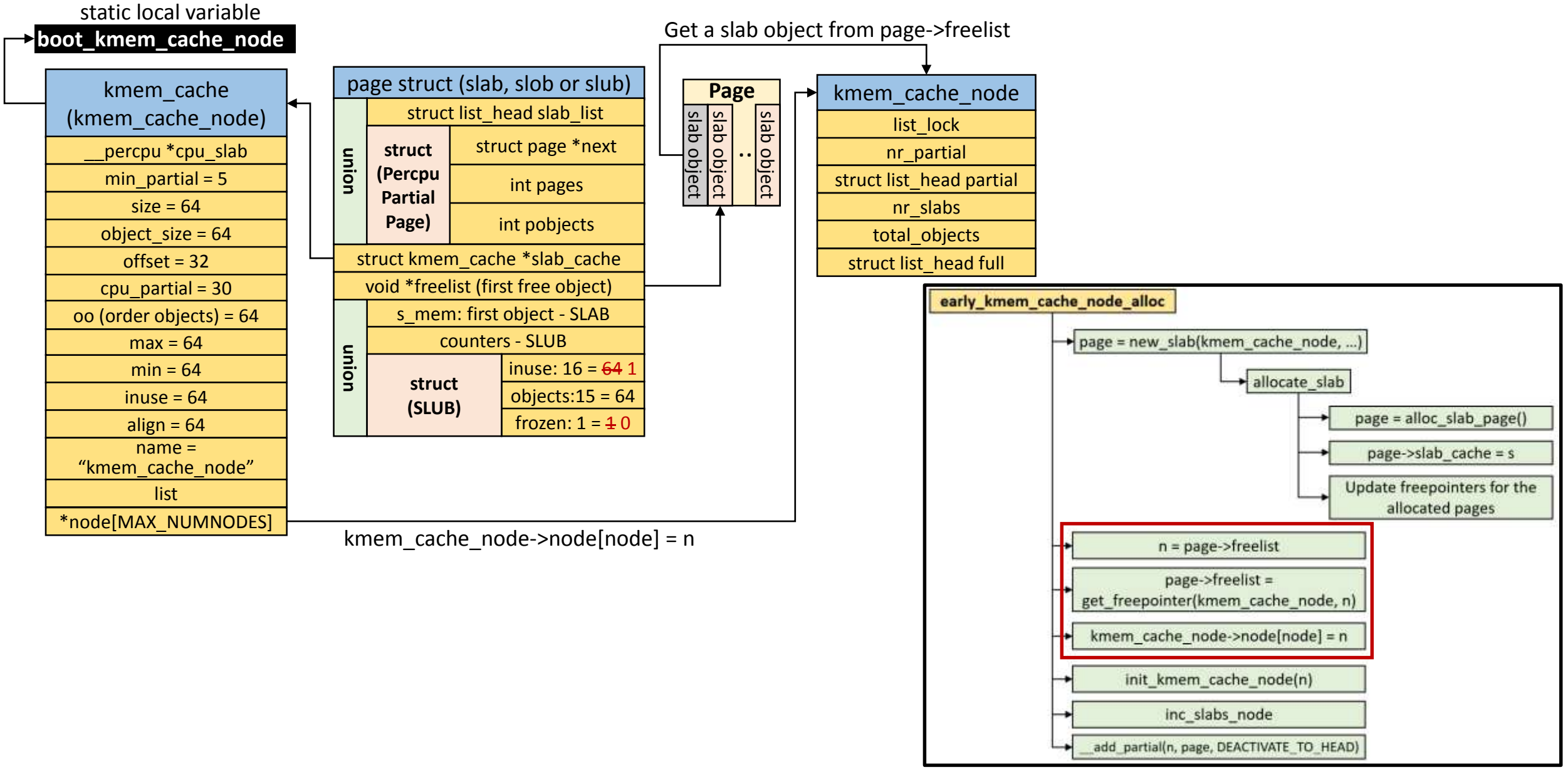
Reference functions: get_freepointer/set_freepointer

allocate_slab() – freelist/freepointer configuration

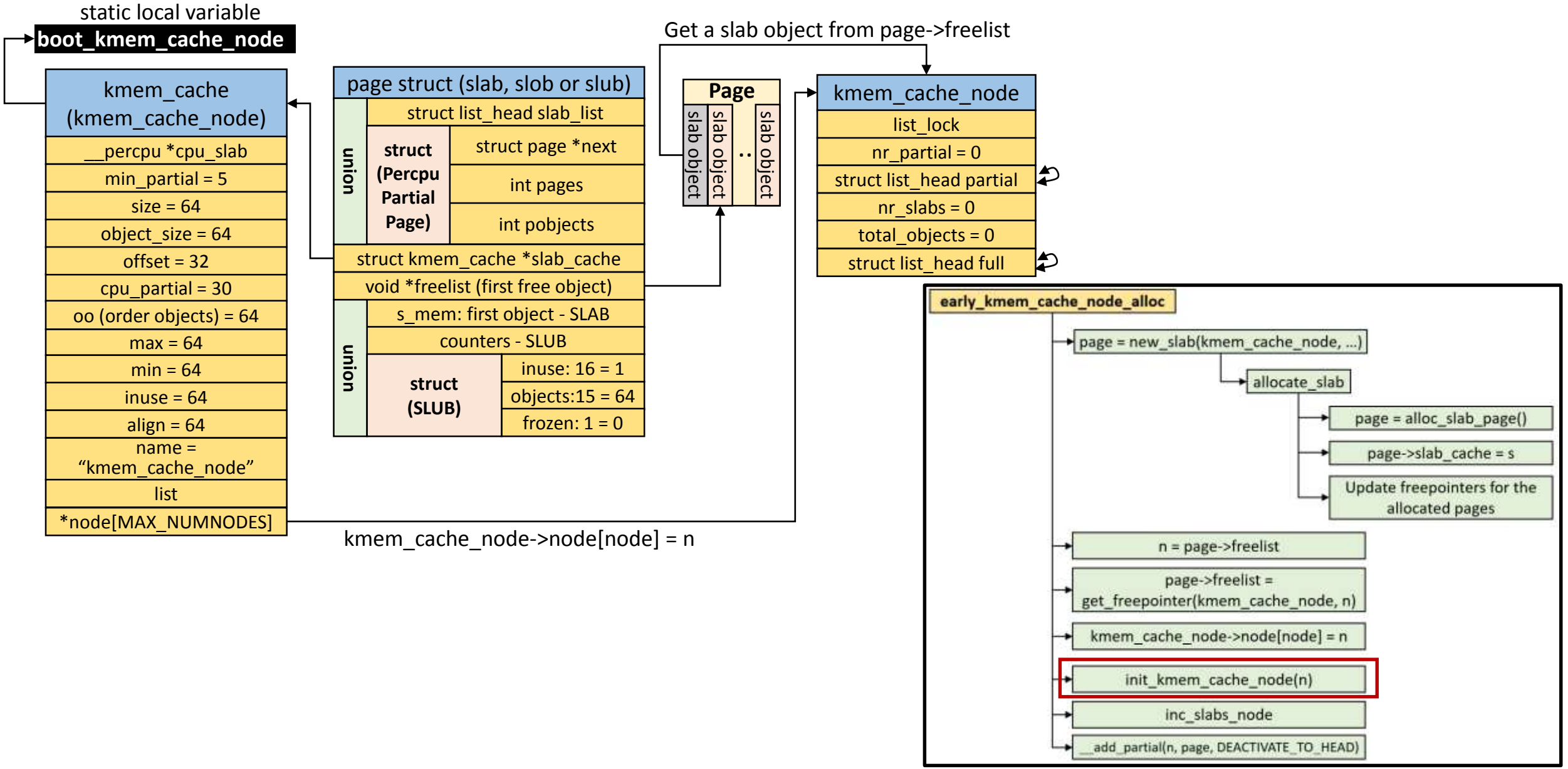


Reference functions: get_freepointer/set_freepointer

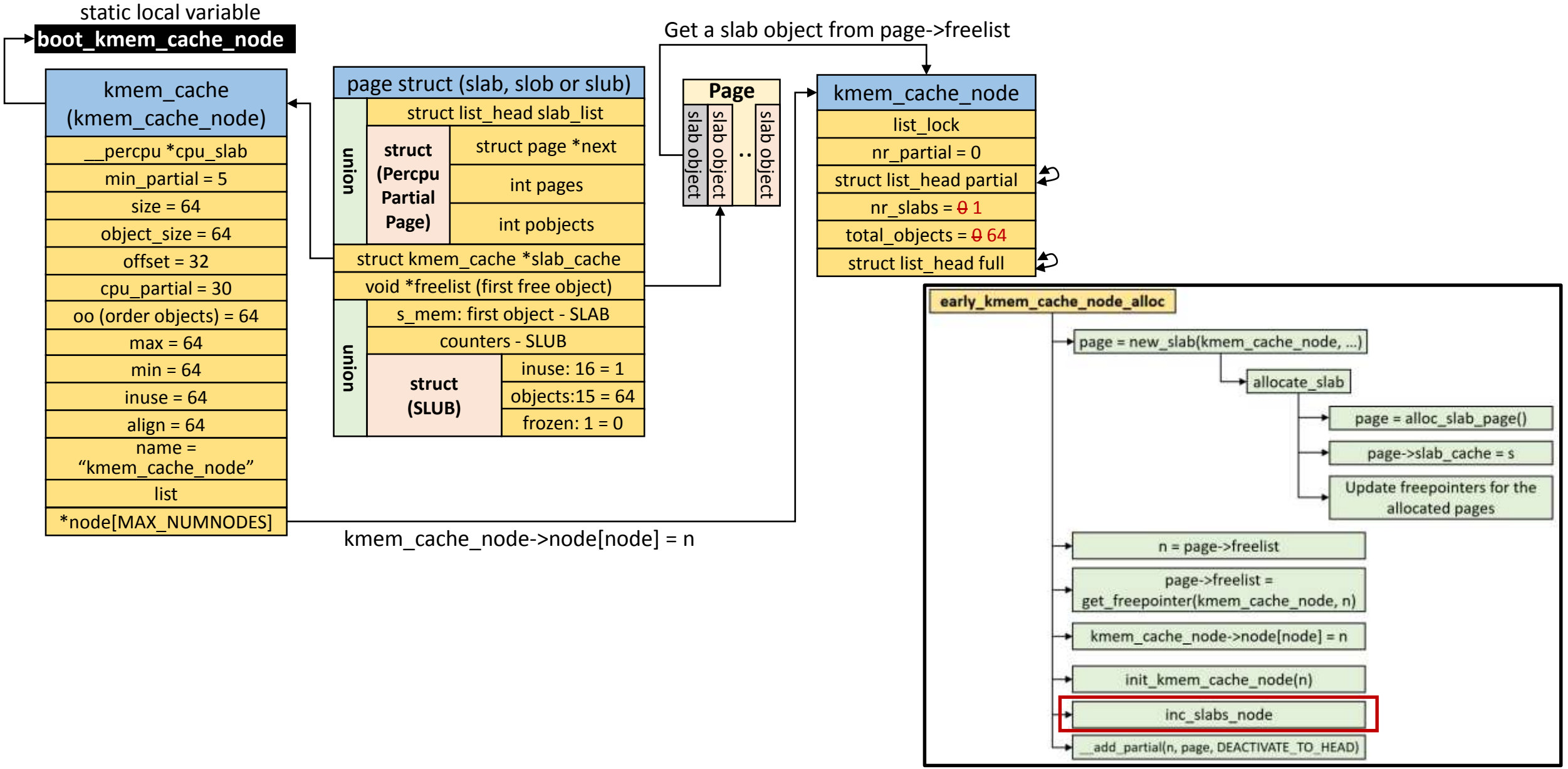
early_kmem_cache_node_alloc() – Get a slab object from freelist



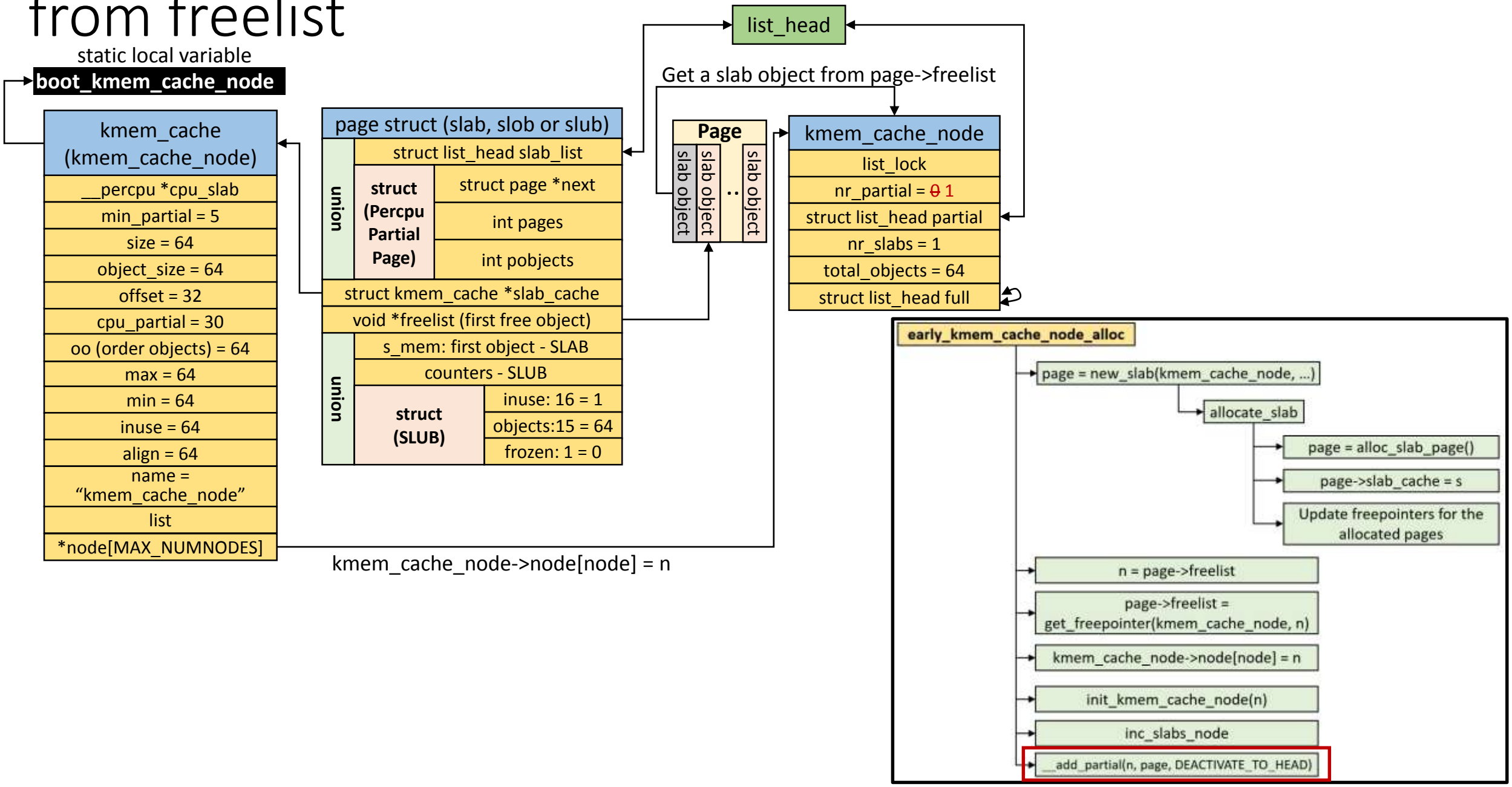
early_kmem_cache_node_alloc() – Get a slab object from freelist



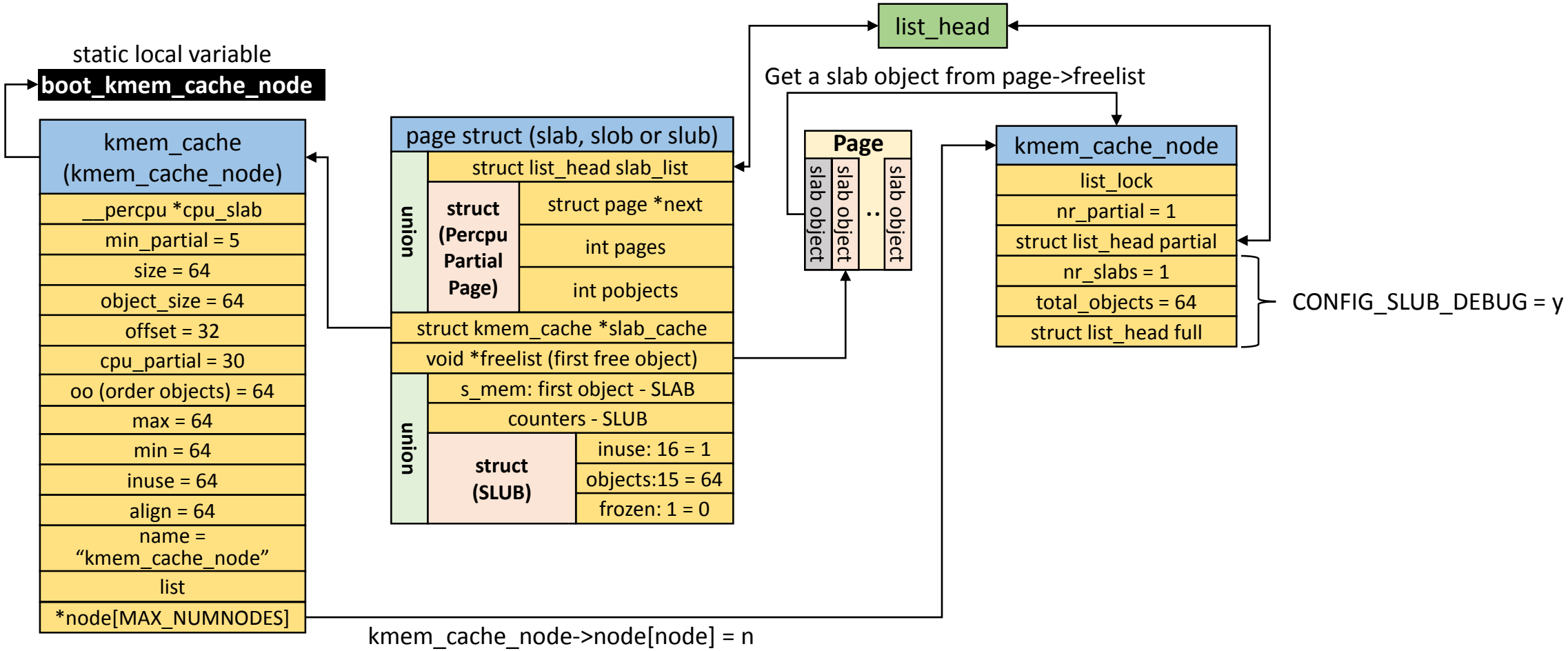
early_kmem_cache_node_alloc() – Get a slab object from freelist



early_kmem_cache_node_alloc() – Get a slab object from freelist



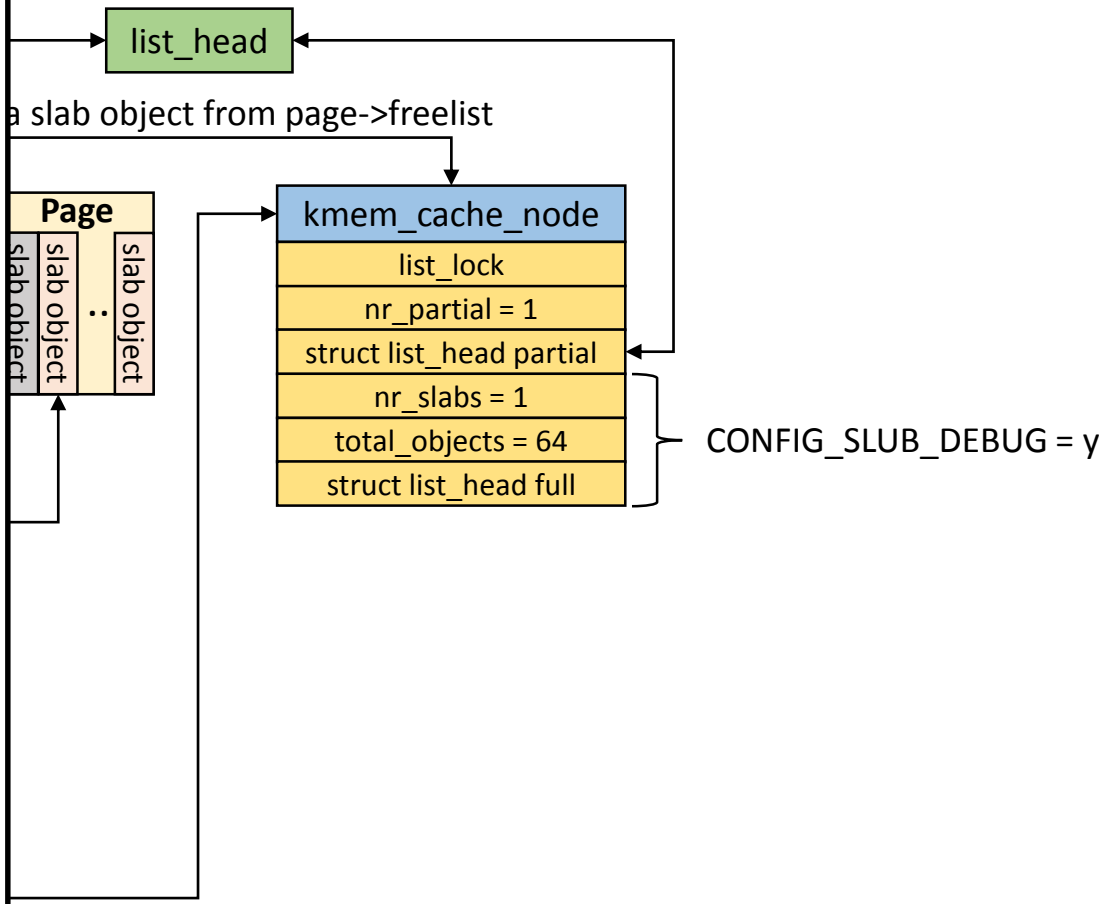
early_kmem_cache_node_alloc(): configure node #0



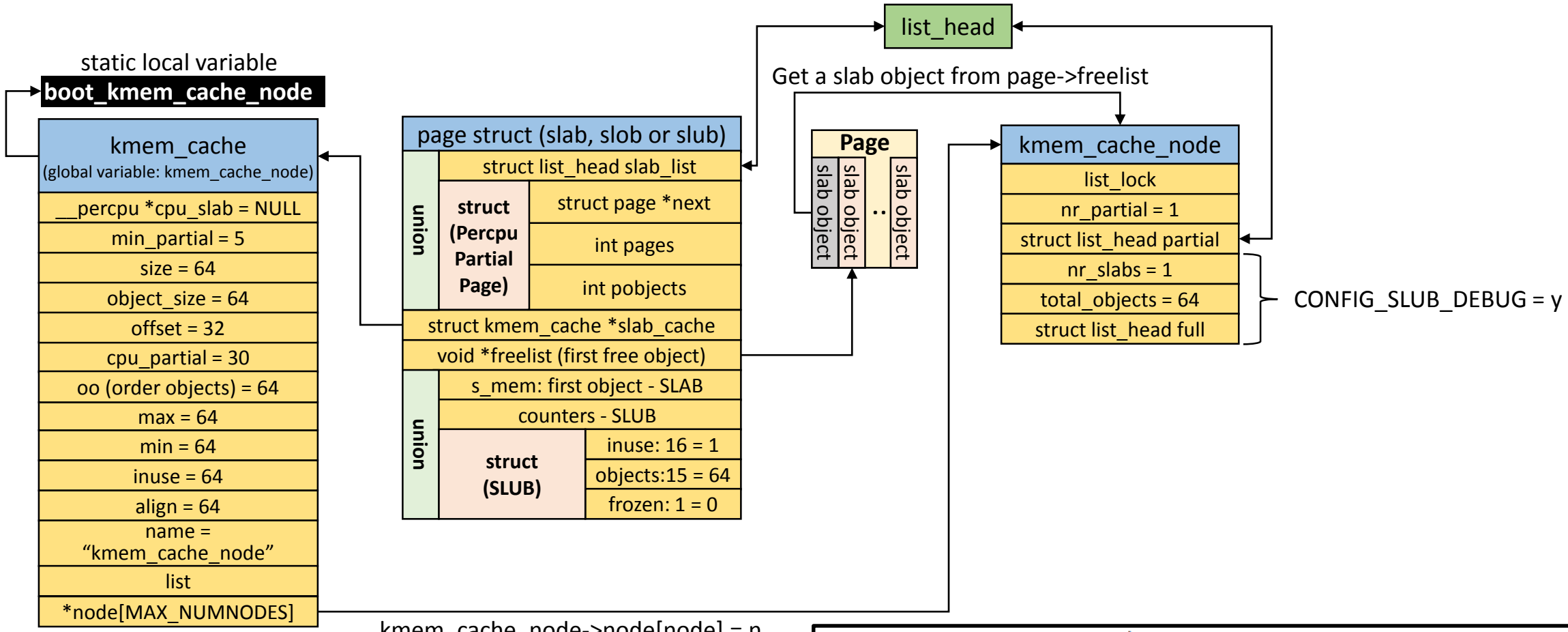
Allocate slab page(s) for kmem_cache_node struct allocation: Fix “the chicken or egg problem”

early_kmem_cache_node_alloc(): configure node #0

```
(gdb) bt 2
#0  early_kmem_cache_node_alloc (node=1)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3593
#1  init_kmem_cache_nodes (s=0xffffffff81beb2a0 <boot_kmem_cache_node>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:3593
(More stack frames follow...)
(gdb) p s->node[0]
$13 = (struct kmem_cache_node *) 0xffff888100040000
(gdb) p *s->node[0]
$14 = {
  list_lock = {
    {
      rlock = {
        raw_lock = {
          {
            val = {
              counter = 0
            },
            {
              locked = 0 '\000',
              pending = 0 '\000'
            },
            {
              locked_pending = 0,
              tail = 0
            }
          }
        }
      }
    }
  },
  nr_partial = 1,
  partial = {
    next = 0xffffea0004001008,
    prev = 0xffffea0004001008
  },
  nr_slabs = {
    counter = 1
  },
  total_objects = {
    counter = 64
  },
  full = {
    next = 0xffff888100040030,
    prev = 0xffff888100040030
  }
}
```

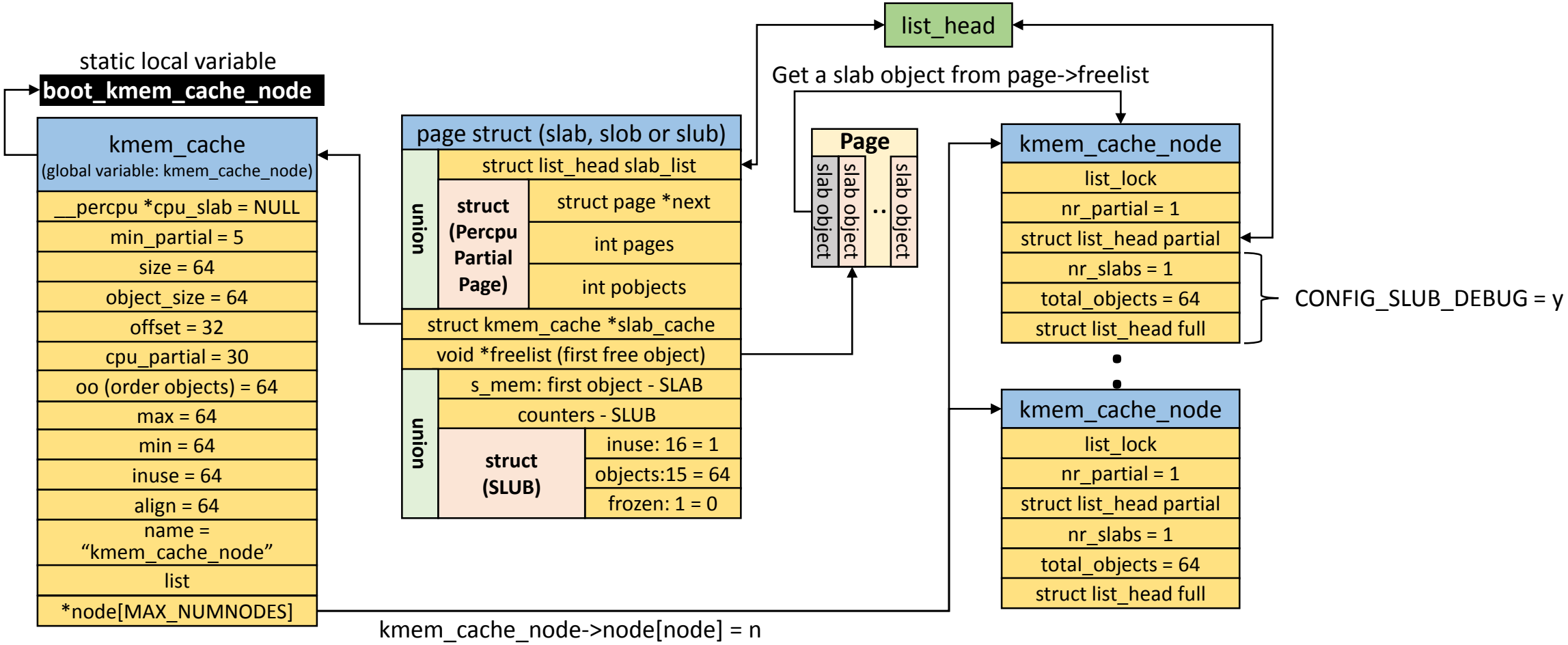


early_kmem_cache_node_alloc(): configure node #0

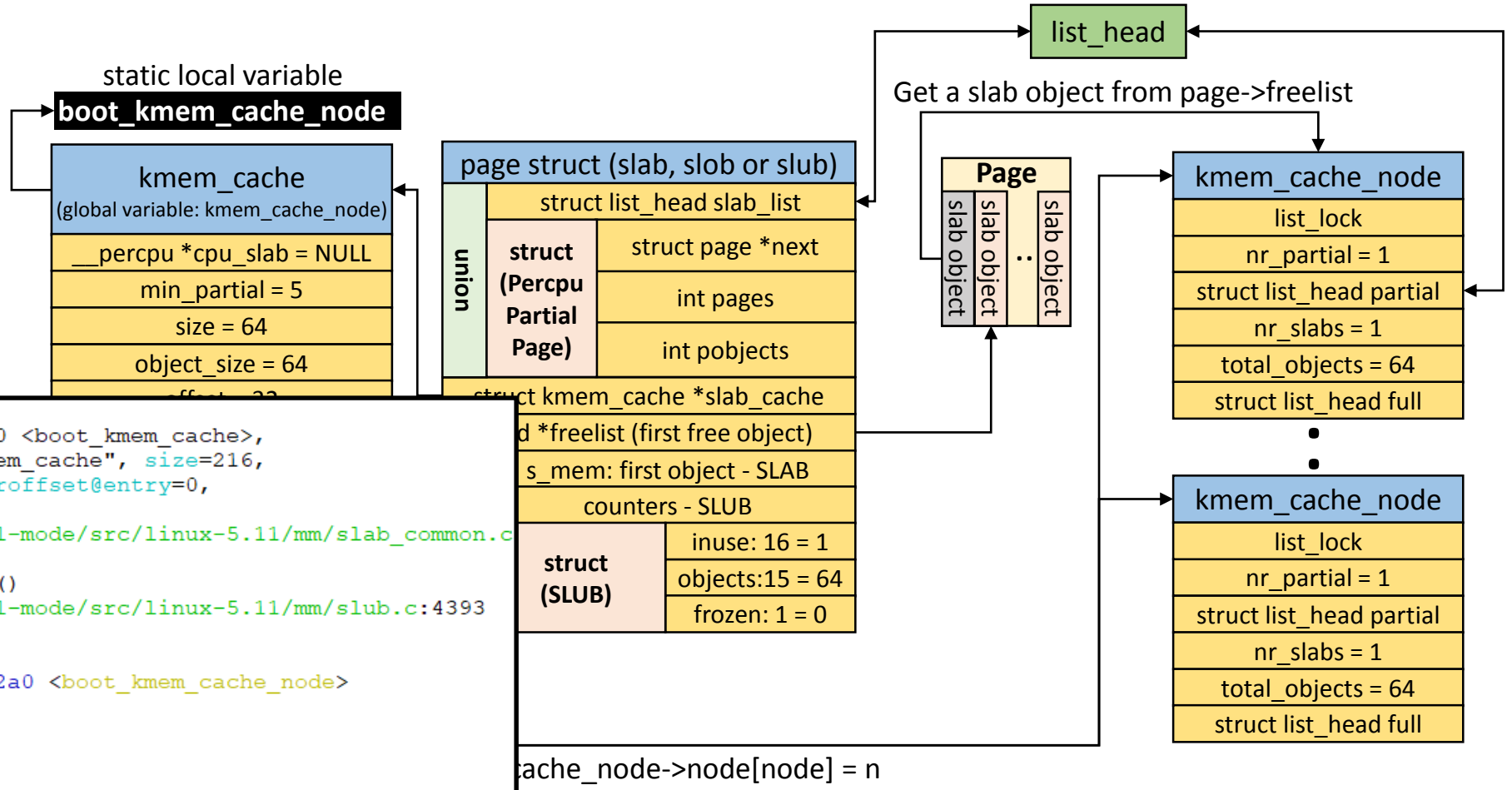



```
(gdb) p s->node[0]->partial
$23 = {
  next = 0xffffea0004001008,
  prev = 0xffffea0004001008
}
(gdb) p ((struct page *) 0xffffea0004001000)->slab_cache
$24 = (struct kmem_cache *) 0xffffffff81beb2a0 <boot_kmem_cache_node>
(gdb) p ((struct page *) 0xffffea0004001000)->freelist
$25 = (void *) 0xffff888100040040
(gdb) x /g 0xffff888100040060
0xffff888100040060:     0xffff888100040080
```

early_kmem_cache_node_alloc(): Fully-configured

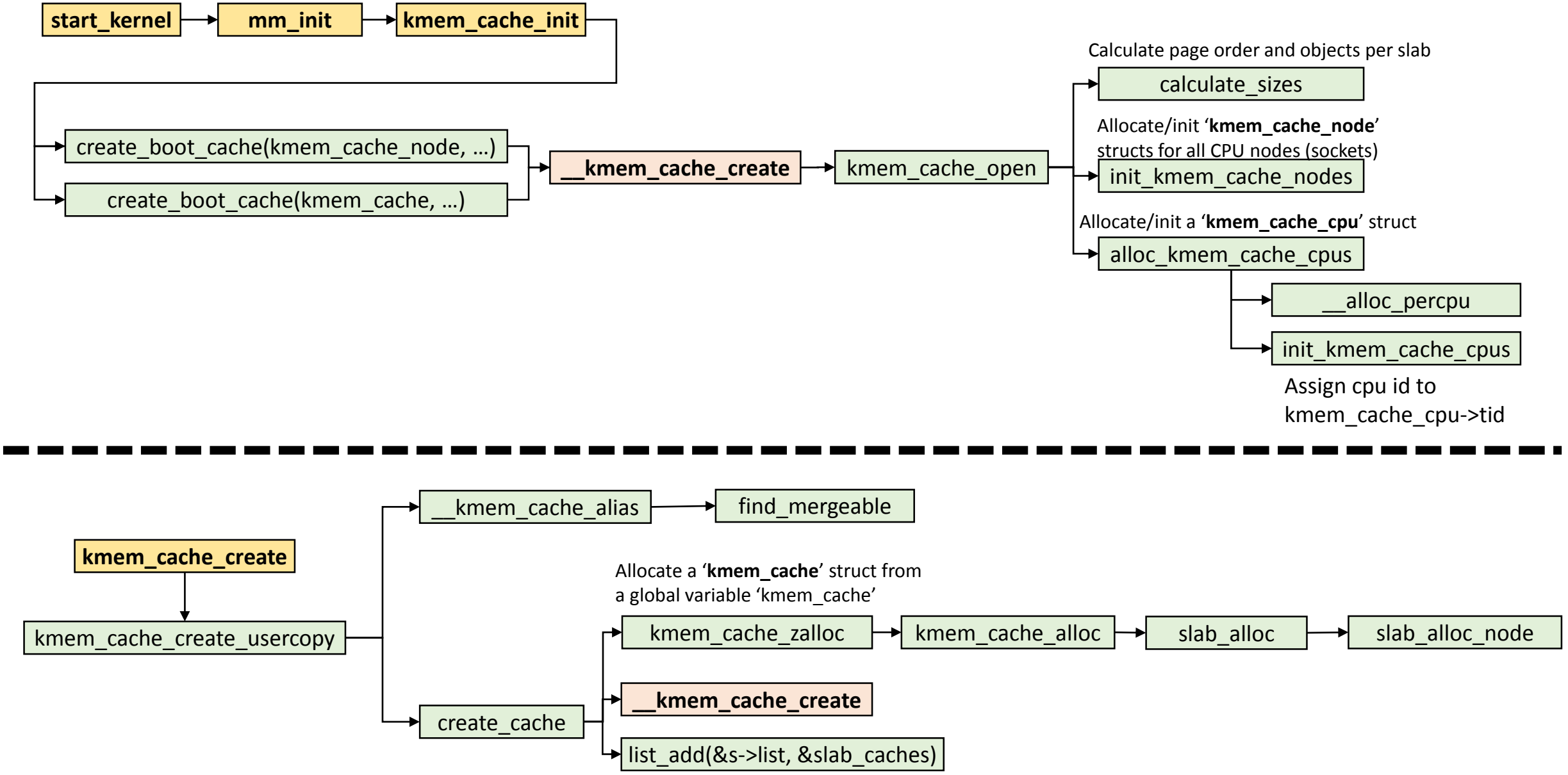


early_kmem_cache_node_alloc(): Fully-configured




```
(gdb) bt 2
#0  create_boot_cache (s=0xffffffff81bed380 <boot_kmem_cache>,
    name=name@entry=0xffffffff819186ec "kmem_cache", size=216,
    flags=flags@entry=8192, useroffset=useroffset@entry=0,
    usersize=usersize@entry=0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slab_common.c
:549
#1  0xffffffff81b1f48e in kmem_cache_init ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:4393
(More stack frames follow...)
(gdb) p kmem_cache_node
$9 = (struct kmem_cache *) 0xffffffff81beb2a0 <boot_kmem_cache_node>
(gdb) p kmem_cache_node->node[0]->partial
$10 = {
  next = 0xffffea0004001008,
  prev = 0xffffea0004001008
}
(gdb)  p ((struct page *) 0xffffea0004001000)->freelist
$11 = (void *) 0xffff888100040040
(gdb) p kmem_cache_node->node[1]->partial
$12 = {
  next = 0xffffea0009000008,
  prev = 0xffffea0009000008
}
(gdb)  p ((struct page *) 0xffffea0009000000)->freelist
$13 = (void *) 0xffff888240000040
```

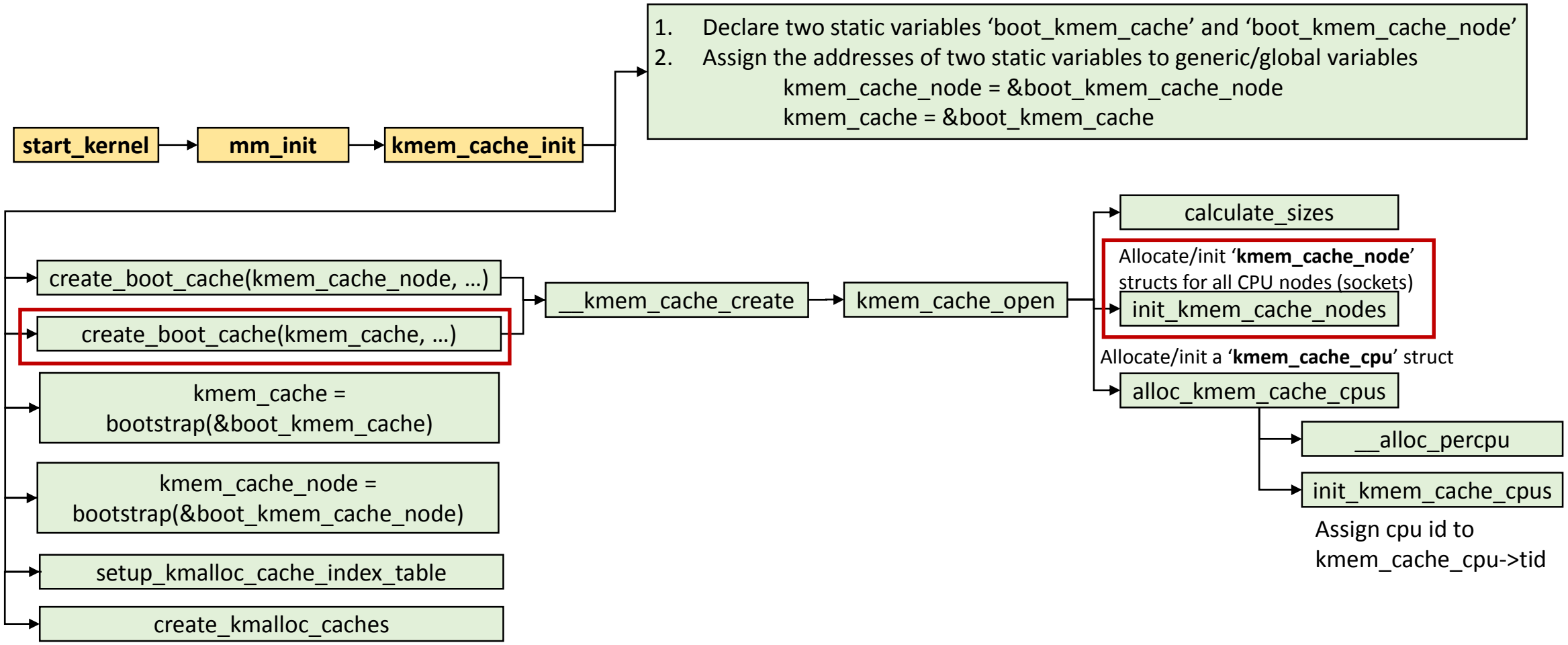
[Comparison] Generic kmem_cache_create() & low-level implementation __kmem_cache_create()



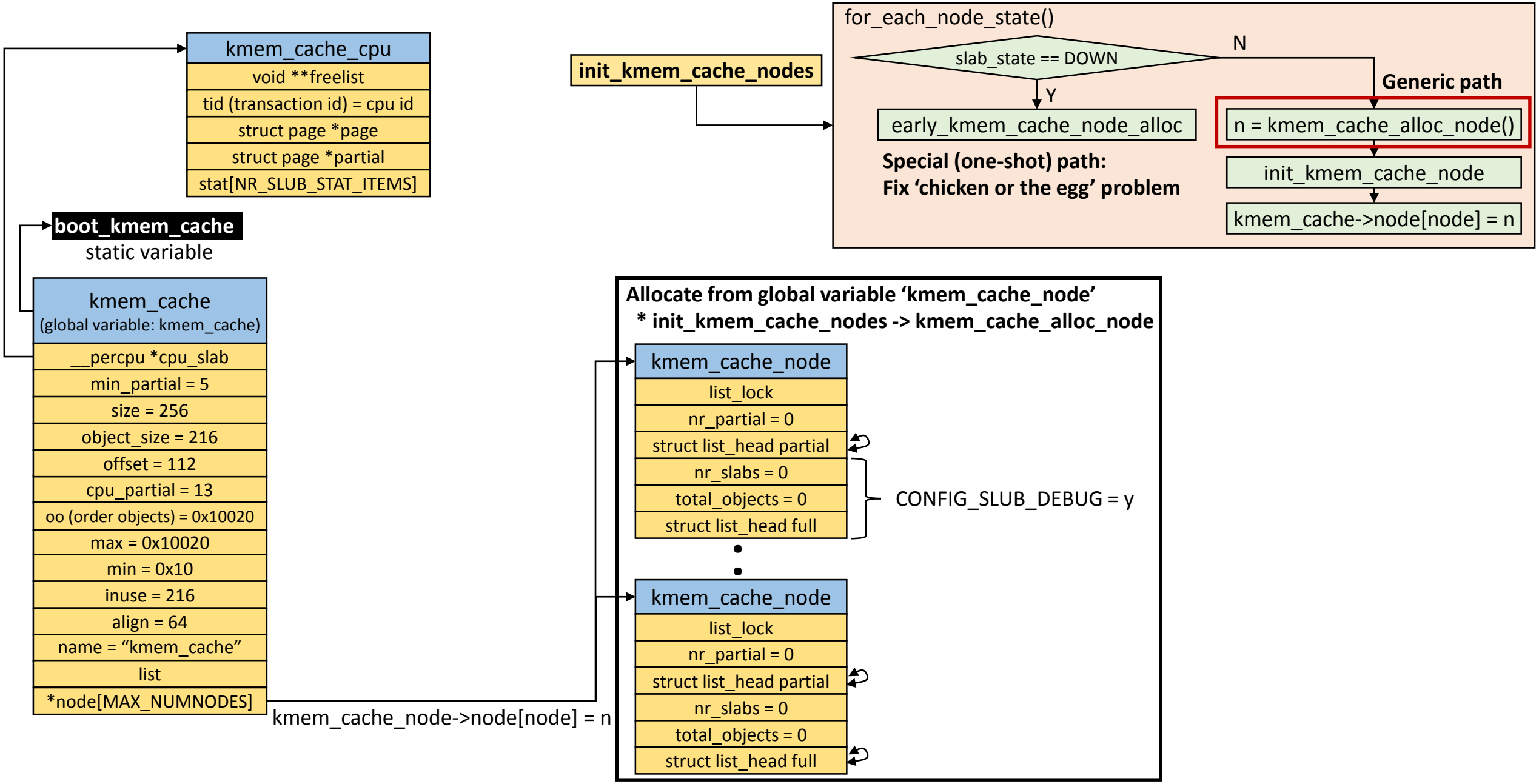
[Cache allocation] slab_alloc_node – Allocate a slab cache with specific node id

- Fast path & slow path
  - Will follow kmem_cache_init() flow

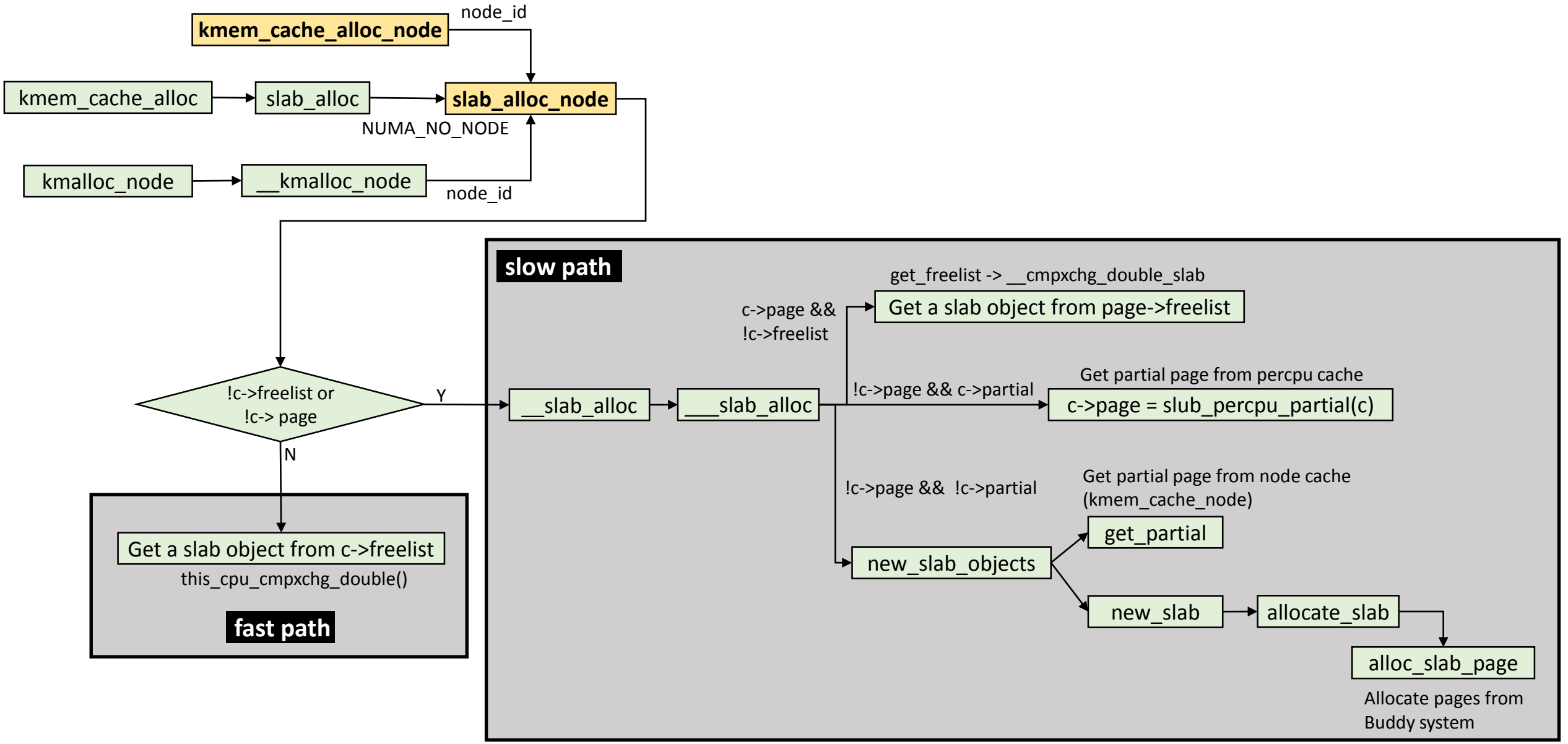
kmem_cache_init()



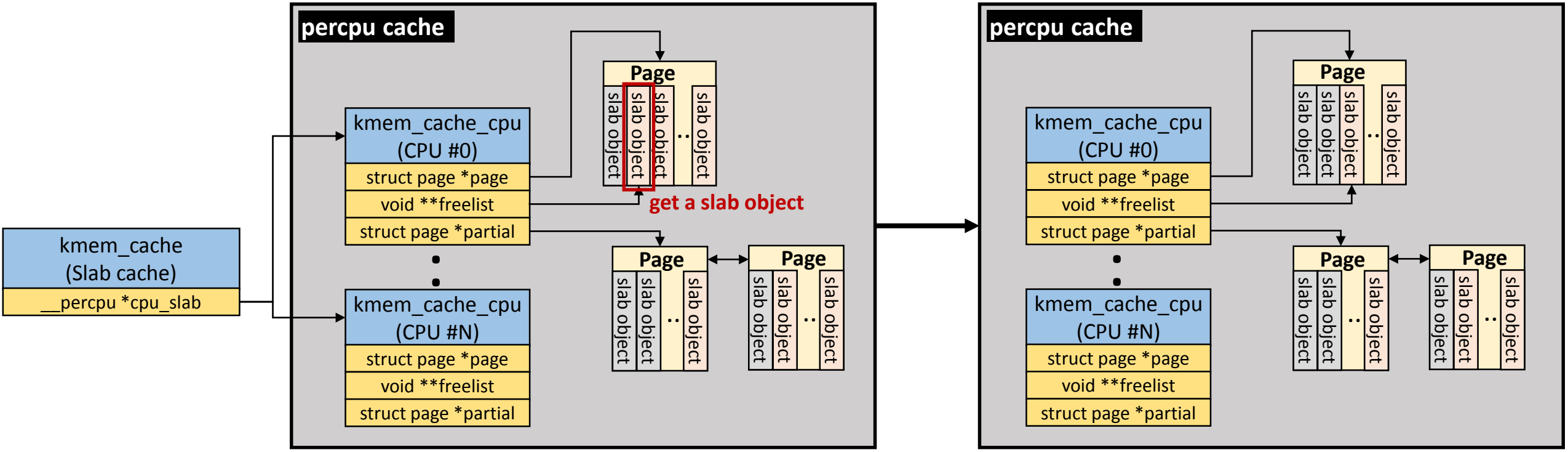
kmem_cache: init_kmem_cache_nodes()



kmem_cache_alloc_node/slab_alloc_node

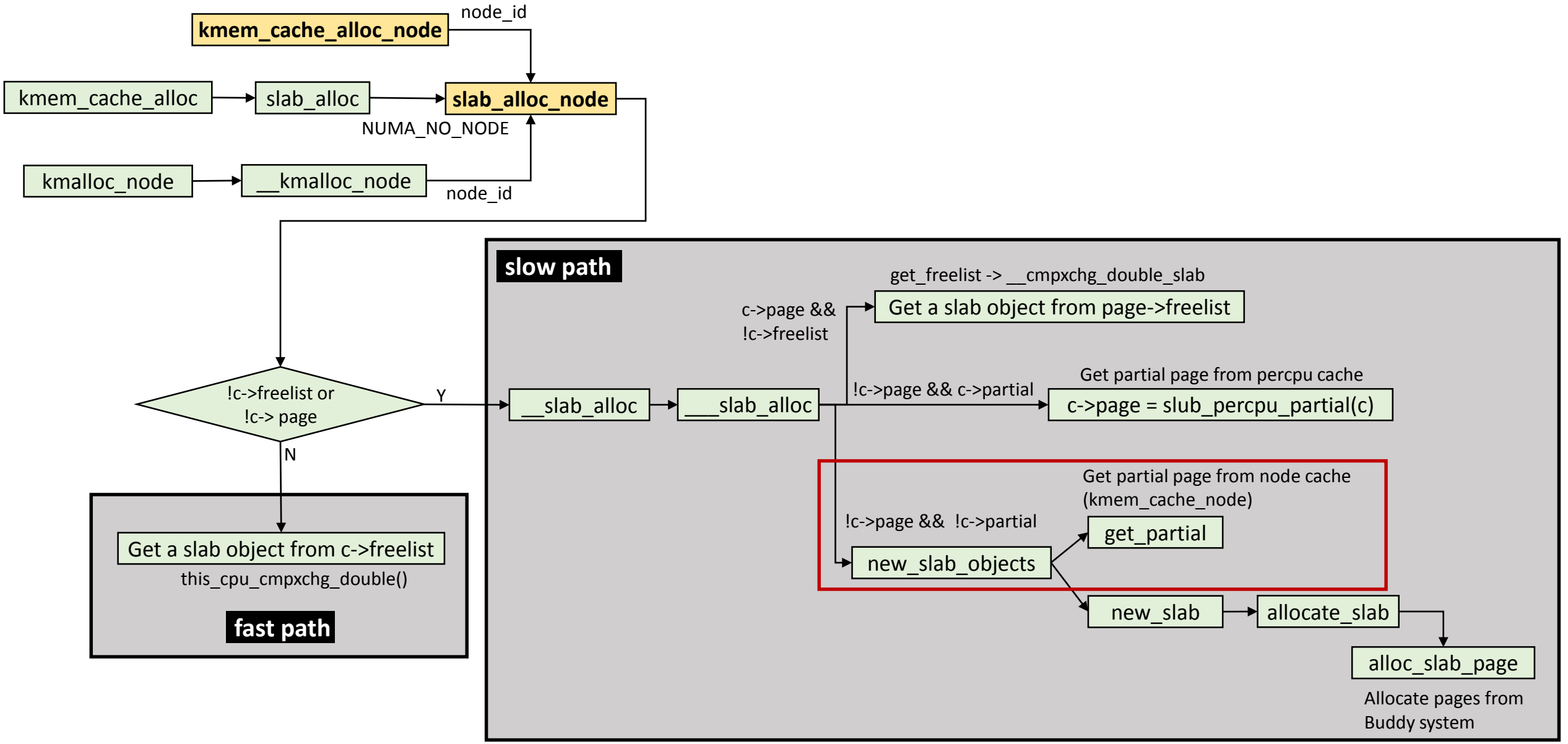


Fast Path

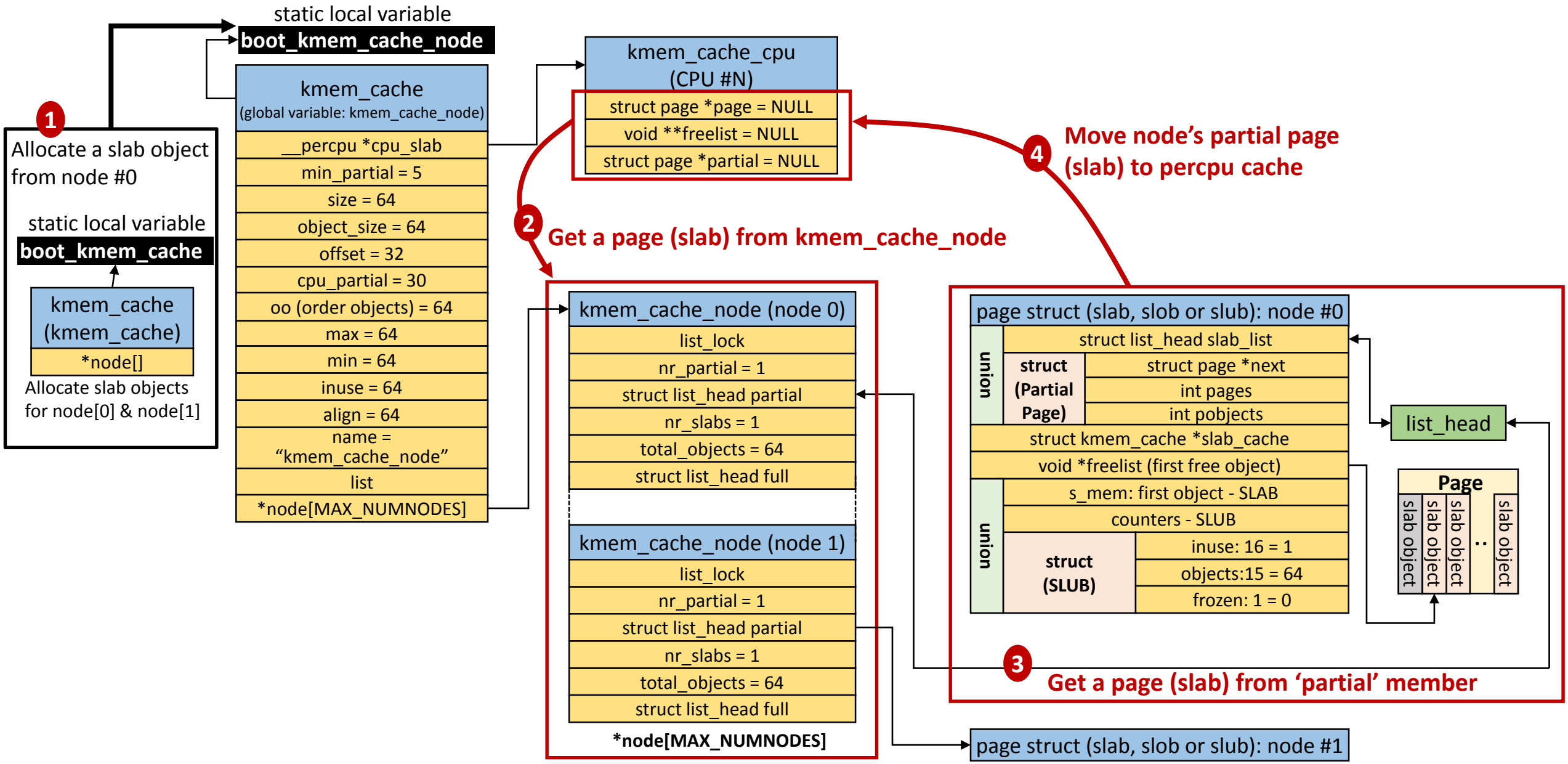


Get a slab object from kmem_cache_cpu->freelist

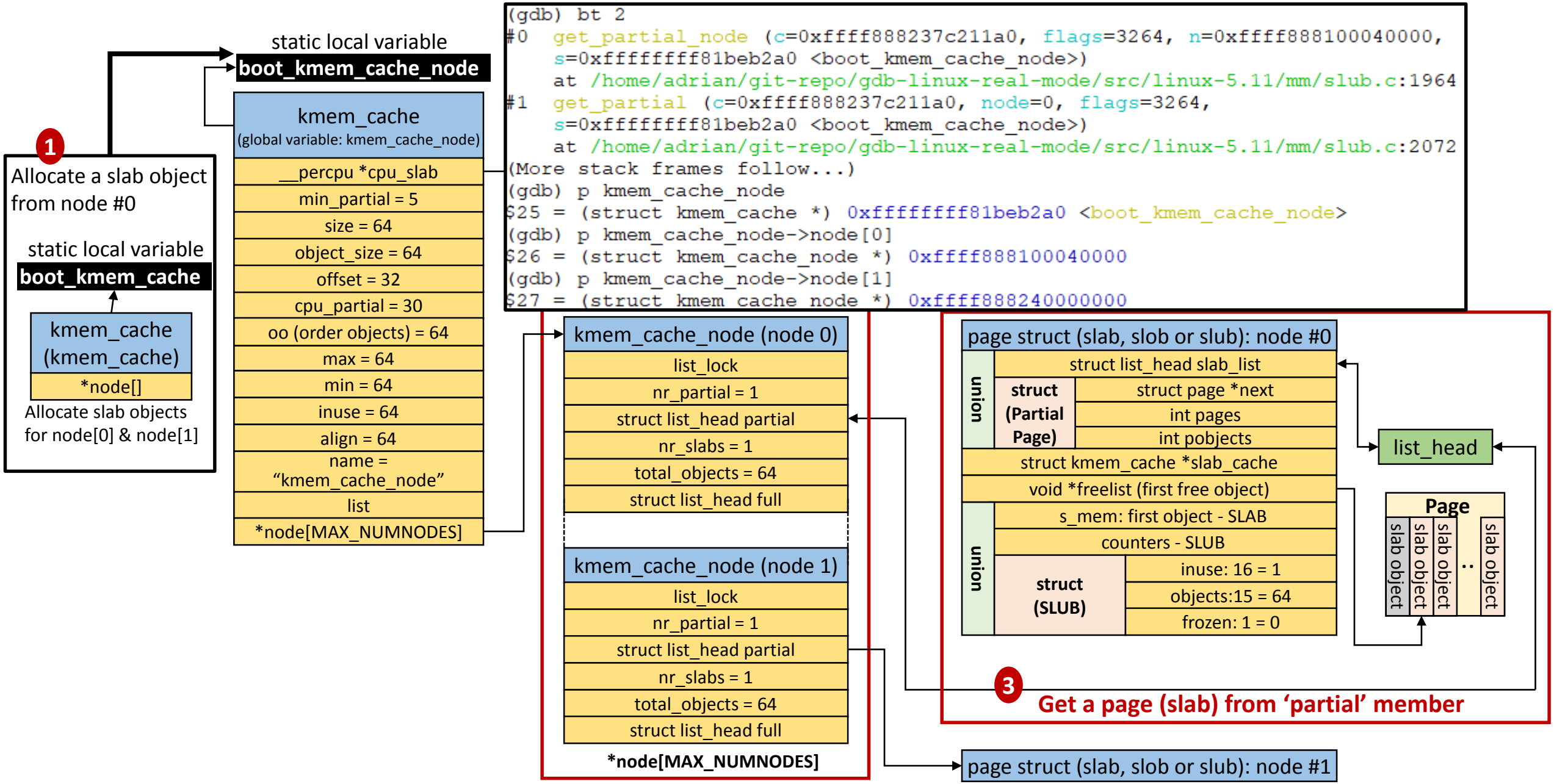
slab_alloc_node(): slowpath #1 – Get an object from node's partial page



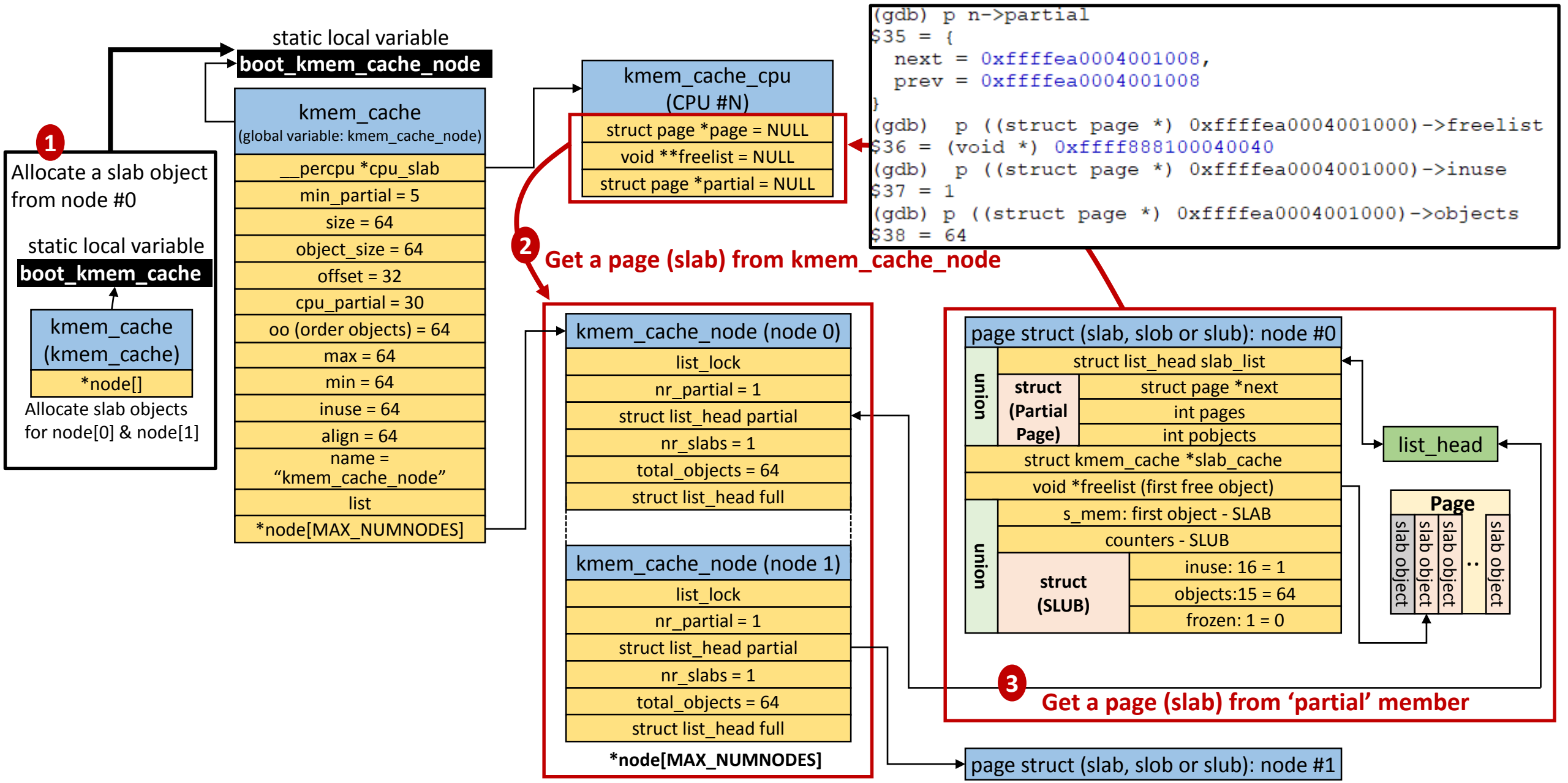
Slowpath #1 – Get an object from node's partial page



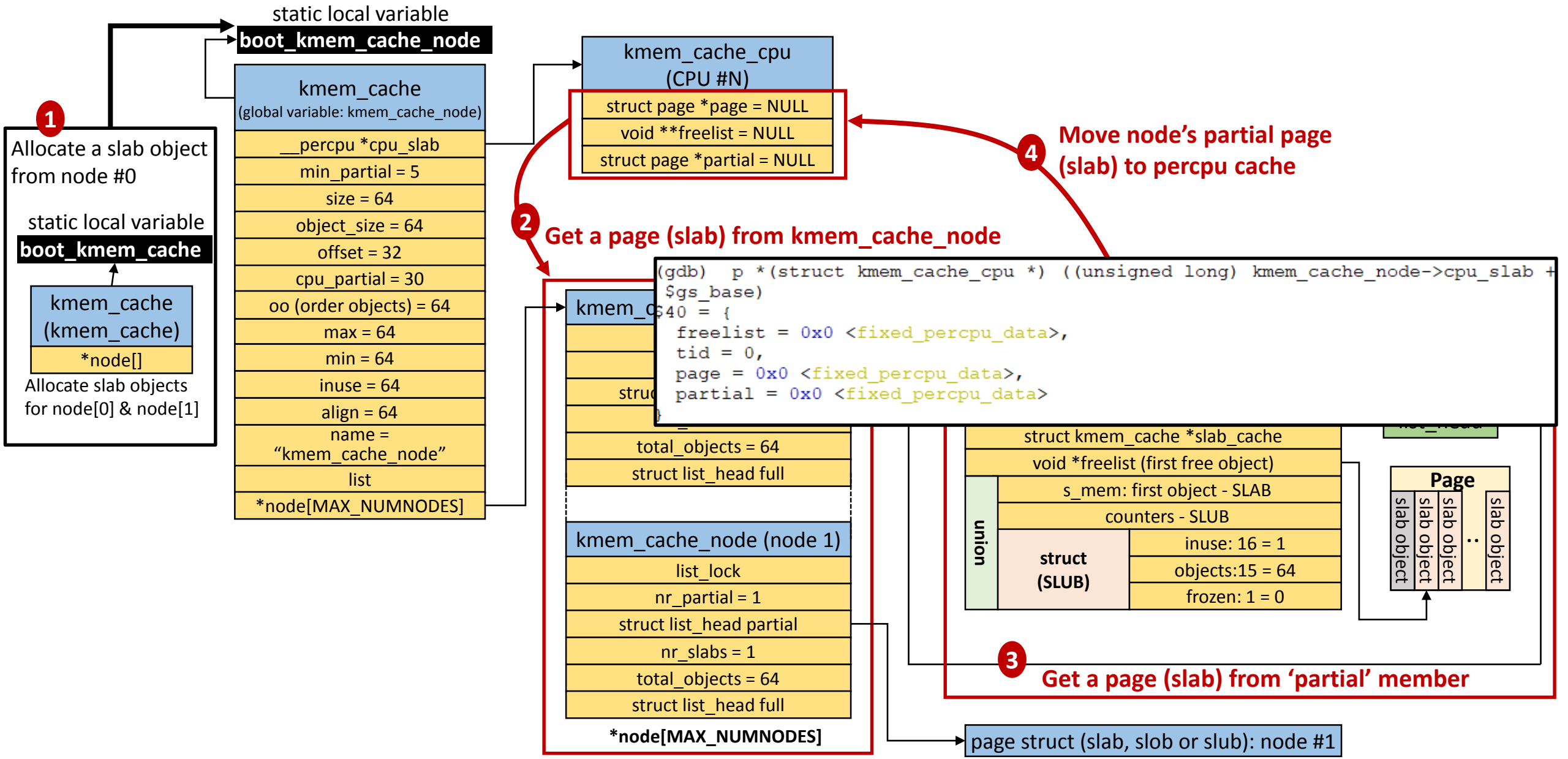
Slowpath #1 – Get an object from node's partial page



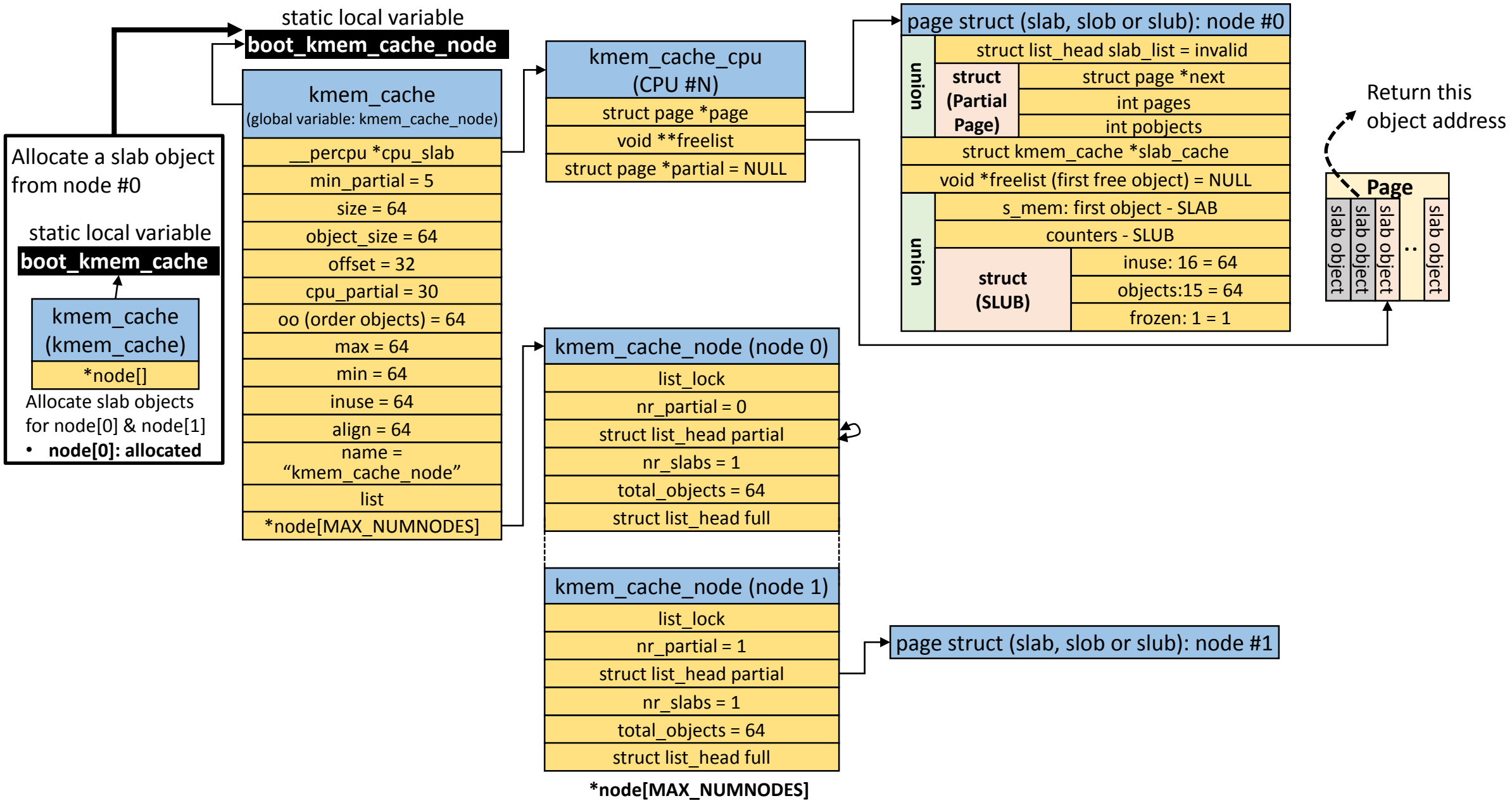
Slowpath #1 – Get an object from node's partial page



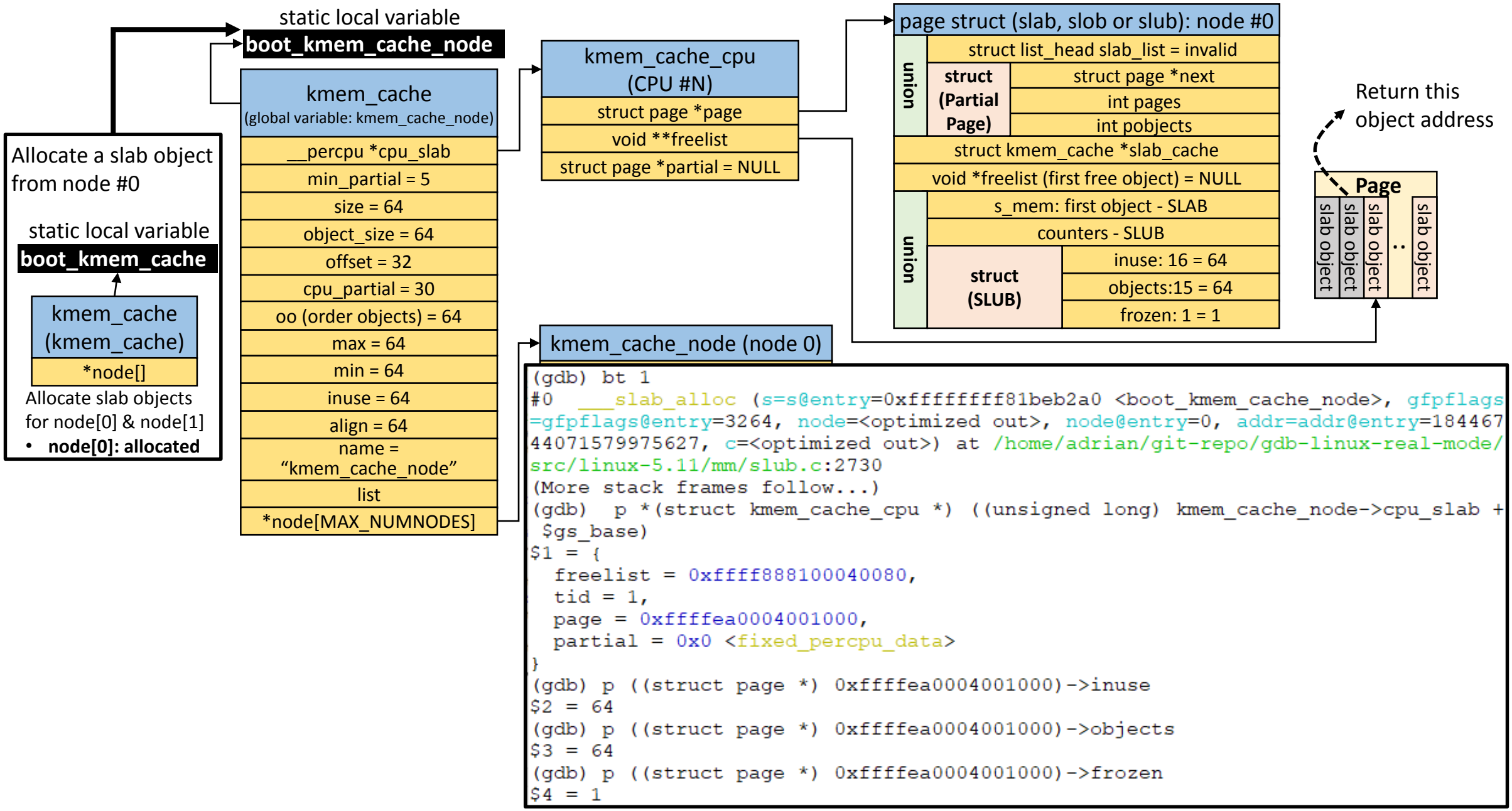
Slowpath #1 – Get an object from node's partial page



Slowpath #1 – Get an object from node's partial page

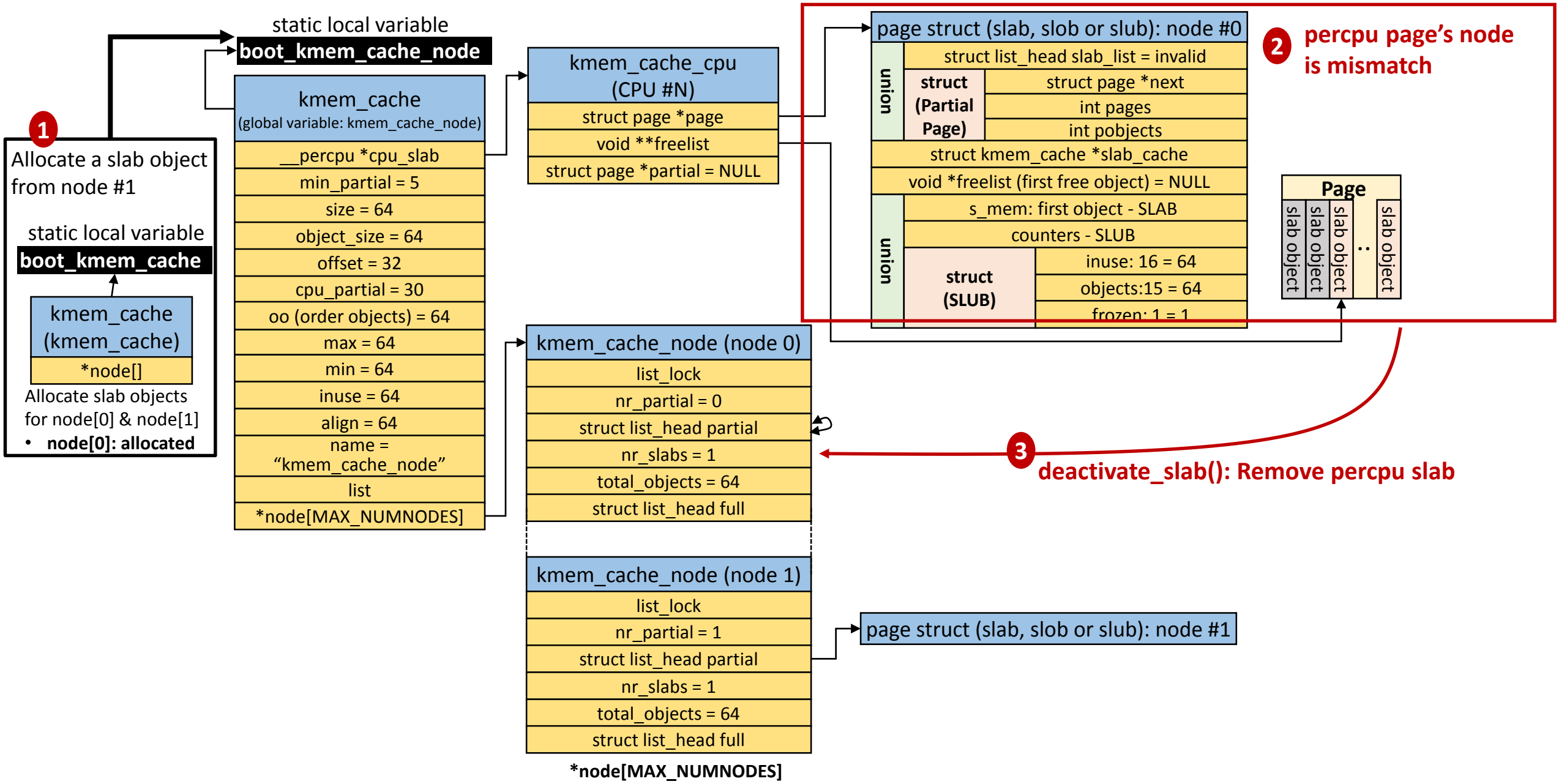


Slowpath #1 – Get an object from node's partial page

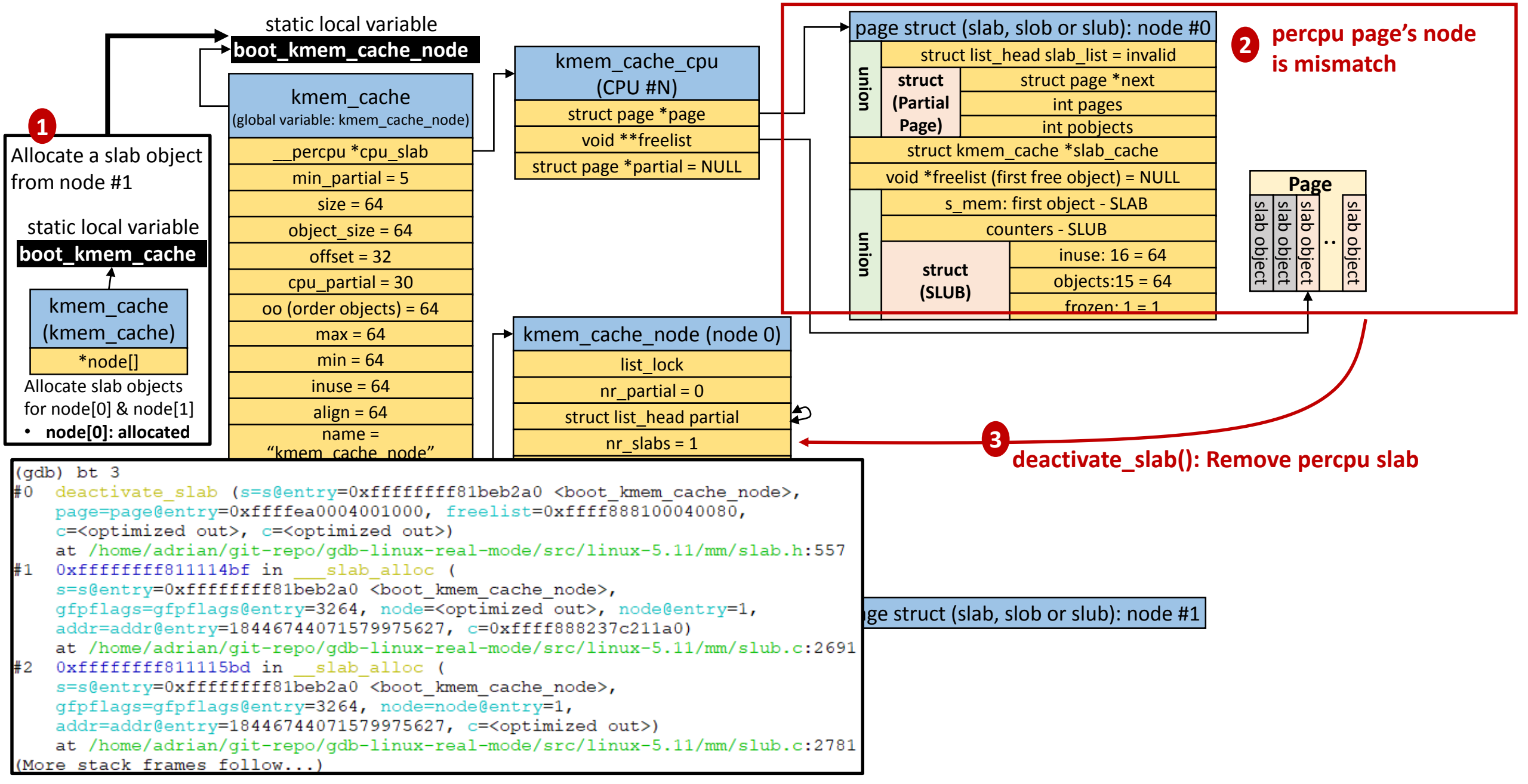



```
(gdb) bt 1
#0  ___slab_alloc (s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>, gfpflags
=gfpflags@entry=3264, node=<optimized out>, node@entry=0, addr=addr@entry=184467
44071579975627, c=<optimized out>) at /home/adrian/git-repo/gdb-linux-real-mode/
src/linux-5.11/mm/slub.c:2730
(More stack frames follow...)
(gdb)  p *(struct kmem_cache_cpu *) ((unsigned long) kmem_cache_node->cpu_slab +
 $gs_base)
$1 = {
  freelist = 0xffff888100040080,
  tid = 1,
  page = 0xffffea0004001000,
  partial = 0x0 <fixed_percpu_data>
}
(gdb) p ((struct page *) 0xffffea0004001000)->inuse
$2 = 64
(gdb) p ((struct page *) 0xffffea0004001000)->objects
$3 = 64
(gdb) p ((struct page *) 0xffffea0004001000)->frozen
$4 = 1
```

Slowpath #1 – Get an object from node's partial page

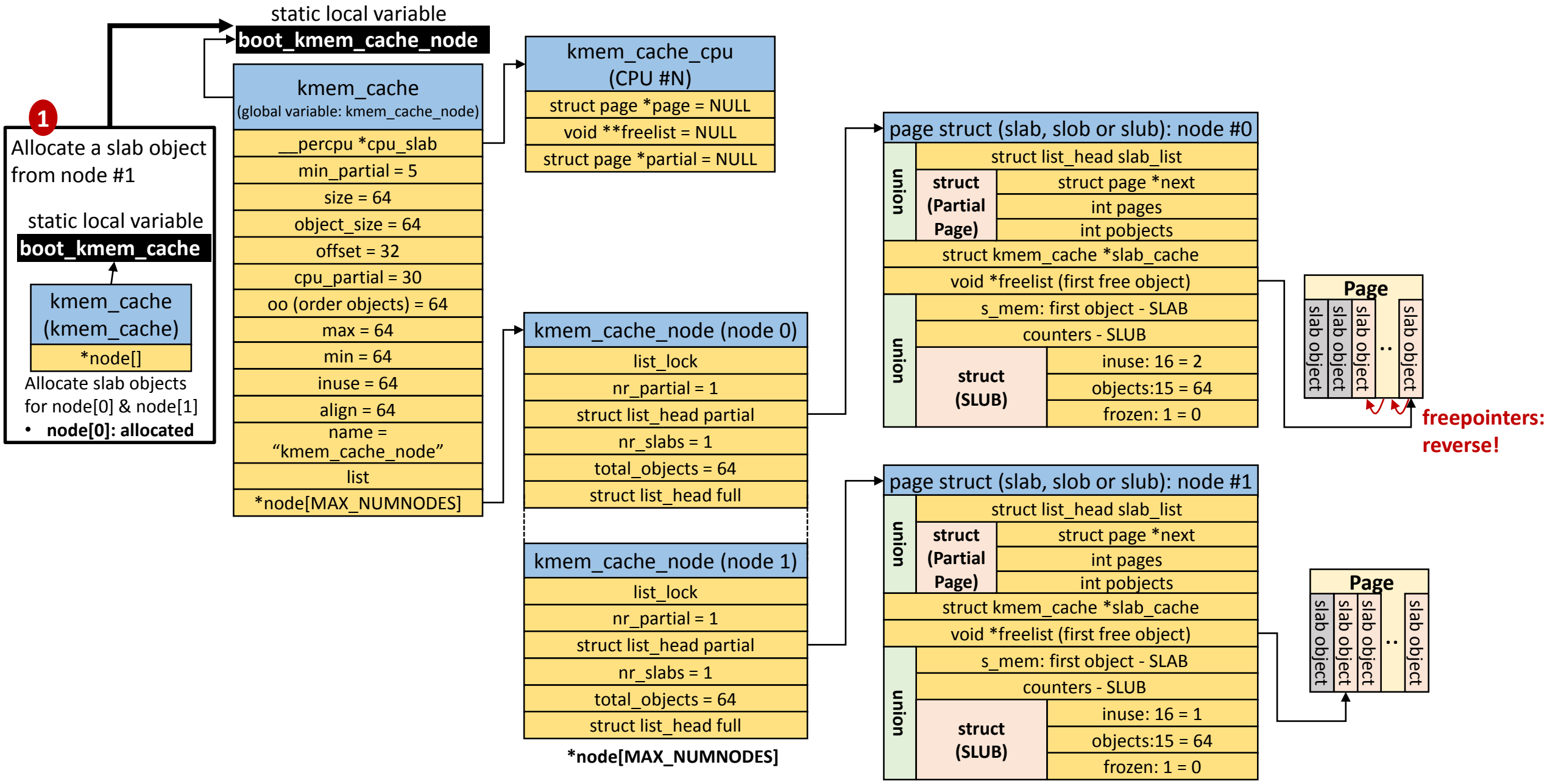


Slowpath #1 – Get an object from node's partial page

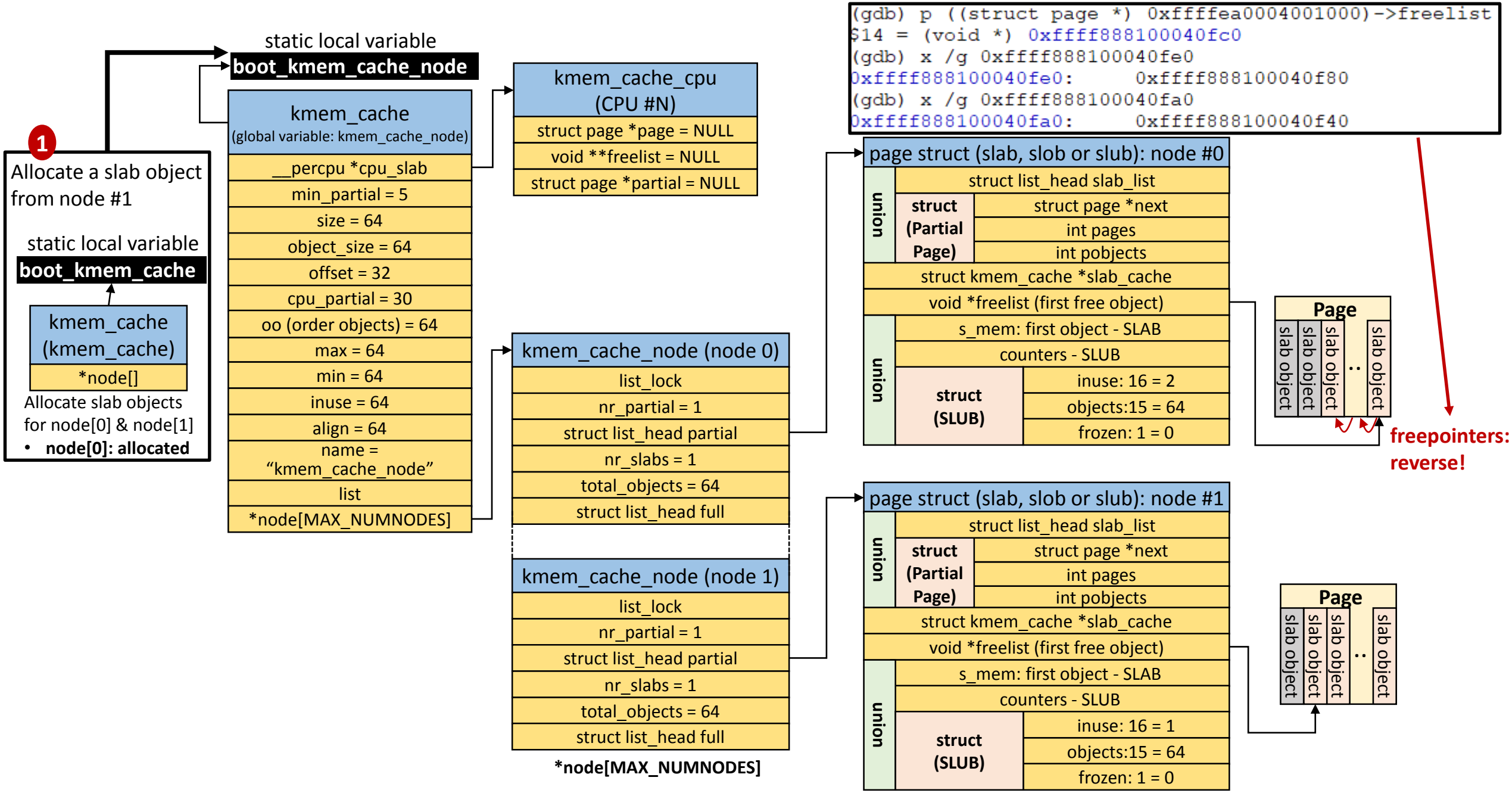



```
(gdb) bt 3
#0  deactivate_slab (s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>,
    page=page@entry=0xffffea0004001000, freelist=0xffff888100040080,
    c=<optimized out>, c=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slab.h:557
#1  0xffffffff811114bf in ___slab_alloc (
    s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>,
    gfpflags=gfpflags@entry=3264, node=<optimized out>, node@entry=1,
    addr=addr@entry=18446744071579975627, c=0xffff888237c211a0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2691
#2  0xffffffff811115bd in __slab_alloc (
    s=s@entry=0xffffffff81beb2a0 <boot_kmem_cache_node>,
    gfpflags=gfpflags@entry=3264, node=node@entry=1,
    addr=addr@entry=18446744071579975627, c=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2781
(More stack frames follow...)
```

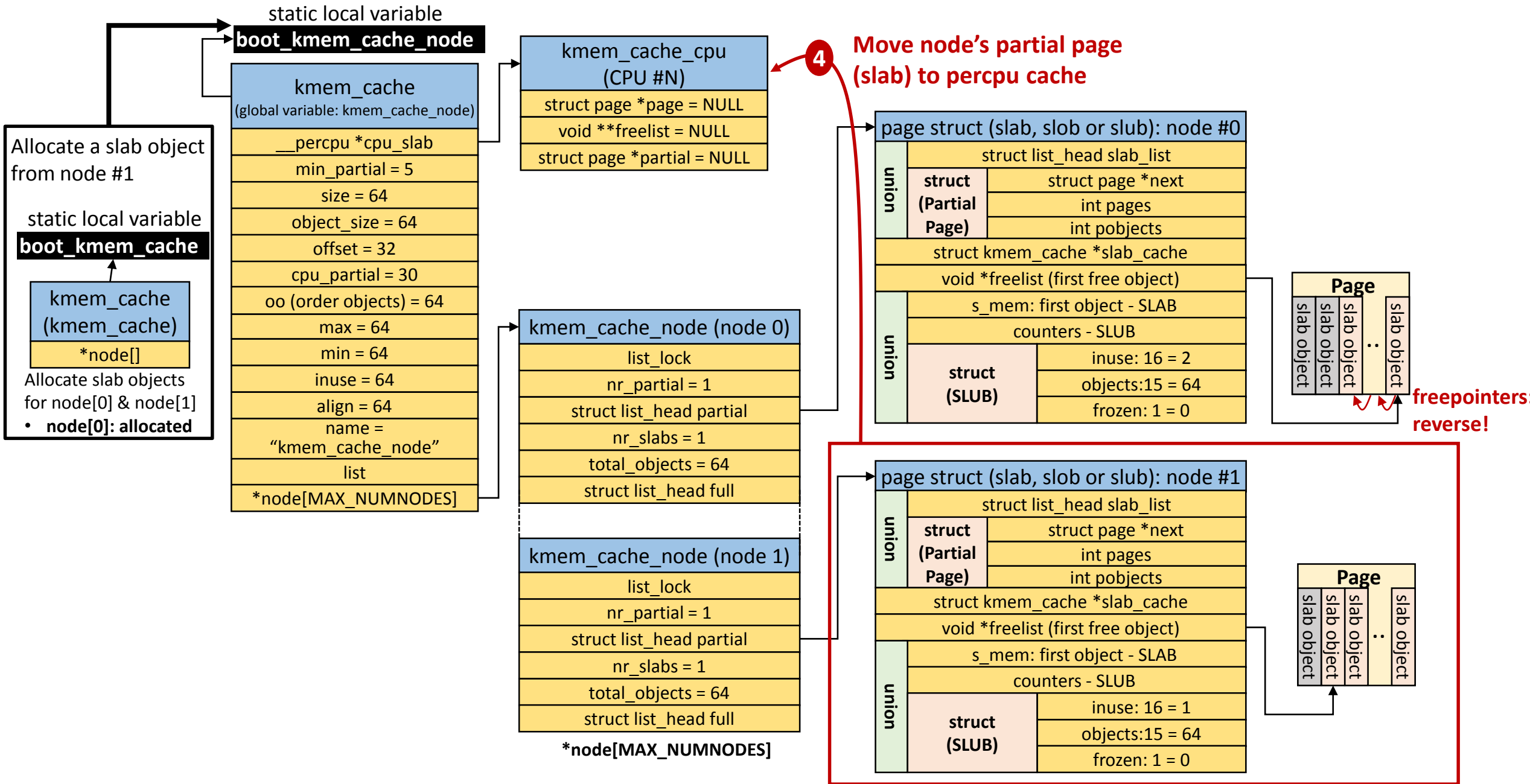
Slowpath #1 – Get an object from node's partial page



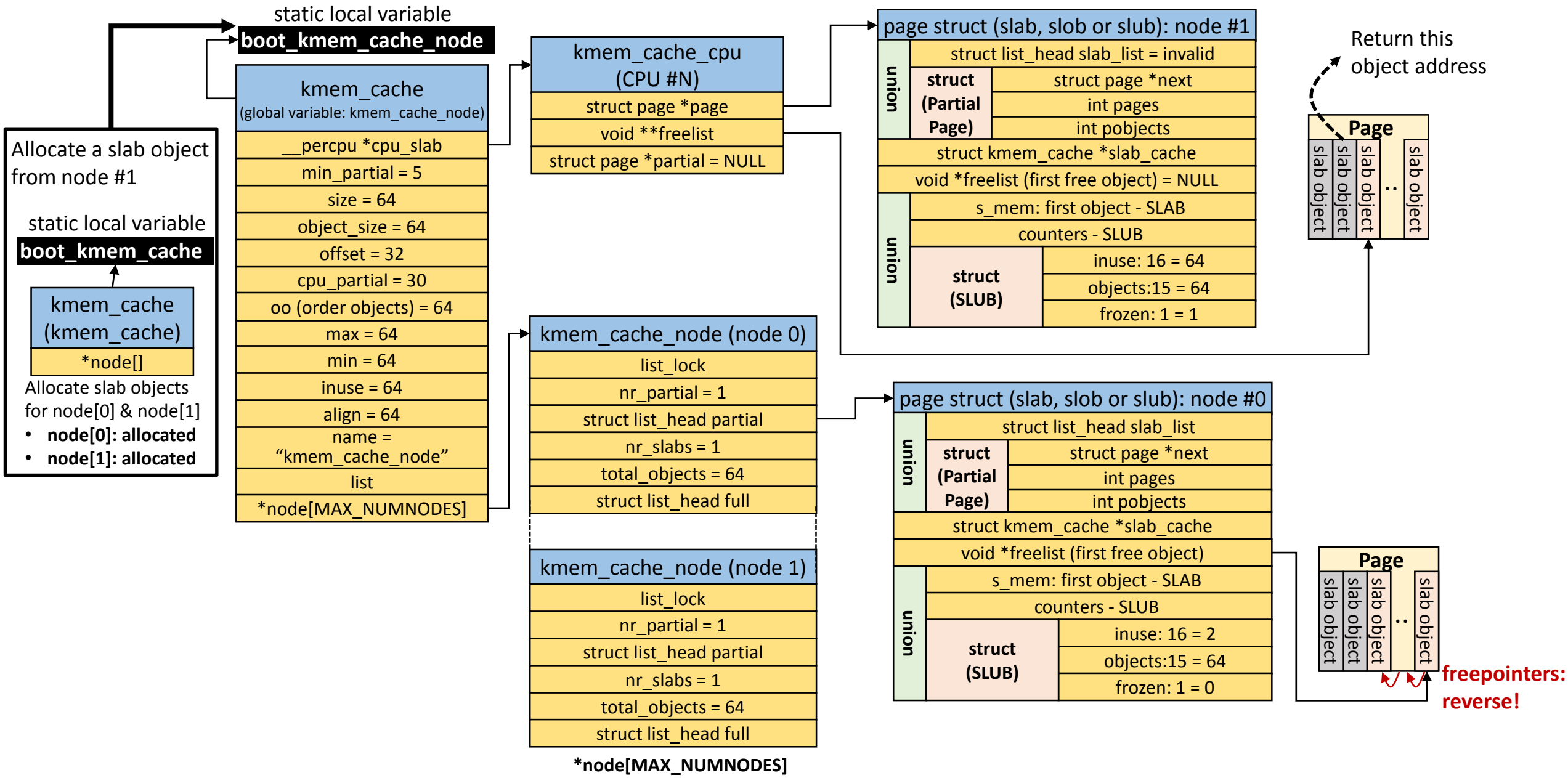
Slowpath #1 – Get an object from node's partial page



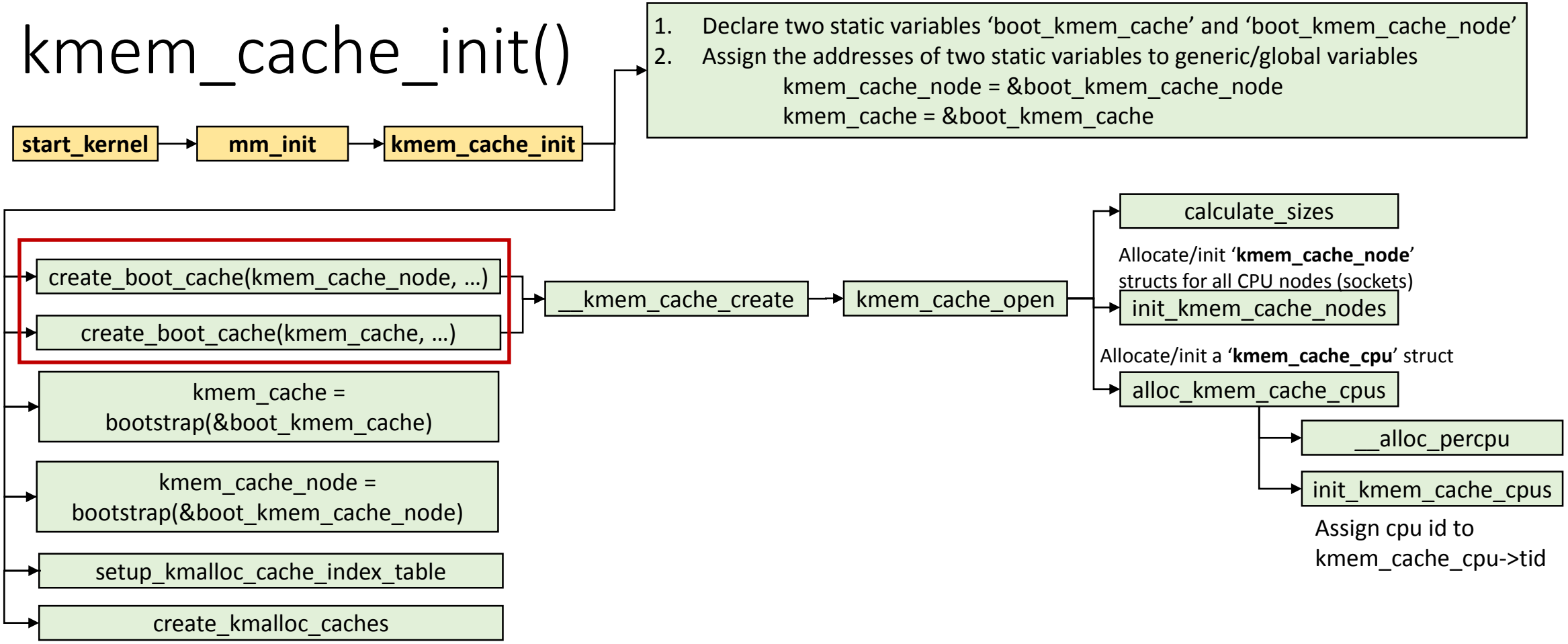
Slowpath #1 – Get an object from node's partial page



Slowpath #1 – Get an object from node's partial page

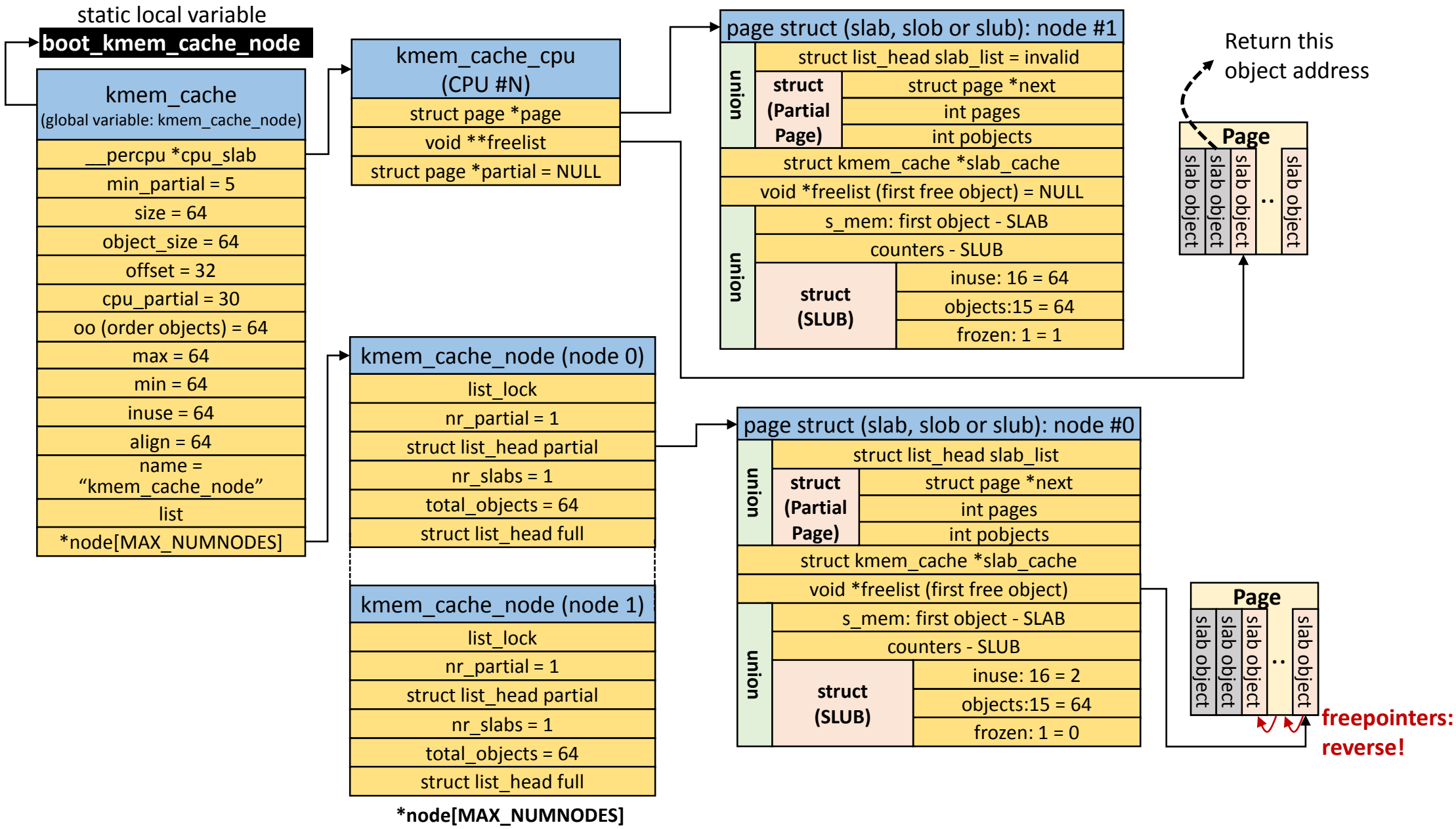


kmem_cache_init()

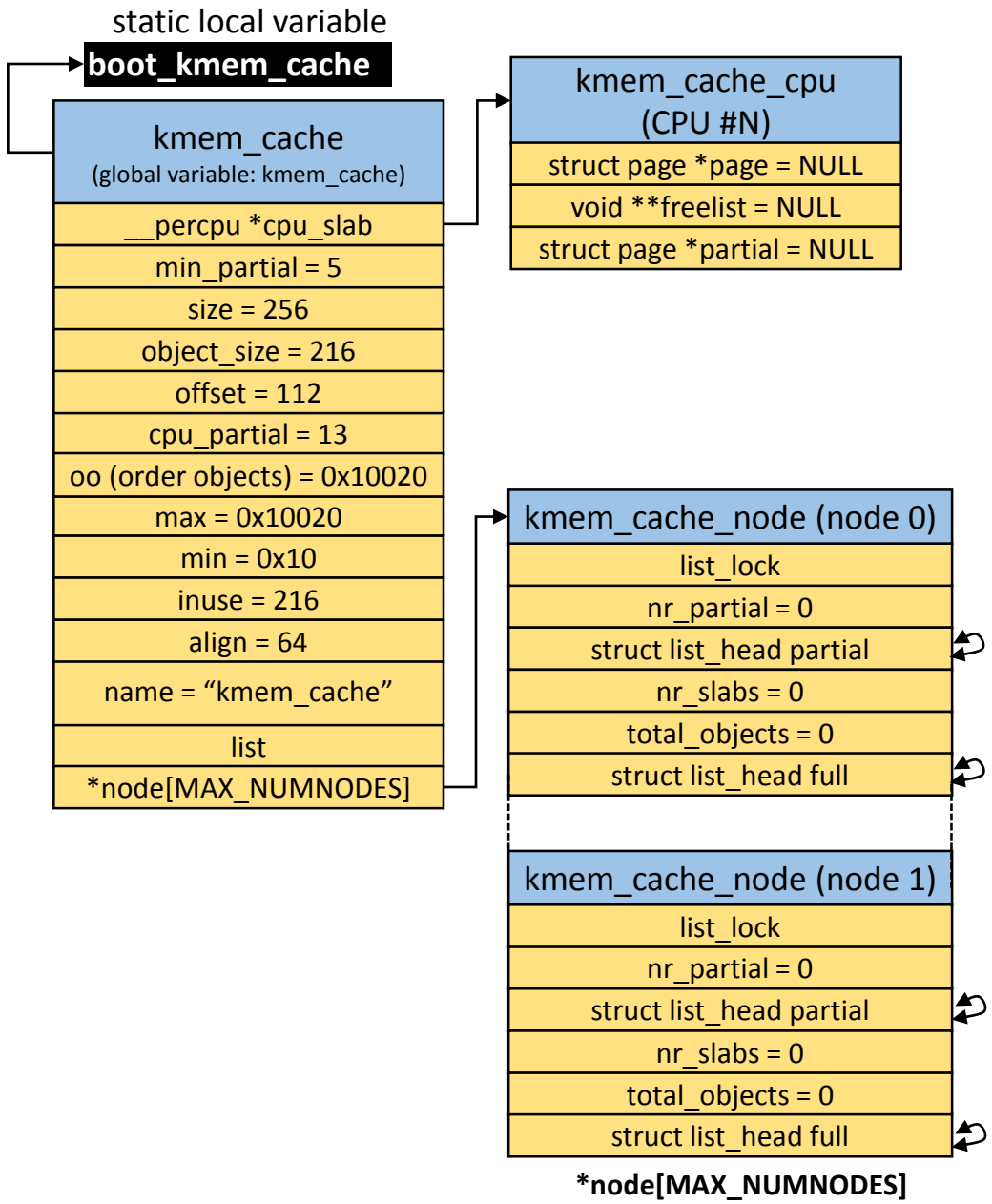


Let's check data structures about kmem_cache & kmem_cache_node after function call "create_boot_cache"

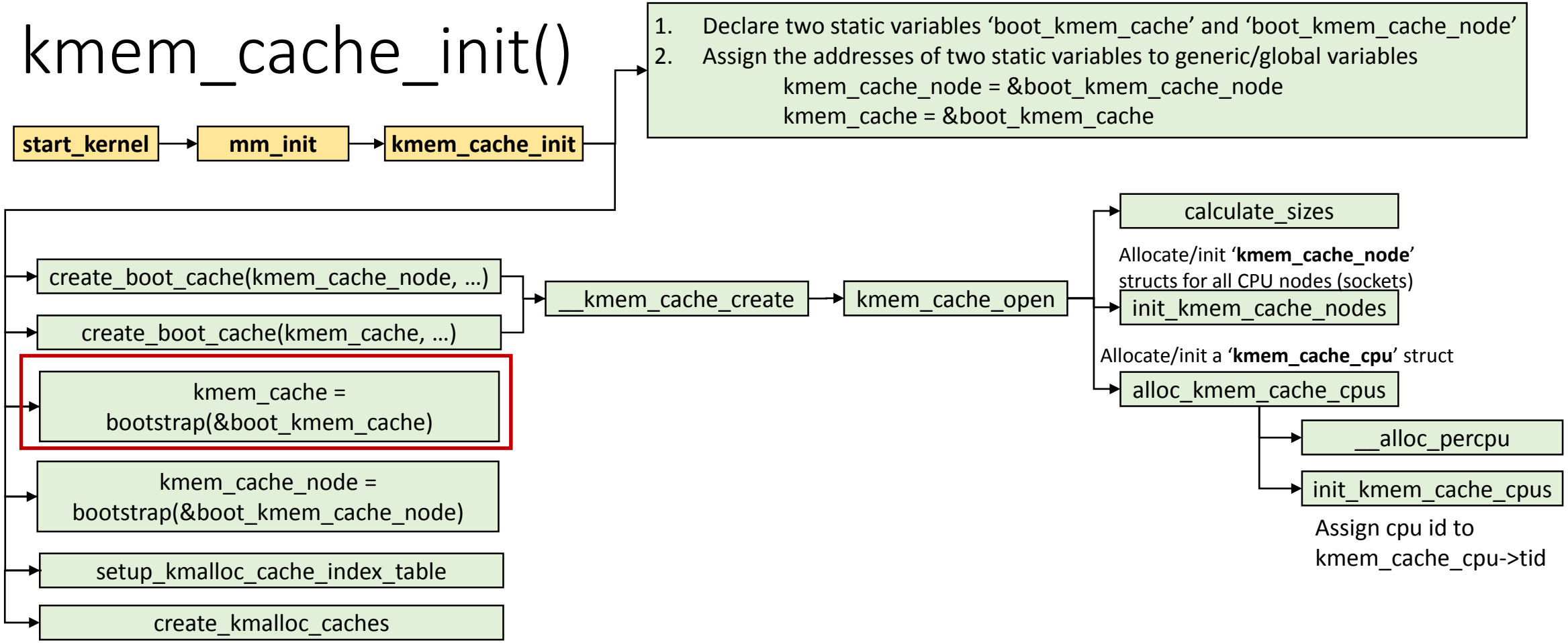
create_boot_cache(): 1/2



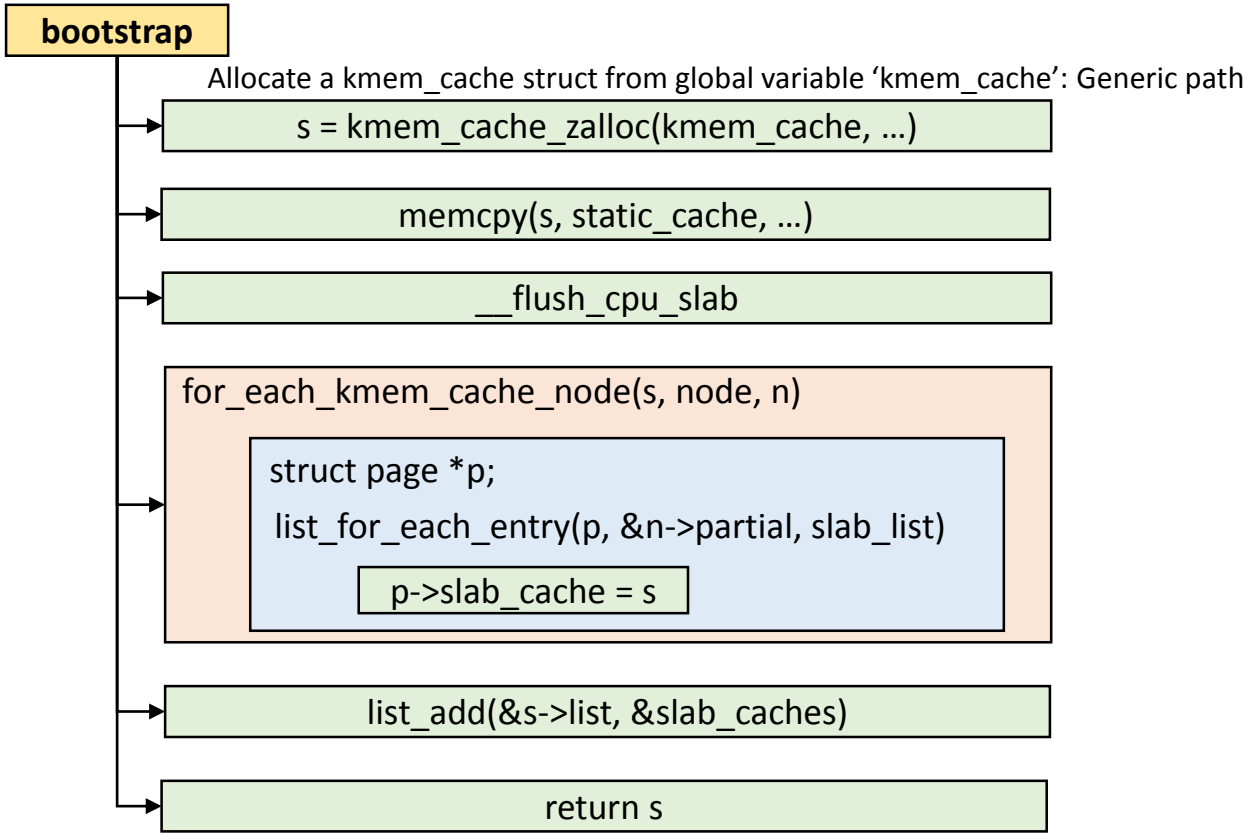
create_boot_cache(): 2/2



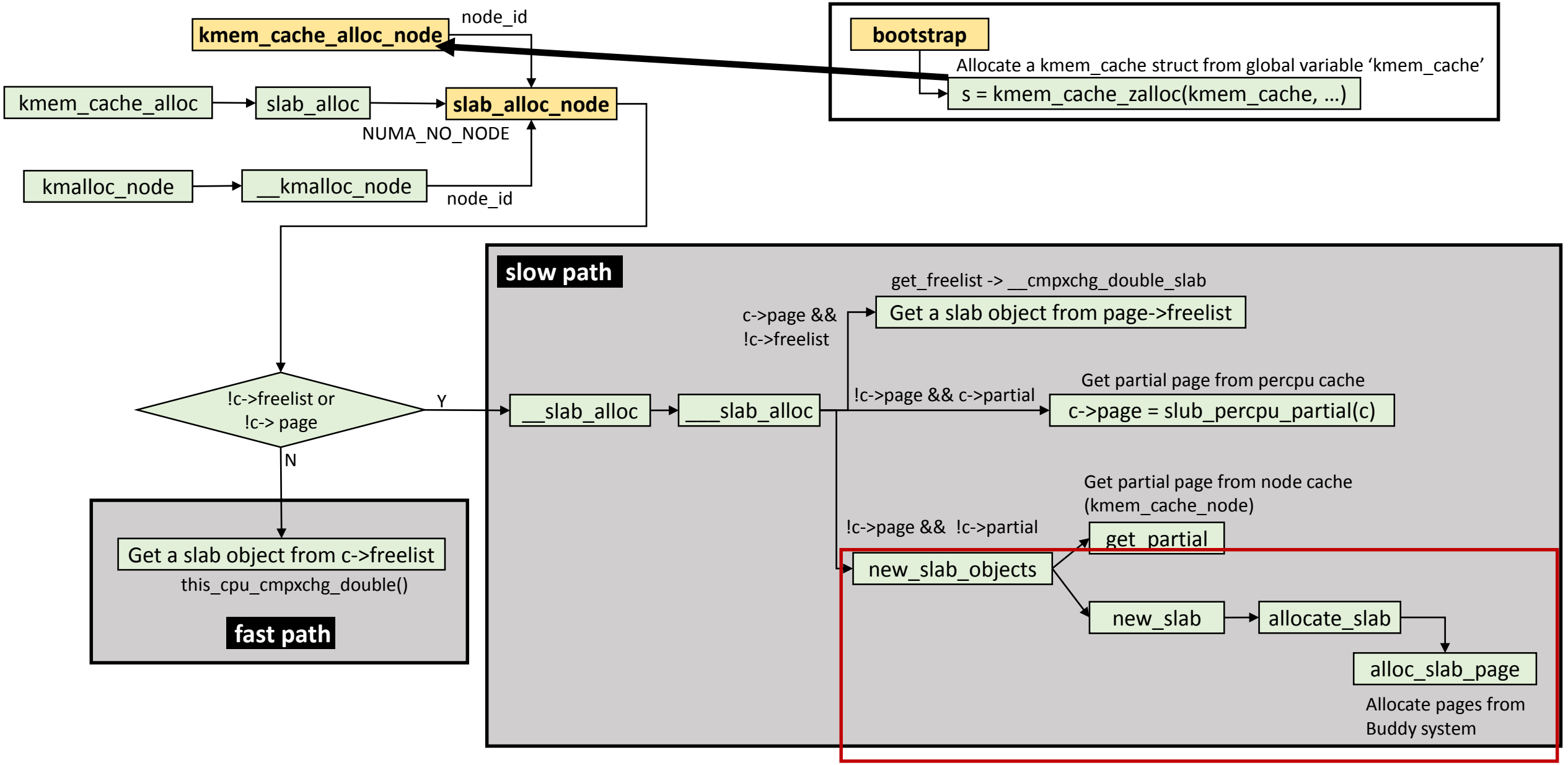
kmem_cache_init()



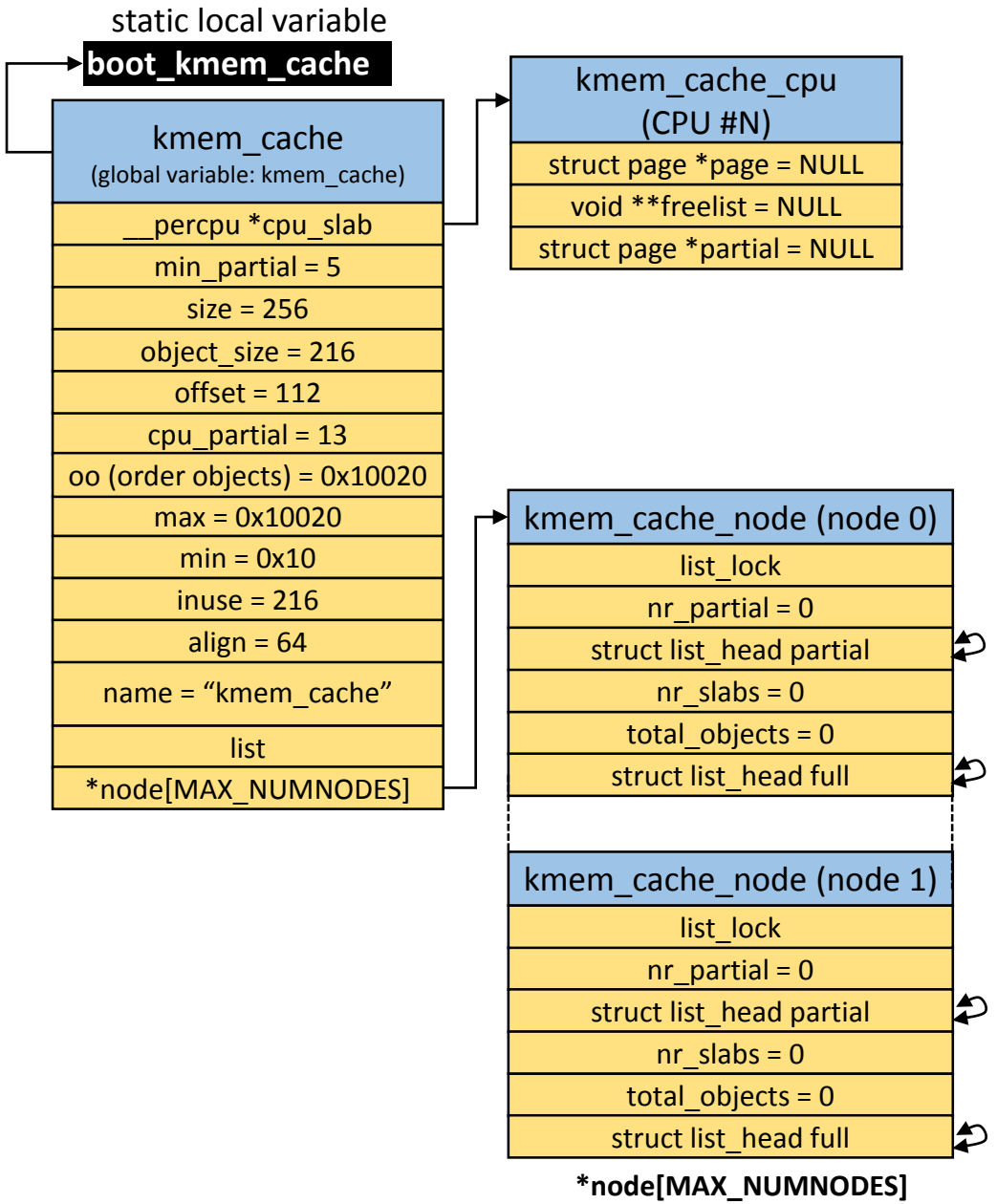
kmem_cache = bootstrap(&boot_kmem_cache)



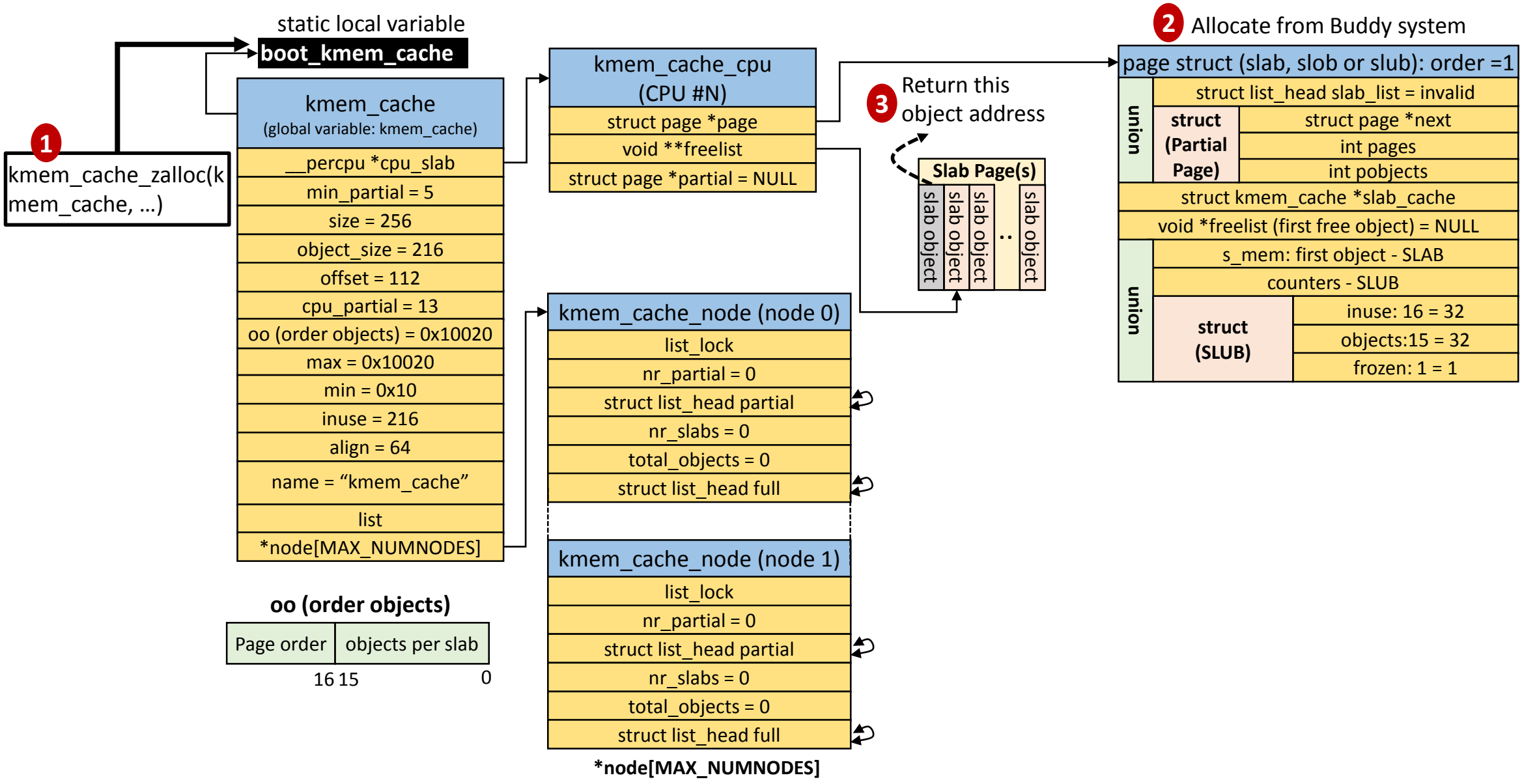
slab_alloc_node(): slowpath #2 – Allocate a page from Buddy system



[slowpath #2] **Before**: bootstrap(&boot_kmem_cache) -> kmem_cache_zalloc()

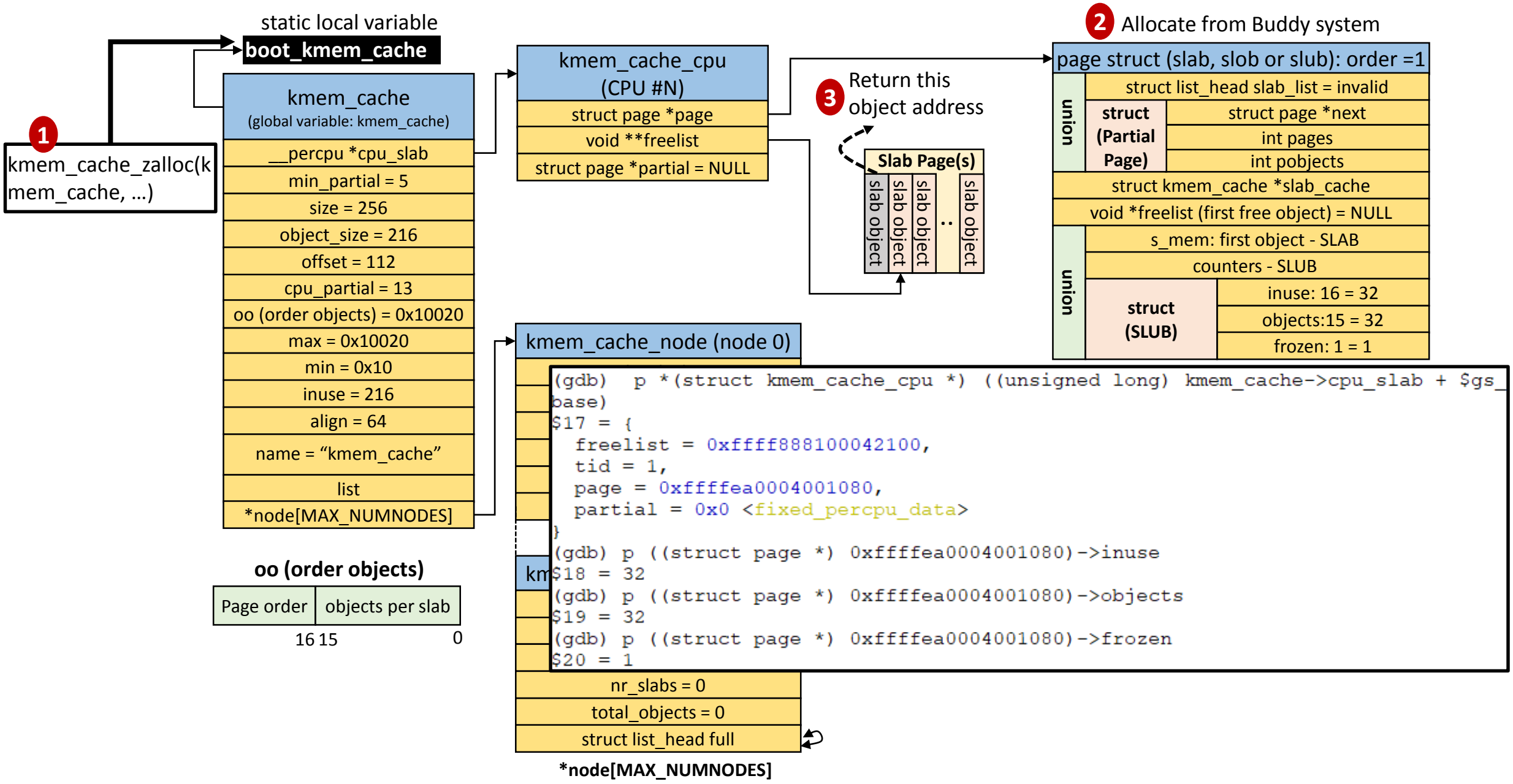


[slowpath #2] After: bootstrap(&boot_kmem_cache) -> kmem_cache_zalloc()



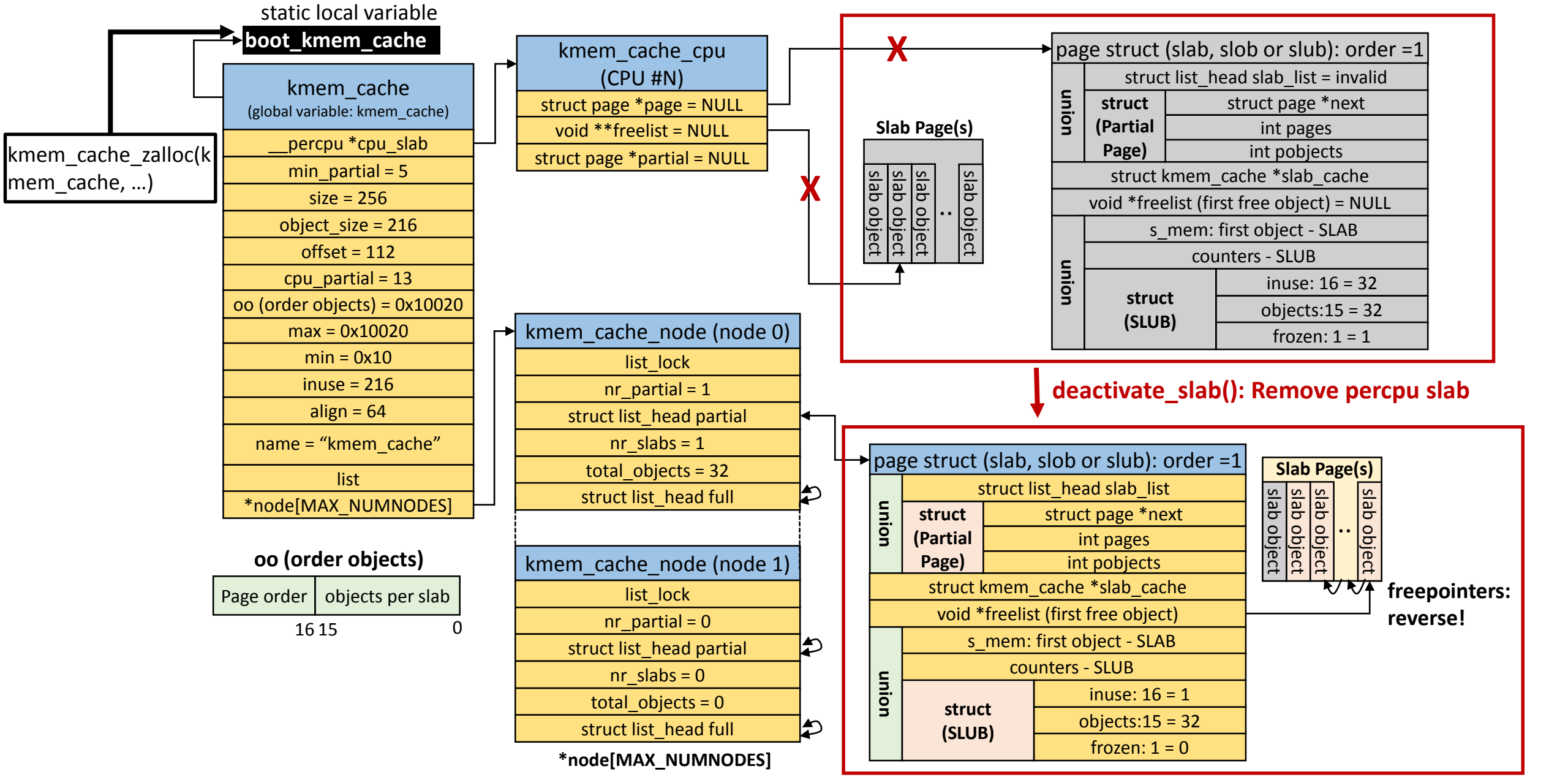
slab_alloc_node(): slowpath #2 – Allocate a page from Buddy system

[slowpath #2] After: bootstrap(&boot_kmem_cache) -> kmem_cache_zalloc()

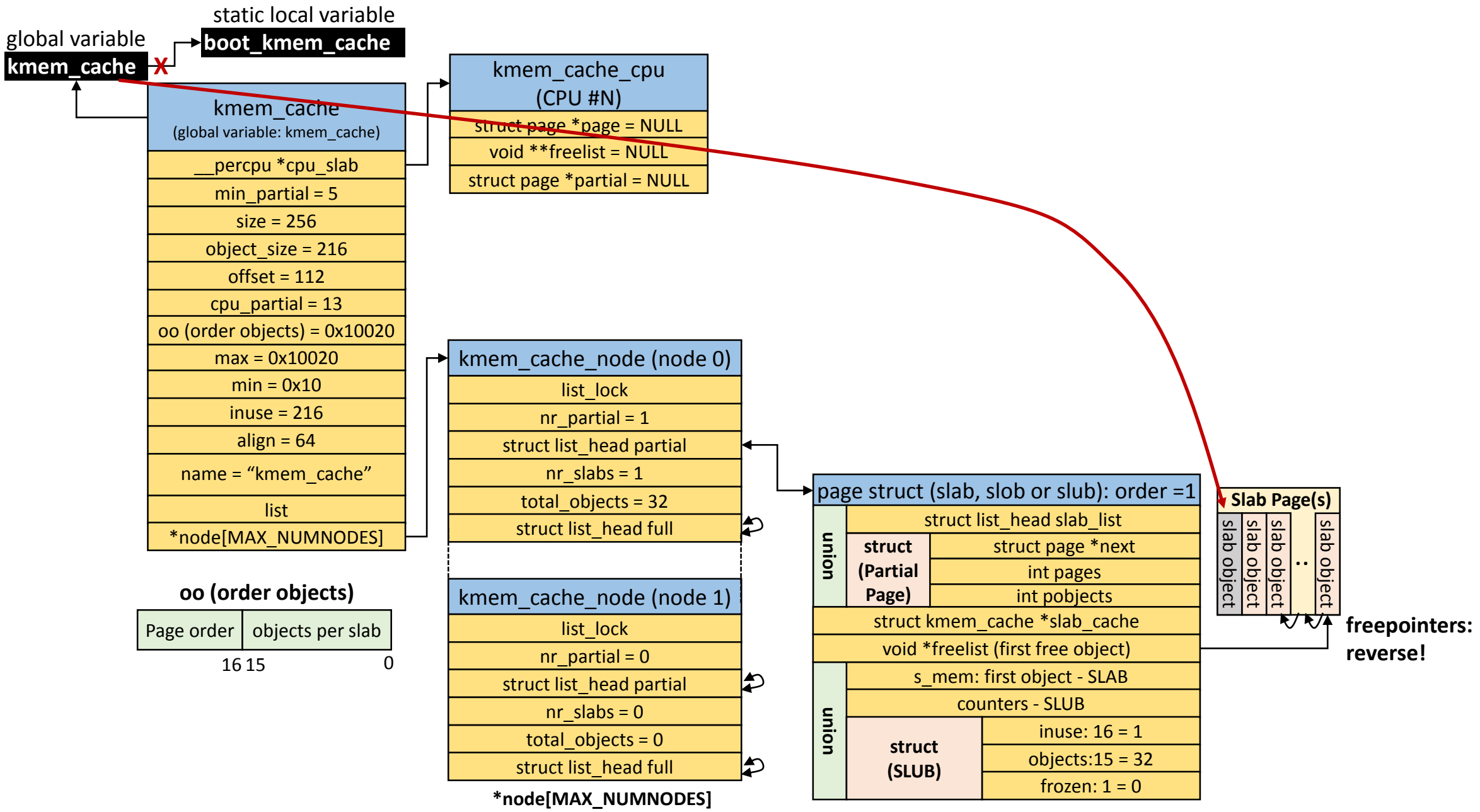



```
(gdb)  p *(struct kmem_cache_cpu *) ((unsigned long) kmem_cache->cpu_slab + $gs_
base)
$17 = {
  freelist = 0xffff888100042100,
  tid = 1,
  page = 0xffffea0004001080,
  partial = 0x0 <fixed_percpu_data>
}
(gdb) p ((struct page *) 0xffffea0004001080)->inuse
$18 = 32
(gdb) p ((struct page *) 0xffffea0004001080)->objects
$19 = 32
(gdb) p ((struct page *) 0xffffea0004001080)->frozen
$20 = 1
```

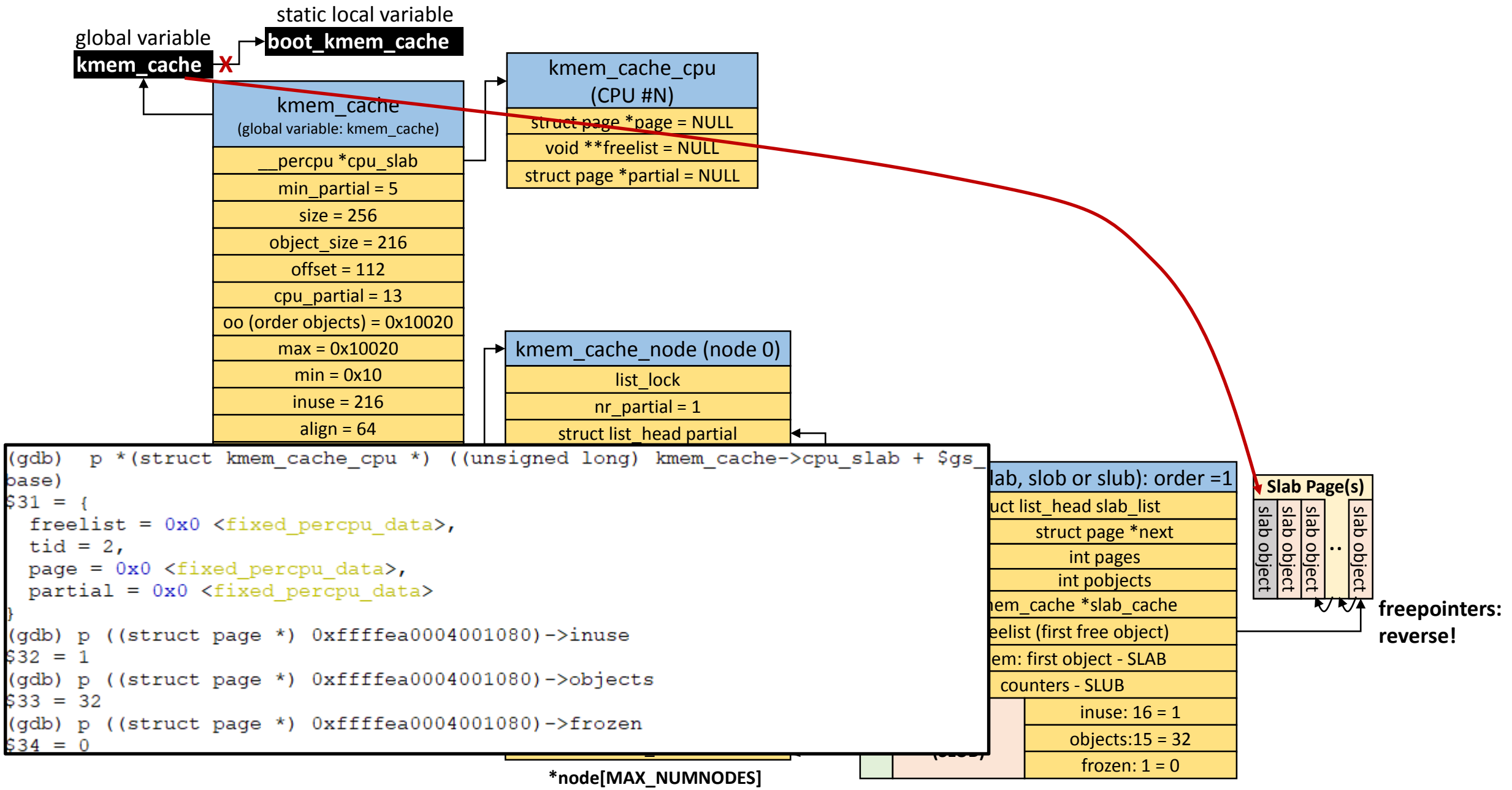
[slowpath #2] **After**: bootstrap(&boot_kmem_cache) -> __flush_cpu_slab()



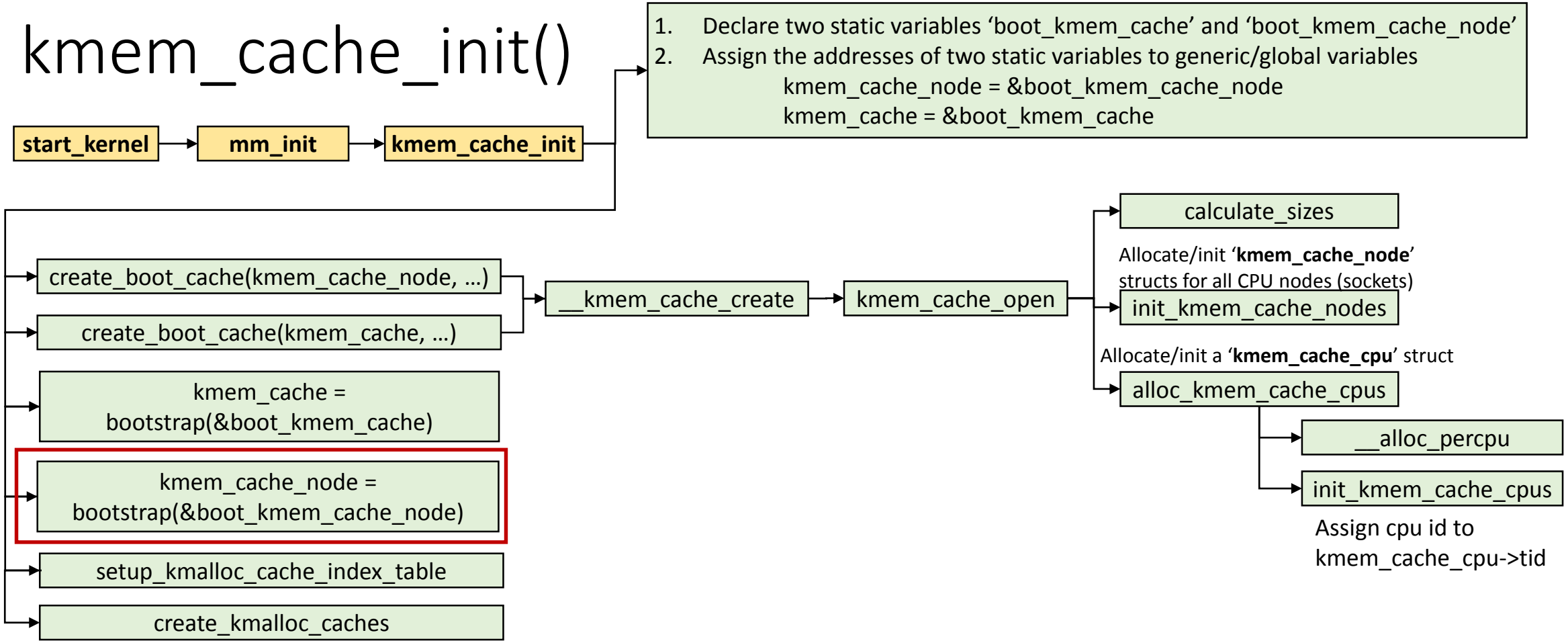
kmem_cache = bootstrap(&boot_kmem_cache): after function returns



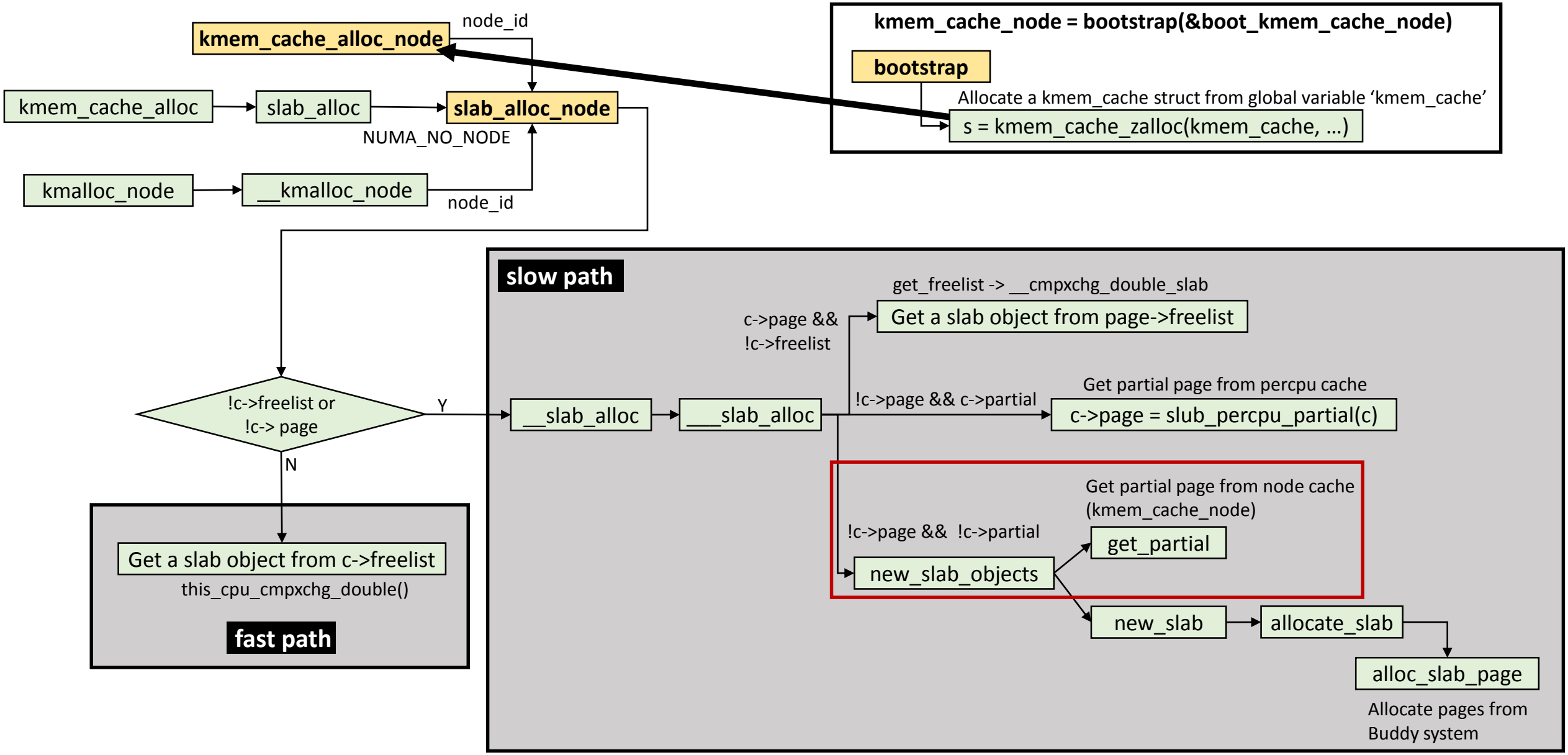
kmem_cache = bootstrap(&boot_kmem_cache): after function returns



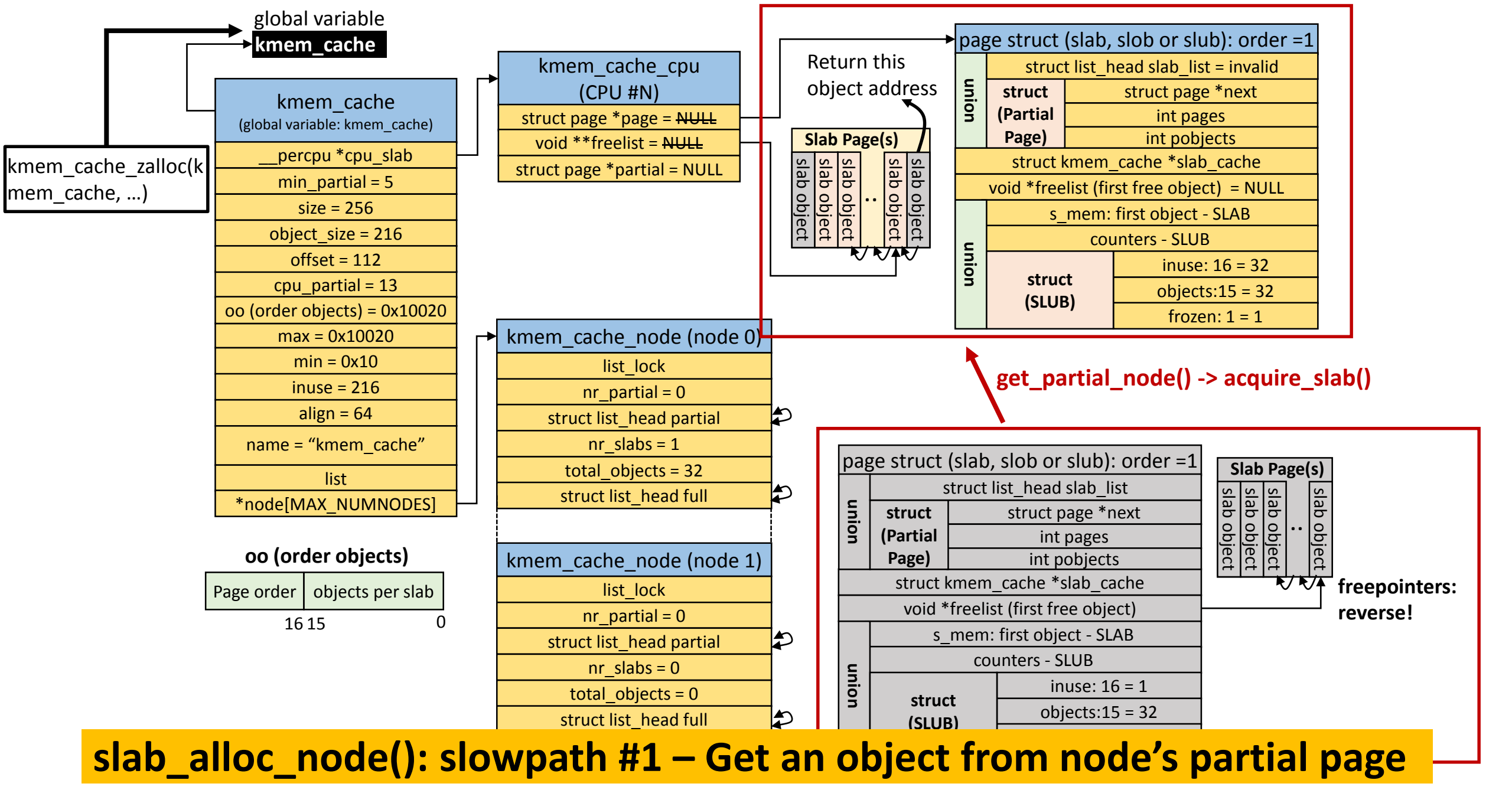
kmem_cache_init()



slab_alloc_node(): slowpath #1 – Get an object from node's partial page

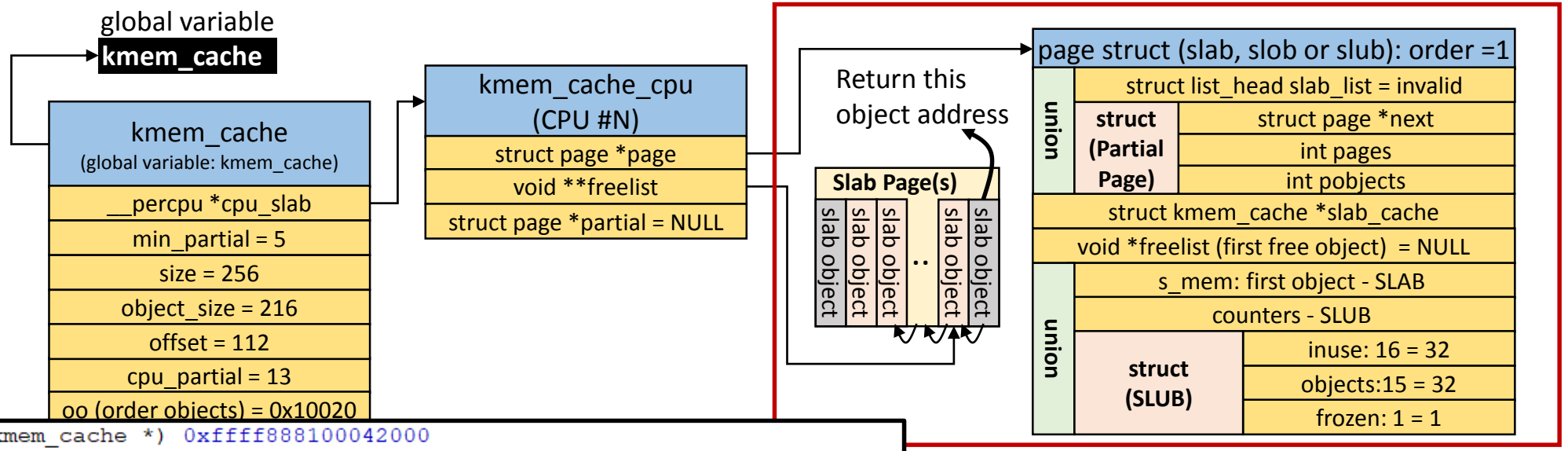


kmem_cache_node = bootstrap(&boot_kmem_cache_node) -> kmem_cache_zalloc(...)



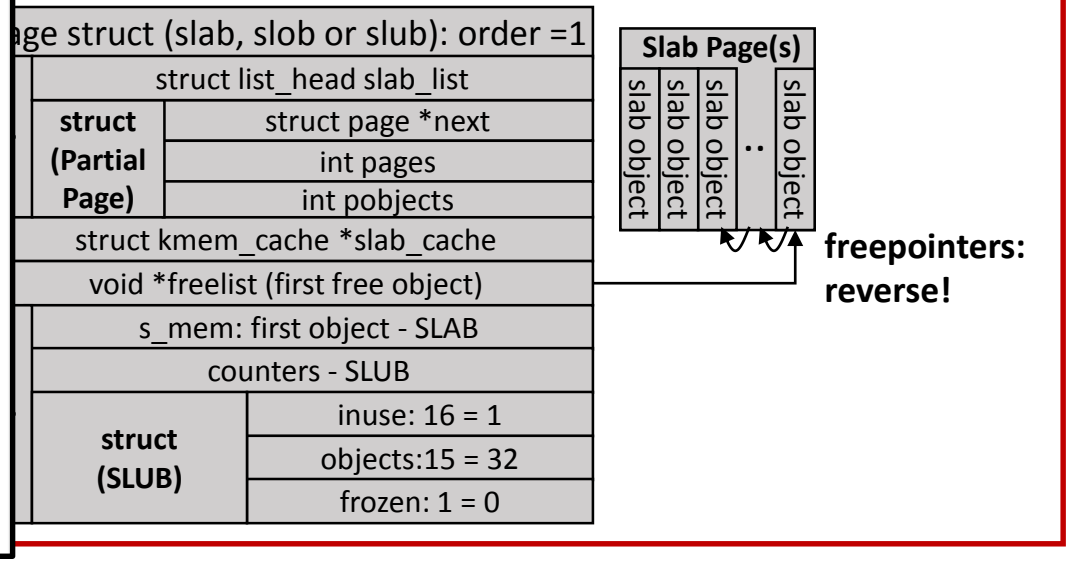
slab_alloc_node(): slowpath #1 – Get an object from node's partial page

kmem_cache_node = bootstrap(&boot_kmem_cache_node) -> kmem_cache_zalloc(...)

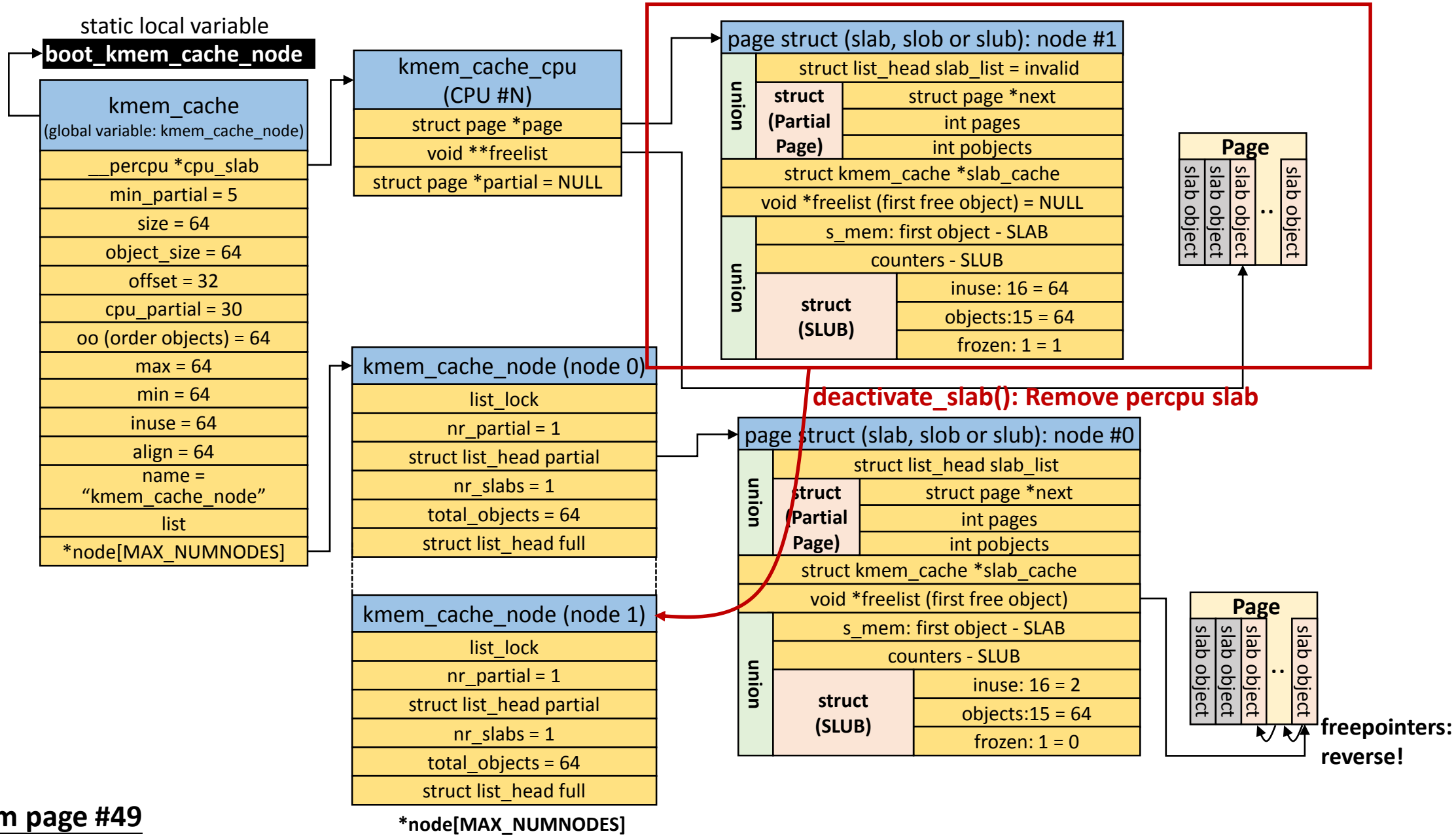



```
$63 = (struct kmem_cache *) 0xffff888100042000
(gdb)  p *(struct kmem_cache_cpu *) ((unsigned long) kmem_cache->cpu_slab + $gs_
base)
$64 = {
  freelist = 0xffff888100043e00,
  tid = 3,
  page = 0xffffea0004001080,
  partial = 0x0 <fixed_percpu_data>
}
(gdb) p ((struct page *) 0xffffea0004001080)->inuse
$65 = 32
(gdb) p ((struct page *) 0xffffea0004001080)->objects
$66 = 32
(gdb) p ((struct page *) 0xffffea0004001080)->frozen
$67 = 1
(gdb) p ((struct page *) 0xffffea0004001080)->slab_list
$68 = {
  next = 0xdead000000000100,
  prev = 0xdead000000000122
}
(gdb) p ((struct page *) 0xffffea0004001080)->slab_cache
$69 = (struct kmem_cache *) 0xffff888100042000
(gdb) p kmem_cache
$70 = (struct kmem_cache *) 0xffff888100042000
```

get_partial_node() -> acquire_slab()

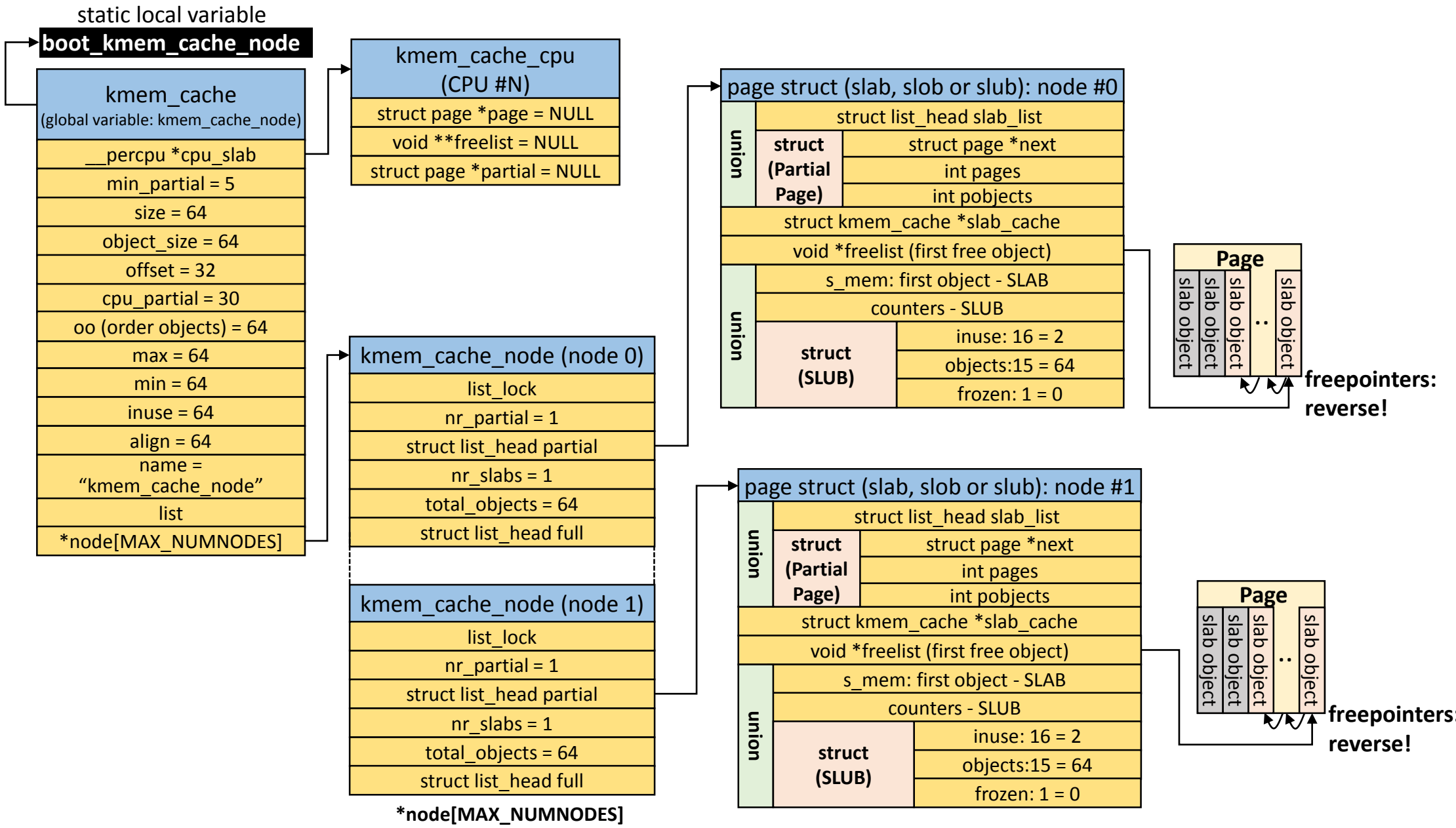


kmem_cache_node = bootstrap(&boot_kmem_cache_node) -> __flush_cpu_slab()



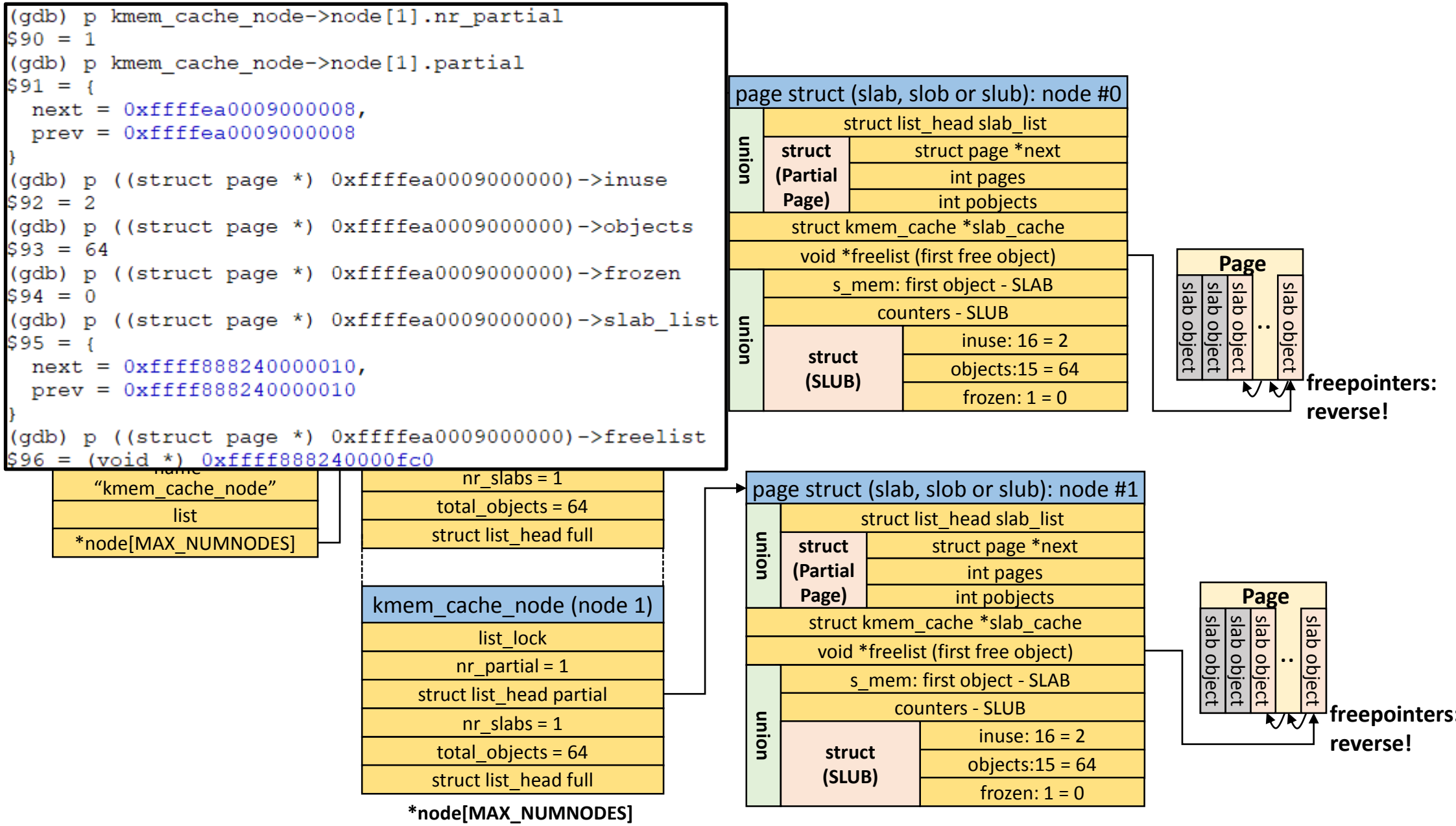
Reference from page #49

kmem_cache_node = bootstrap(&boot_kmem_cache_node) -> __flush_cpu_slab()

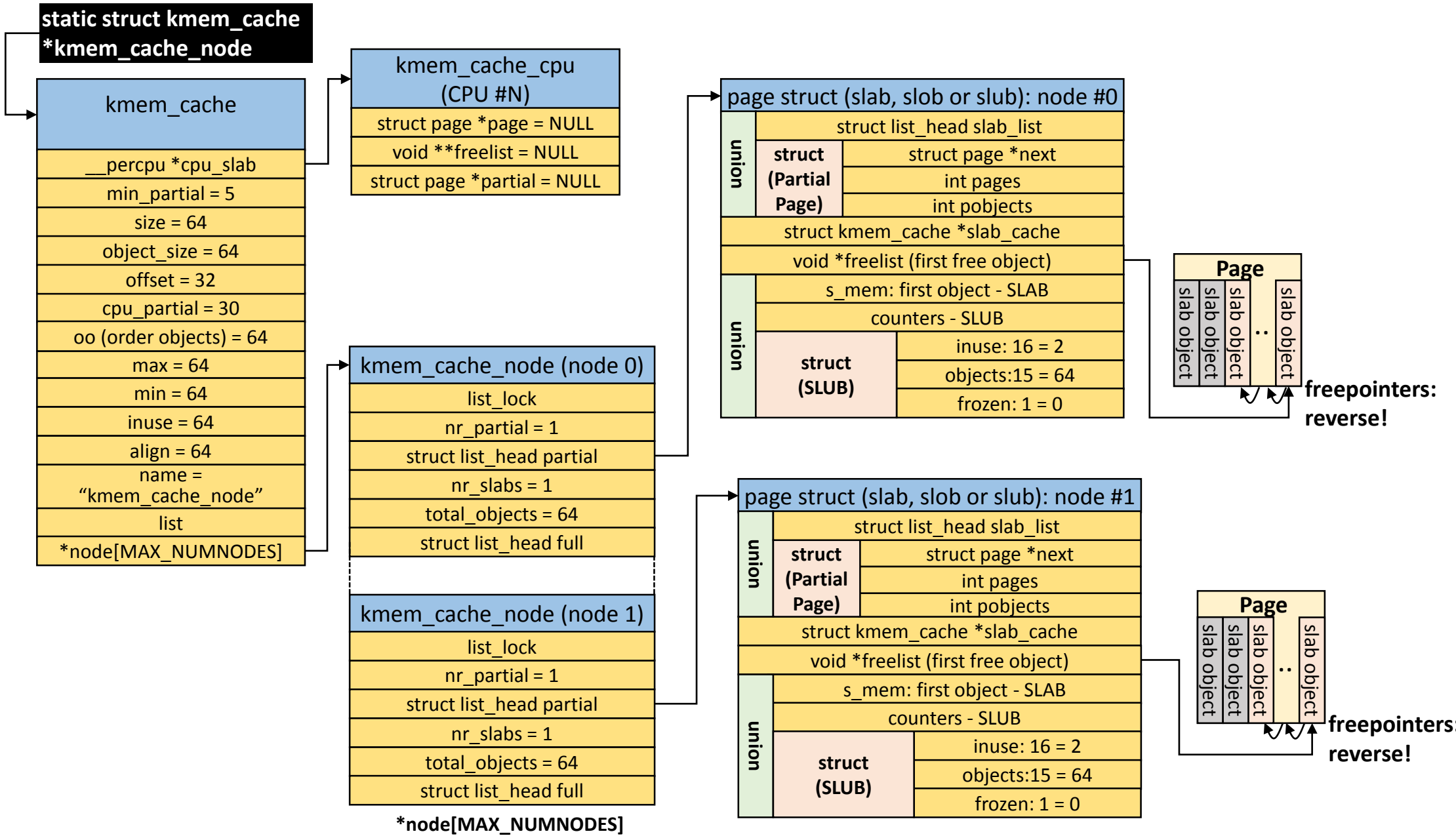


kmem_cache_node = bootstrap(&boot_kmem_cache_node) -> __flush_cpu_slab()

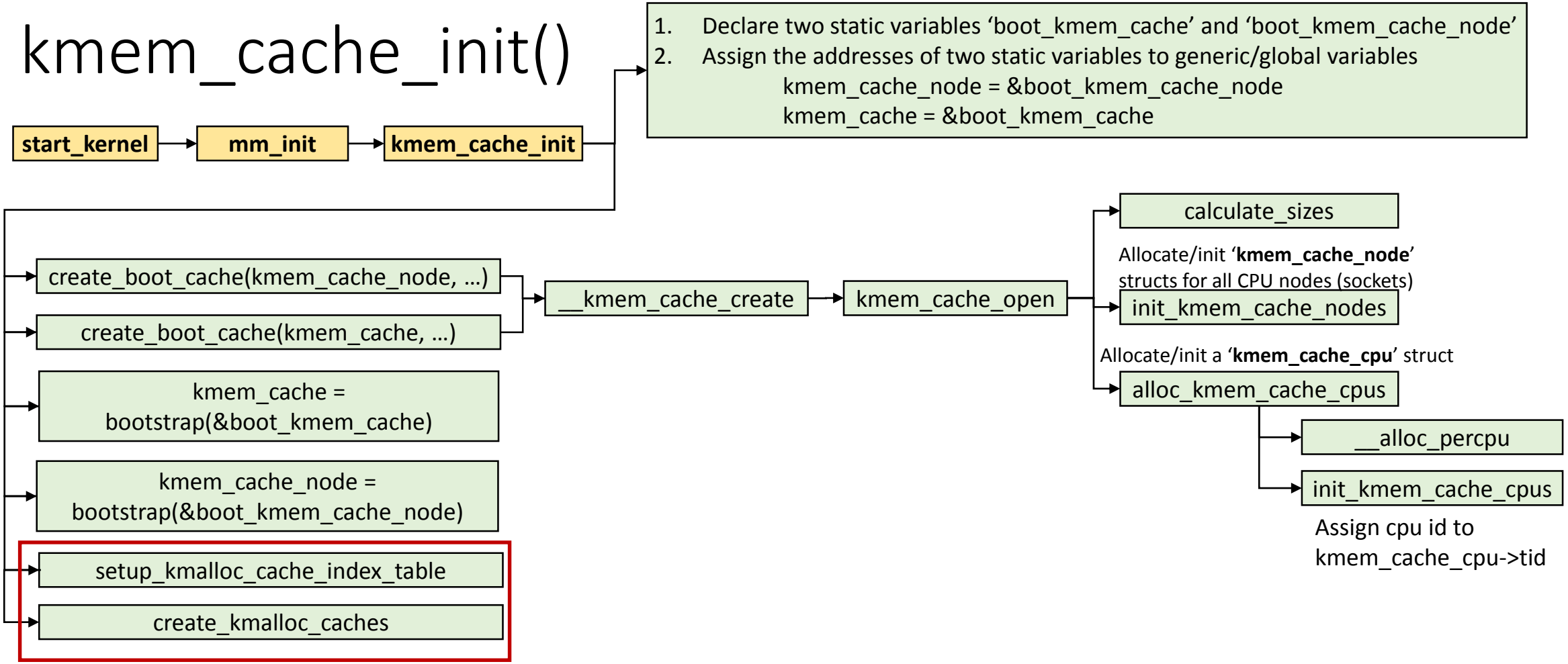
```
(gdb) p kmem_cache_node->node[1].nr_partial
$90 = 1
(gdb) p kmem_cache_node->node[1].partial
$91 = {
  next = 0xffffea0009000008,
  prev = 0xffffea0009000008
}
(gdb) p ((struct page *) 0xffffea0009000000)->inuse
$92 = 2
(gdb) p ((struct page *) 0xffffea0009000000)->objects
$93 = 64
(gdb) p ((struct page *) 0xffffea0009000000)->frozen
$94 = 0
(gdb) p ((struct page *) 0xffffea0009000000)->slab_list
$95 = {
  next = 0xffff888240000010,
  prev = 0xffff888240000010
}
(gdb) p ((struct page *) 0xffffea0009000000)->freelist
$96 = (void *) 0xffff888240000fc0
```



After ‘kmem_cache_node = bootstrap(&boot_kmem_cache_node)’

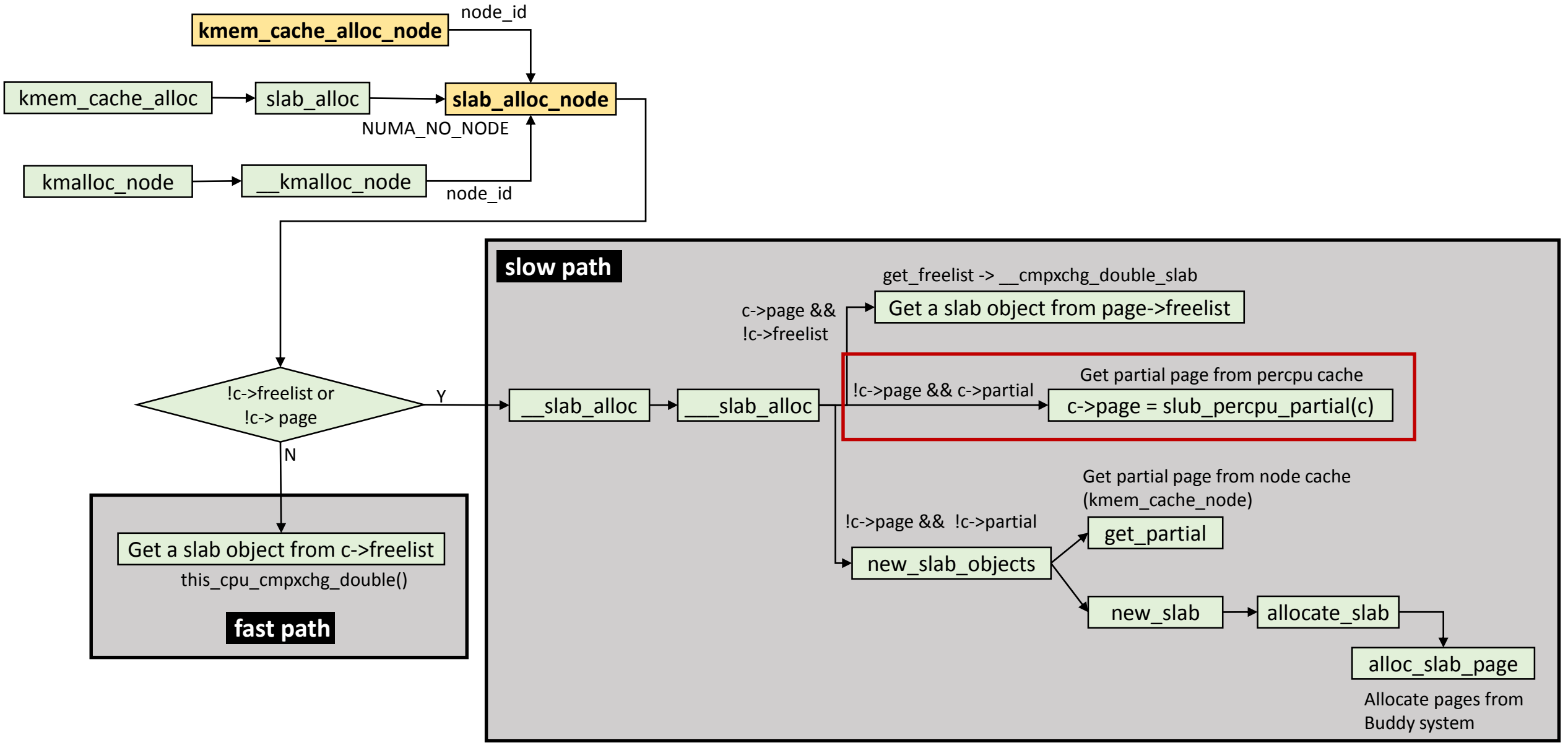


kmem_cache_init()



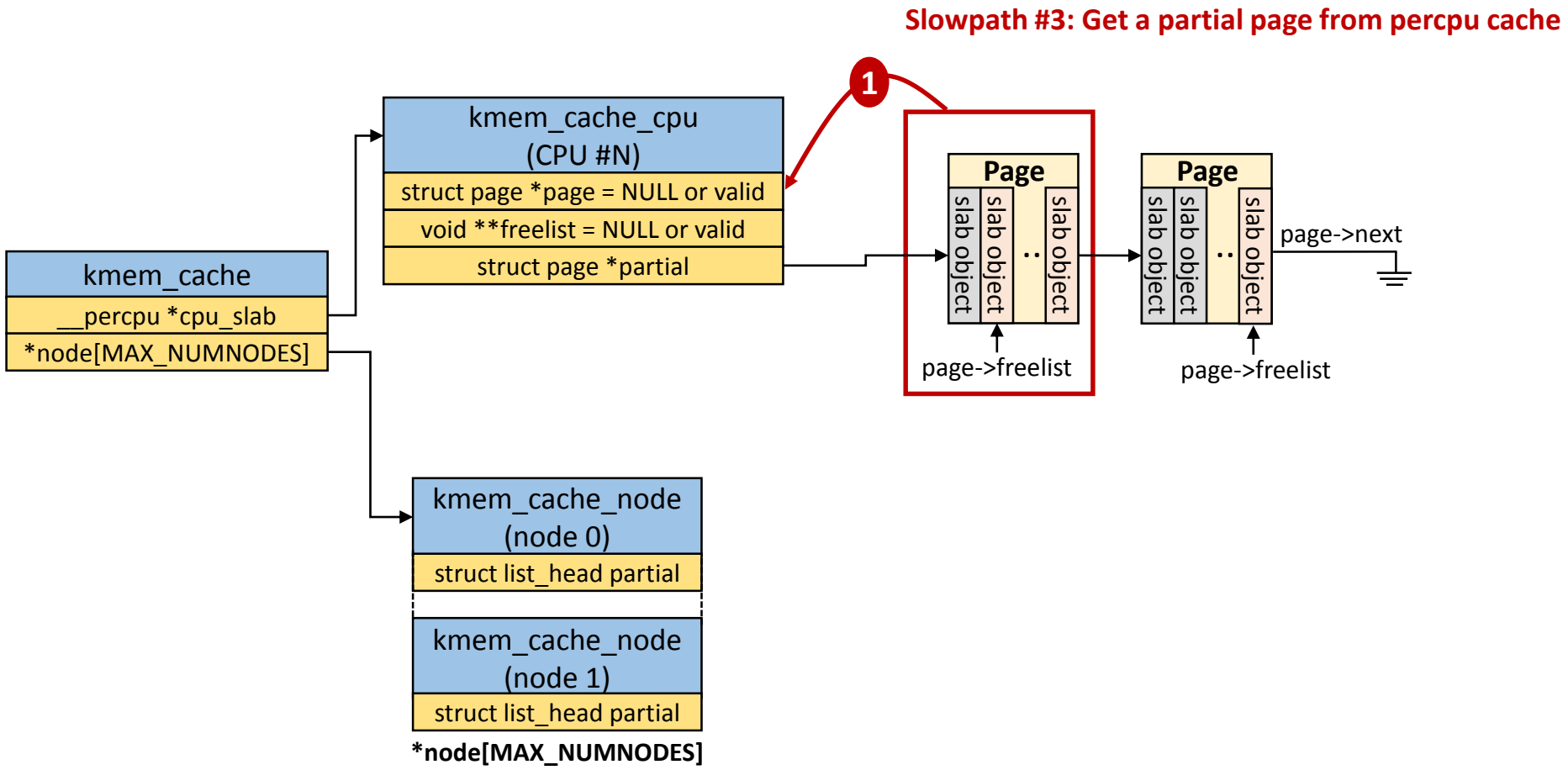
Will talk about this later

slab_alloc_node(): slowpath #3 – Get a partial page from percpu cache

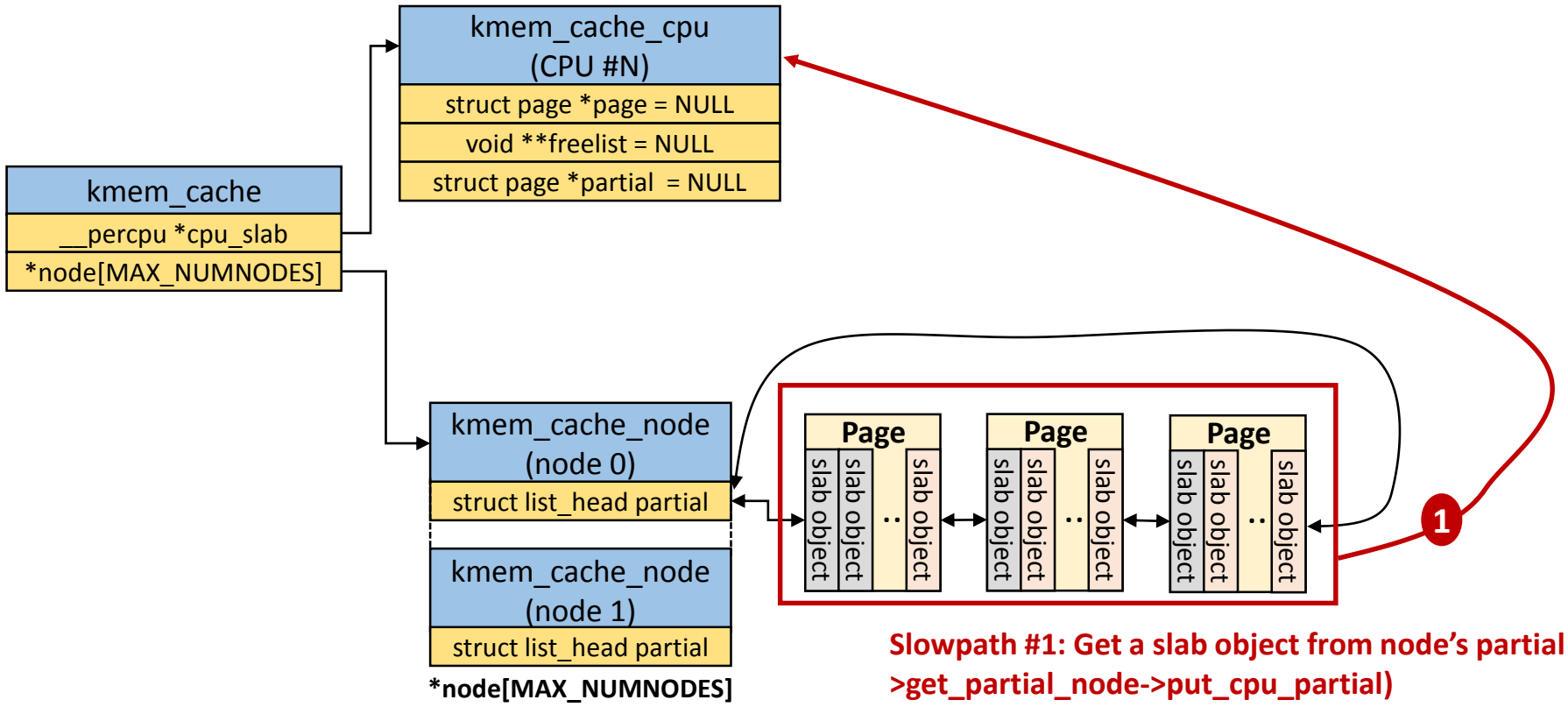


Let's talk about another slow path

slowpath #3: Get a partial page from percpu cache



When to add a page to percpu partial page? – Case 1 (1/2)

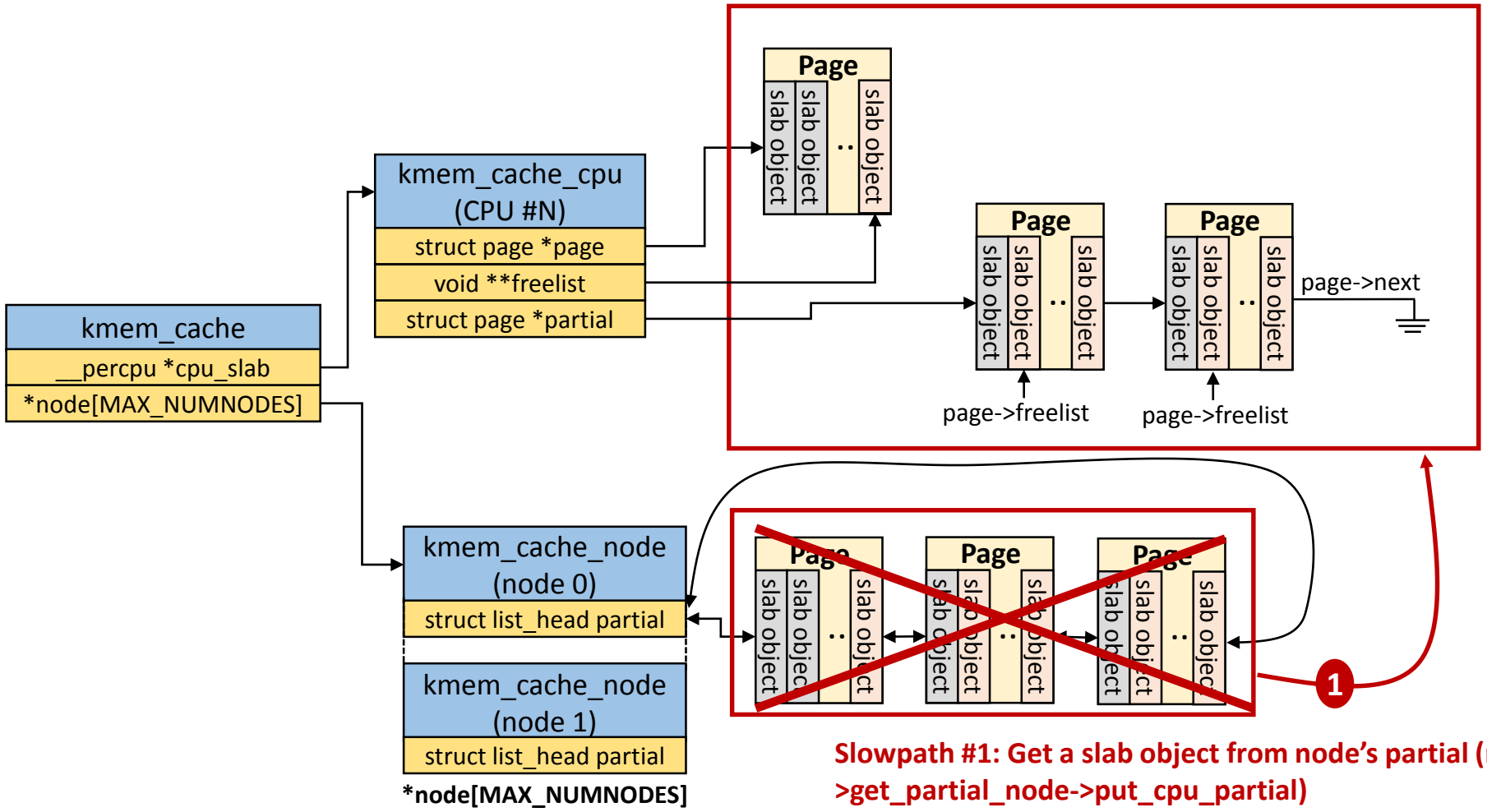


Slowpath #1: Get a slab object from node's partial (new_slab_objects->get_partial->get_partial_node->put_cpu_partial)

- **[Just for a scenario] Acquire 3 pages because of the number of kmem_cache->cpu_partial**

put_cpu_partial(): Put page(s) into a percpu partial page

When to add a page to percpu partial page? – Case 1 (2/2)

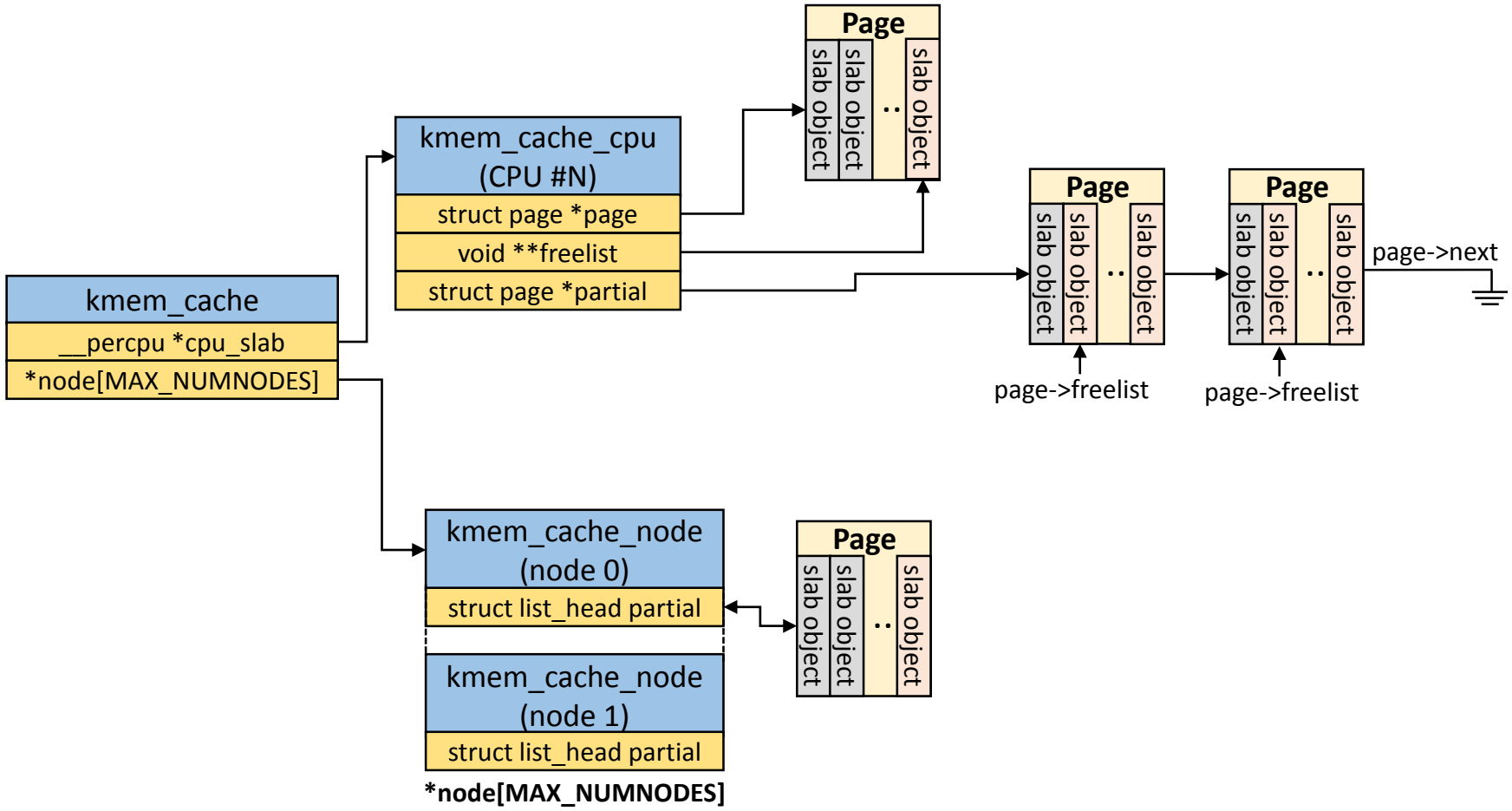


Slowpath #1: Get a slab object from node's partial (new_slab_objects->get_partial->get_partial_node->put_cpu_partial)

- **[Just for a scenario] Acquire 3 pages because of the number of kmem_cache->cpu_partial**

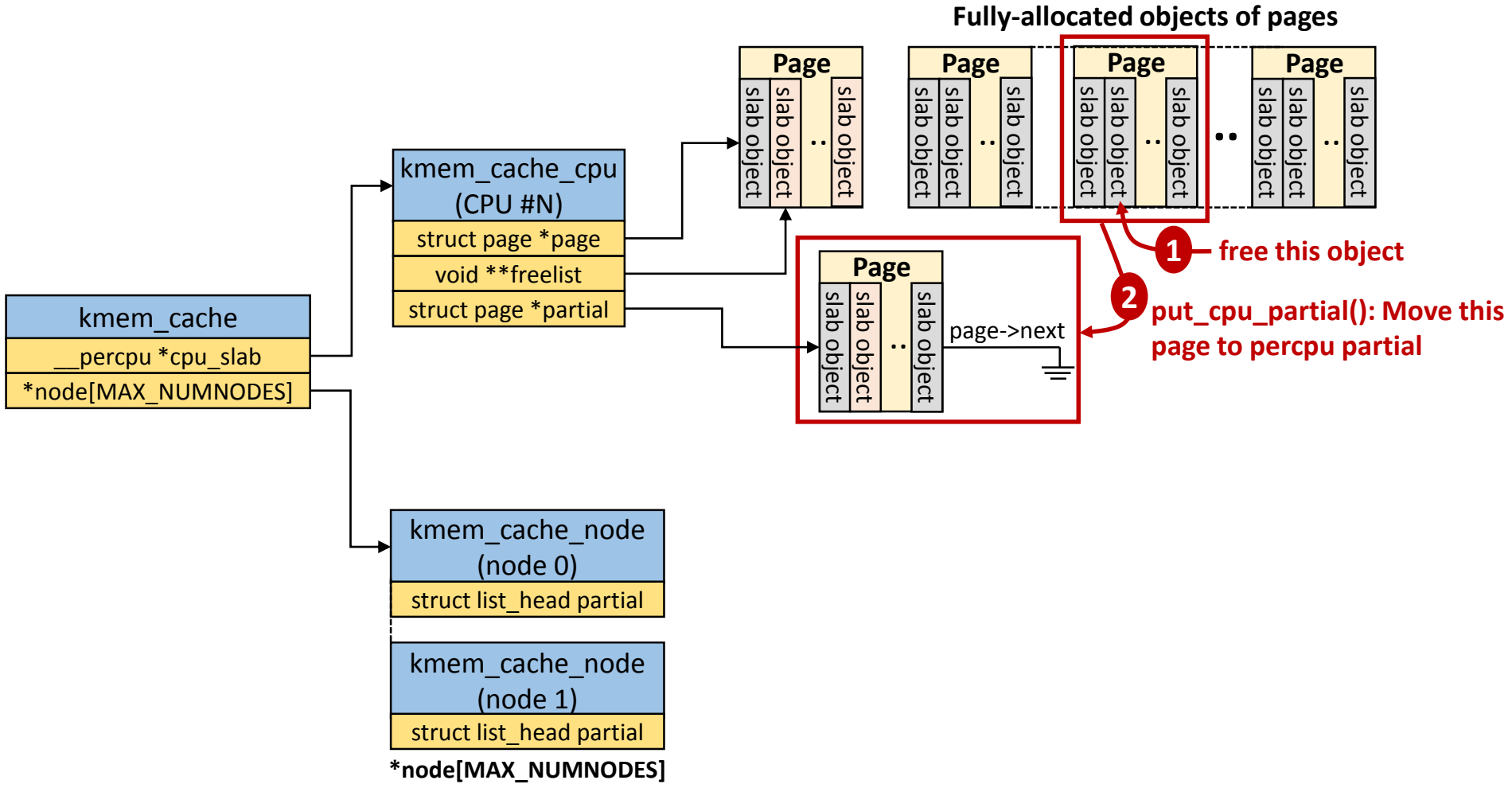
put_cpu_partial(): Put page(s) into a percpu partial page

When to add a page to percpu partial page? – Case 2: slab_free()



slab_free(): Move page(s) to percpu partial and node partial (if possible)

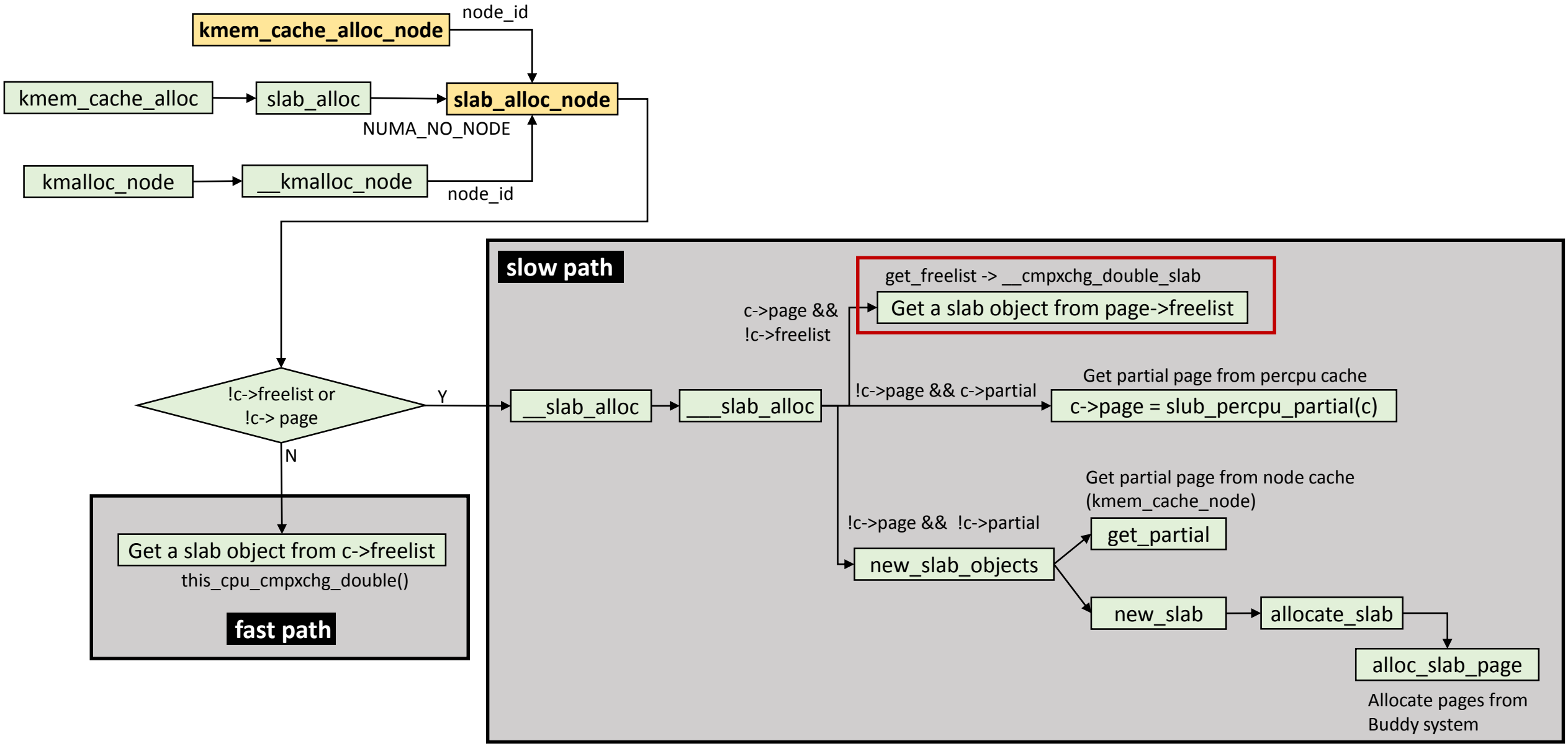
When to add a page to percpu partial page? – Case 2: slab_free()



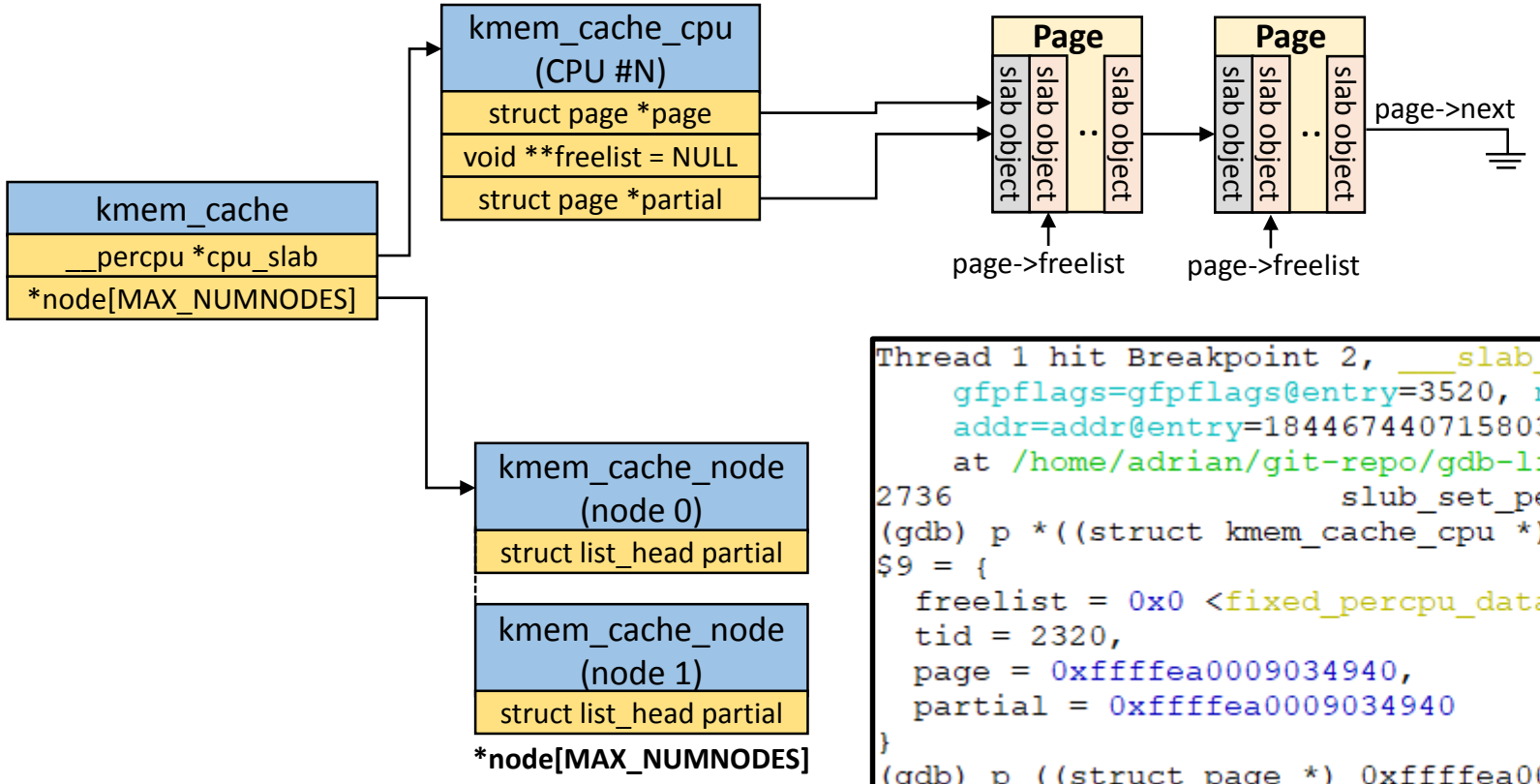
slab_free(): Move page(s) to percpu partial and node partial (if possible)

*** Detail will be discussed in section "[Cache release/free] kmem_cache_free()"**

slab_alloc_node(): slowpath #4 – Get a slab object page from page->freelist



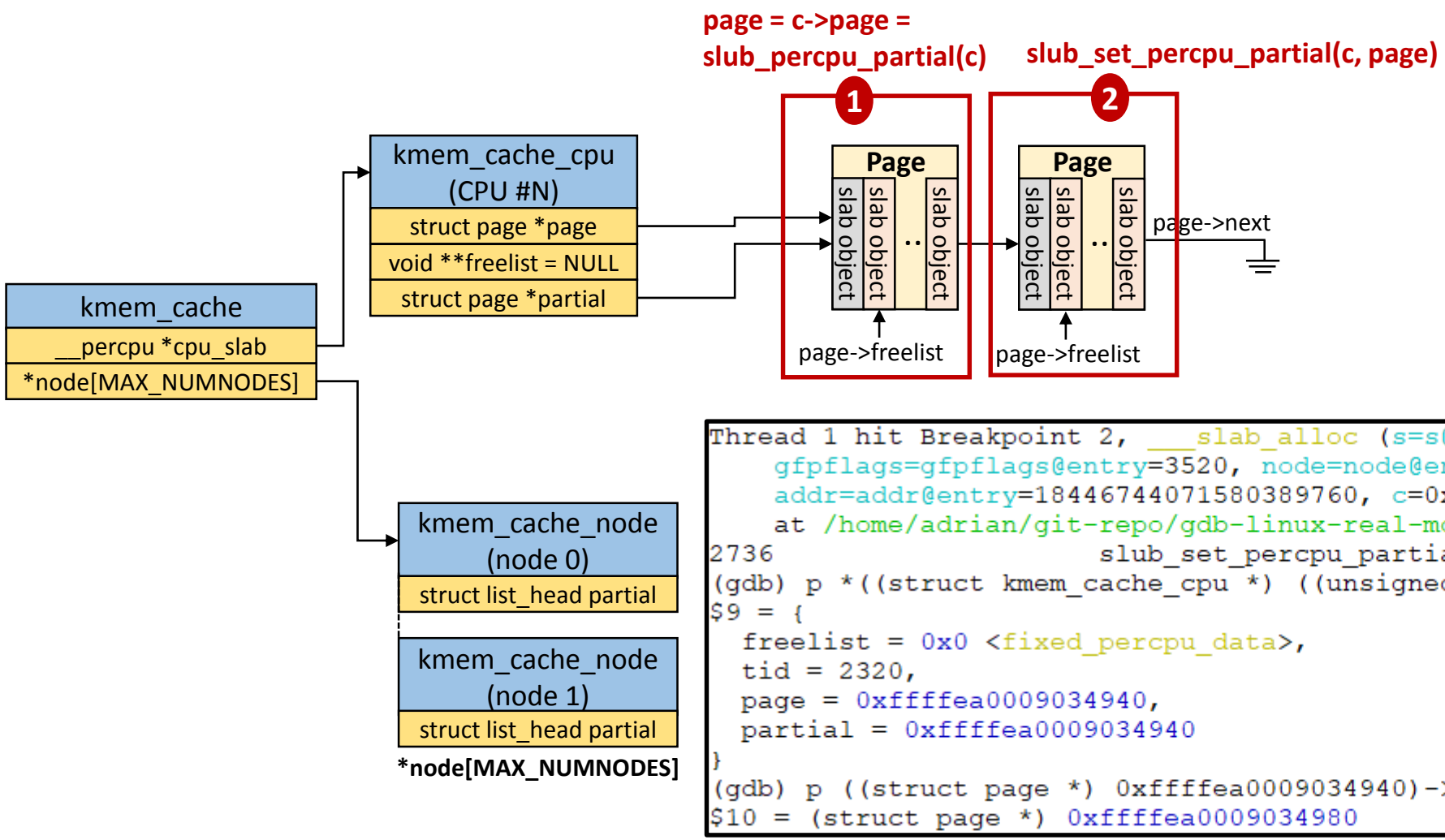
slowpath #4: Get a slab object from page->freelist




```
Thread 1 hit Breakpoint 2, ___slab_alloc (s=s@entry=0xffff888240002700,
    gfpflags=gfpflags@entry=3520, node=node@entry=-1,
    addr=addr@entry=18446744071580389760, c=0xffff888237c22590)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2736
2736                    slub_set_percpu_partial(c, page);
(gdb) p *((struct kmem_cache_cpu *) ((unsigned long) s->cpu_slab + $gs_base))
$9 = {
  freelist = 0x0 <fixed_percpu_data>,
  tid = 2320,
  page = 0xffffea0009034940,
  partial = 0xffffea0009034940
}
(gdb) p ((struct page *) 0xffffea0009034940)->next
$10 = (struct page *) 0xffffea0009034980
```

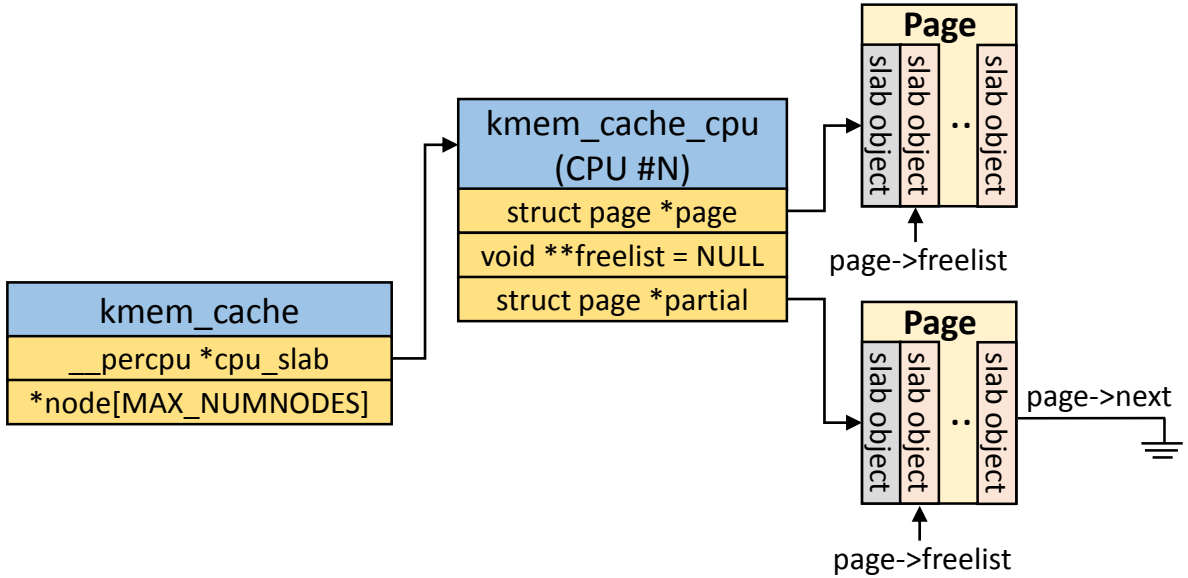
page descriptor address = virtual memory map = vmemmap_base: [page #5 of Decompressed vmlinux: linux kernel initialization from page table configuration perspective]

slowpath #4: Get a slab object from page->freelist




```
Thread 1 hit Breakpoint 2, ___slab_alloc (s=s@entry=0xffff888240002700,
    gfpflags=gfpflags@entry=3520, node=node@entry=-1,
    addr=addr@entry=18446744071580389760, c=0xffff888237c22590)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2736
2736                    slub_set_percpu_partial(c, page);
(gdb) p *((struct kmem_cache_cpu *) ((unsigned long) s->cpu_slab + $gs_base))
$9 = {
  freelist = 0x0 <fixed_percpu_data>,
  tid = 2320,
  page = 0xffffea0009034940,
  partial = 0xffffea0009034940
}
(gdb) p ((struct page *) 0xffffea0009034940)->next
$10 = (struct page *) 0xffffea0009034980
```

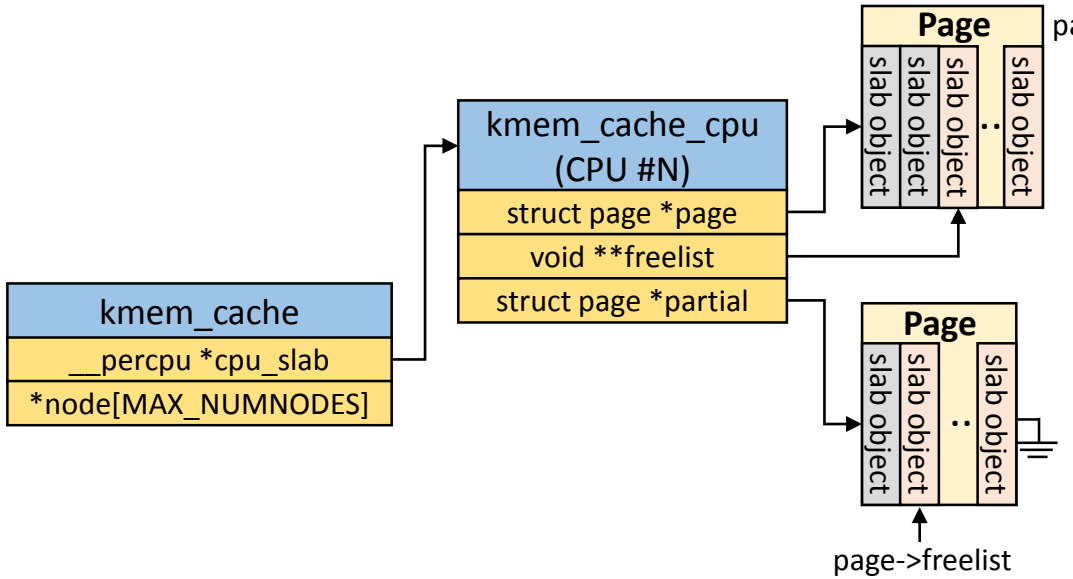
slowpath #4: Get a slab object from page->freelist




```
static void *___slab_alloc(struct kmem_cache *s, gfp_t gfpflags, int node,
                           unsigned long addr, struct kmem_cache_cpu *c)
+-- 43 lines: {-------------------------------------------------------------

        /* must check again c->freelist in case of cpu migration or IRQ */
        freelist = c->freelist;
        if (freelist)
                goto load_freelist;

        freelist = get_freelist(s, page);

        if (!freelist) {
                c->page = NULL;
                stat(s, DEACTIVATE_BYPASS);
                goto new_slab;
        }

        stat(s, ALLOC_REFILL);

load_freelist:
+--  6 lines: freelist is pointing to the list of objects to be used.------
        c->freelist = get_freepointer(s, freelist);
        c->tid = next_tid(c->tid);
        return freelist;

mm/slub.c                                                        2654,1
```

```
Thread 1 hit Breakpoint 4, get_freelist (page=<optimized out>,
    s=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2633
2633            } while (!__cmpxchg_double_slab(s, page,
(gdb) p *((struct kmem_cache_cpu *) ((unsigned long) s->cpu_slab + $gs_base))
value has been optimized out
(gdb) f 1
#1  ___slab_alloc (s=s@entry=0xffff888240002700, gfpflags=gfpflags@entry=3520,
    node=<optimized out>, node@entry=-1, addr=addr@entry=18446744071580389760,
    c=0xffff888237c22590)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2711
2711            freelist = get_freelist(s, page);
(gdb) p *((struct kmem_cache_cpu *) ((unsigned long) s->cpu_slab + $gs_base))
$11 = {
  freelist = 0x0 <fixed_percpu_data>,
  tid = 2320,
  page = 0xffffea0009034940,
  partial = 0xffffea0009034980
}
```

slowpath #4: Get a slab object from page->freelist




```
(gdb) bt 2
#0  ___slab_alloc (s=s@entry=0xffff888240002700, gfpflags=gfpflags@entry=3520,
    node=<optimized out>, node@entry=-1, addr=addr@entry=18446744071580389760,
    c=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2730
#1  0xffffffff811115bd in __slab_alloc (s=s@entry=0xffff888240002700,
    gfpflags=gfpflags@entry=3520, node=node@entry=-1,
    addr=addr@entry=18446744071580389760, c=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2781
(More stack frames follow...)
(gdb) p *((struct kmem_cache_cpu *) ((unsigned long) s->cpu_slab + $gs_base))
$30 = {
  freelist = 0xffff888240d25900,
  tid = 2321,
  page = 0xffffea0009034940,
  partial = 0xffffea0009034980
}
(gdb) p /x s->object_size
$31 = 0x80
(gdb) x /g 0xffff888240d25640
0xffff888240d25640:     0xffff888240d25900
(gdb) p ((struct page *) 0xffffea0009034940)->freelist
$32 = (void *) 0x0 <fixed_percpu_data>
```

2

```
(gdb) f 1
#1  ___slab_alloc (s=s@entry=0xffff888240002700, gfpflags=gfpflags@entry=3520,
    node=<optimized out>, node@entry=-1, addr=addr@entry=18446744071580389760,
    c=0xffff888237c22590)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slub.c:2711
2711            freelist = get_freelist(s, page);
(gdb) p *((struct kmem_cache_cpu *) ((unsigned long) s->cpu_slab + $gs_base))
$11 = {
  freelist = 0x0 <fixed_percpu_data>,
  tid = 2320,
  page = 0xffffea0009034940,
  partial = 0xffffea0009034980
}
(gdb) p ((struct page *) 0xffffea0009034940)->freelist
$12 = (void *) 0xffff888240d25600
```

1

/proc/slabinfo

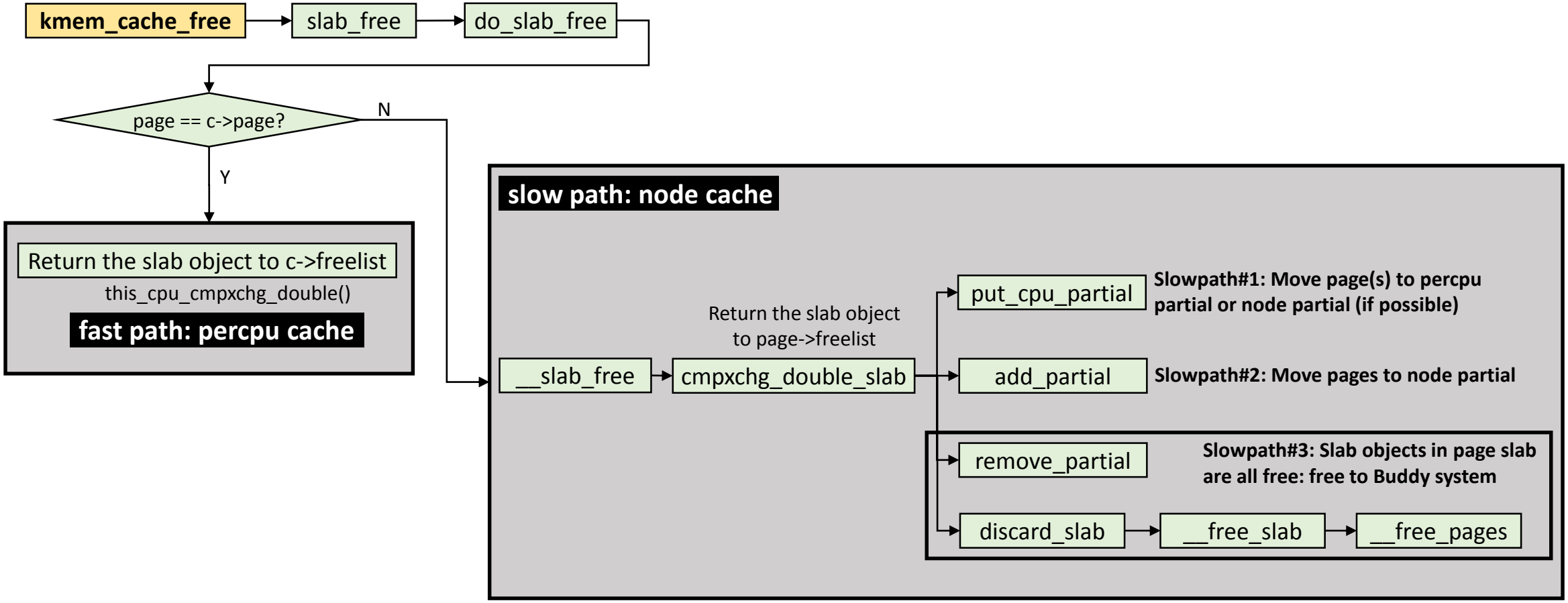
```
# less /proc/slabinfo
slabinfo - version: 2.1
# name            <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> : tunables <limit> <batchcount> <sharedfactor> : slabdata <active_slabs> <num_slabs> <sharedavail>
...
ext4_inode_cache       0      0   1088   15    4 : tunables    0    0    0 : slabdata      0      0      0
...
bio-2                 21     21    192   21    1 : tunables    0    0    0 : slabdata      1      1      0
...
bio-1                 32     32    256   16    1 : tunables    0    0    0 : slabdata      2      2      0
...
bio-0                 63     63    192   21    1 : tunables    0    0    0 : slabdata      3      3      0
...
filp                  32     32    256   16    1 : tunables    0    0    0 : slabdata      2      2      0
inode_cache         1590   1590    544   15    2 : tunables    0    0    0 : slabdata    106    106      0
dentry              1638   1638    192   21    1 : tunables    0    0    0 : slabdata     78     78      0
...
anon_vma_chain       128    128     64   64    1 : tunables    0    0    0 : slabdata      2      2      0
anon_vma              92     92     88   46    1 : tunables    0    0    0 : slabdata      2      2      0
pid                   96     96    128   32    1 : tunables    0    0    0 : slabdata      3      3      0
...
kmem_cache_node      295    320     64   64    1 : tunables    0    0    0 : slabdata      5      5      0
kmem_cache           128    128    256   16    1 : tunables    0    0    0 : slabdata      8      8      0
```

*** active_objs = num_objs - free_objs**

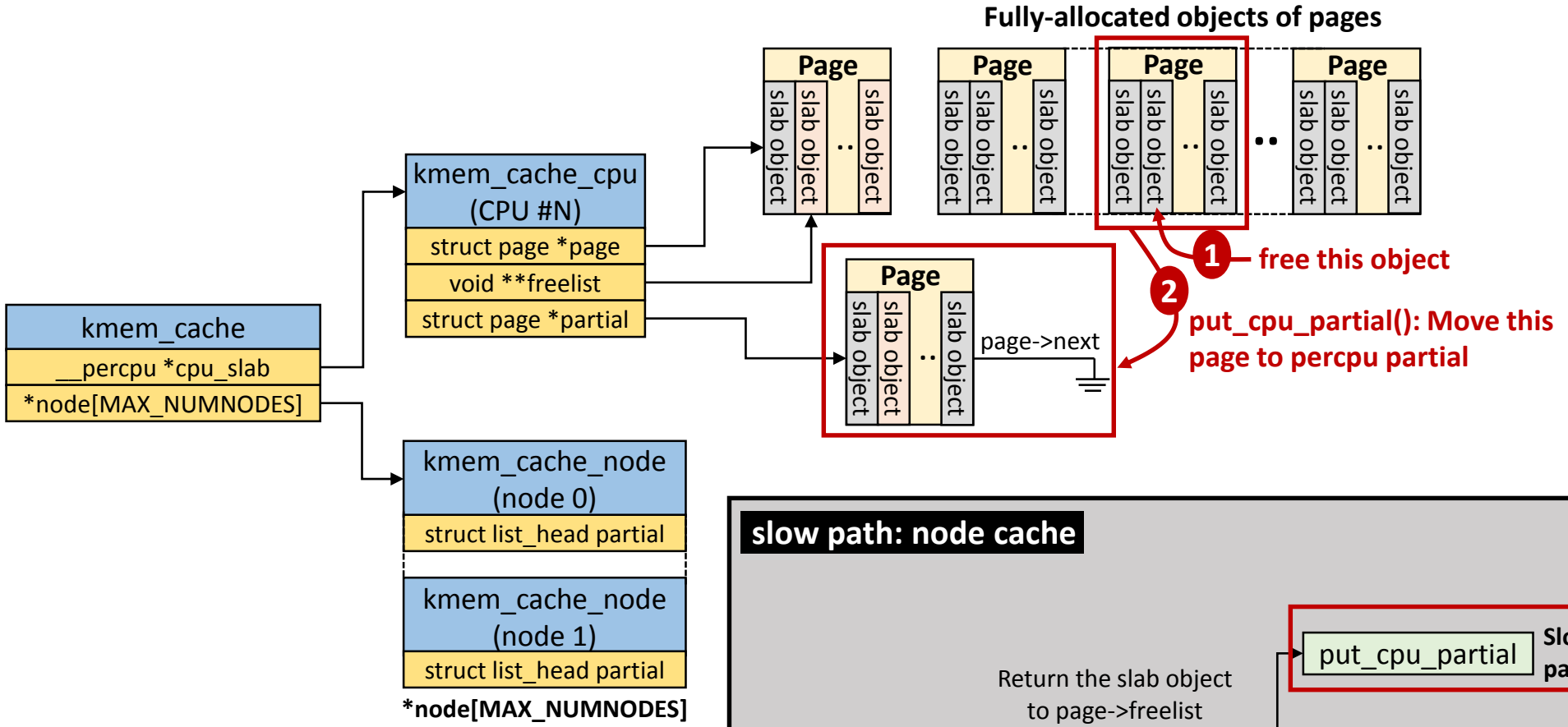
[Cache release/free] kmem_cache_free()

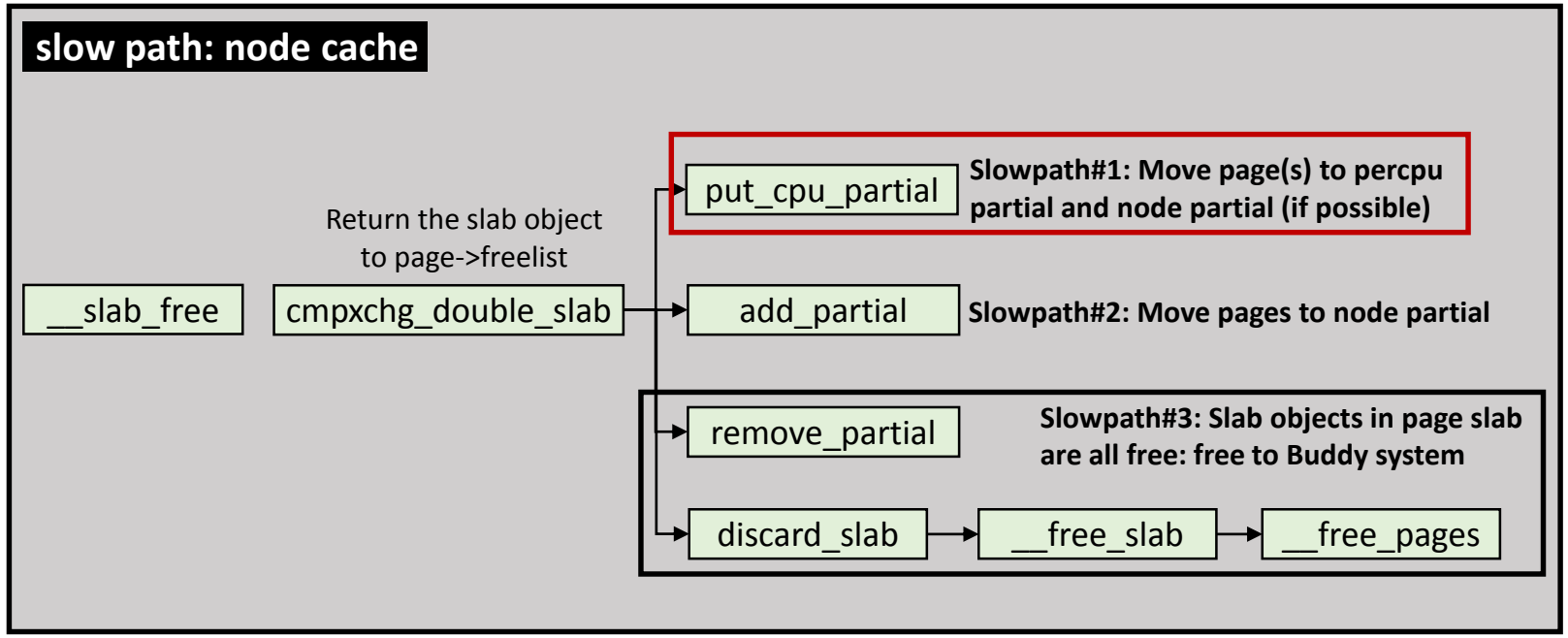
- Fast path & slow path

kmem_cache_free()

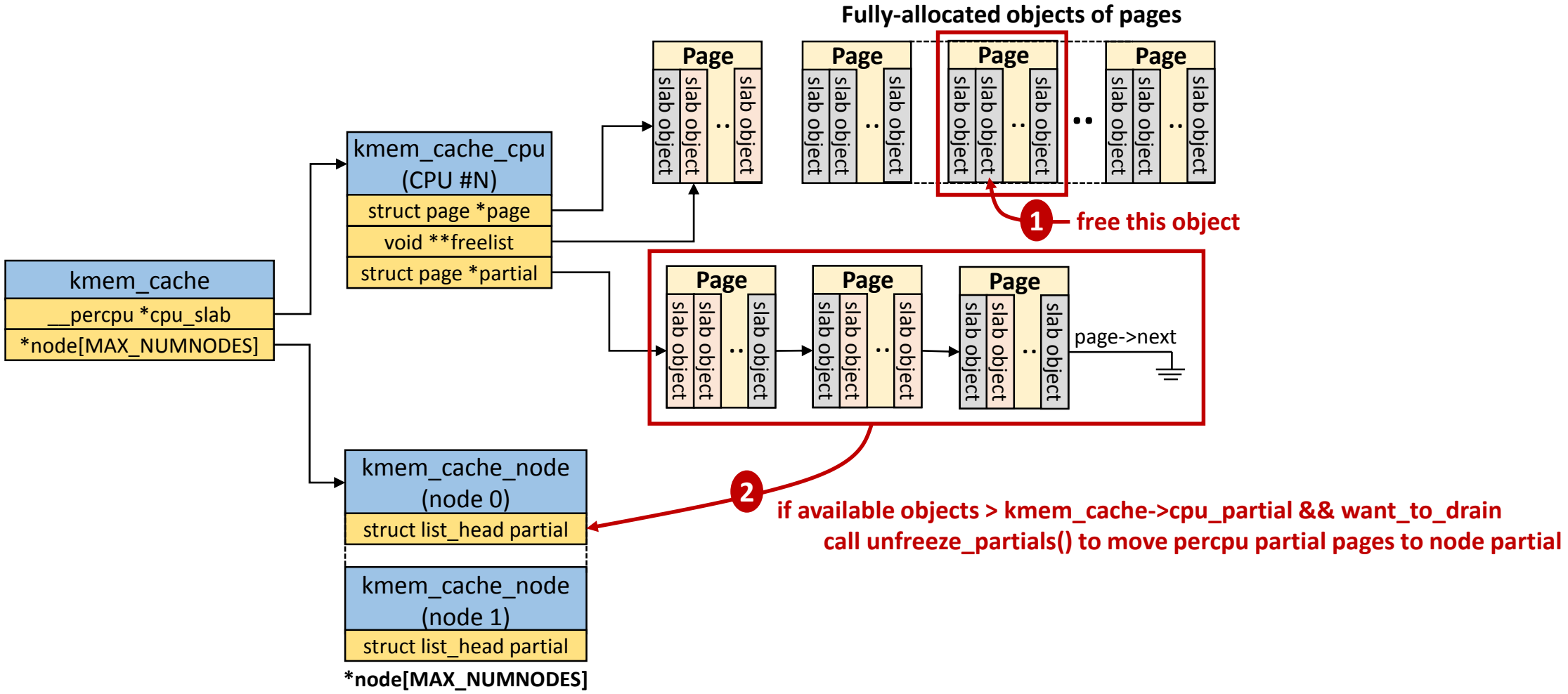


slowpath #1:put_cpu_partial() – Case 1: Move page(s) to percpu partial

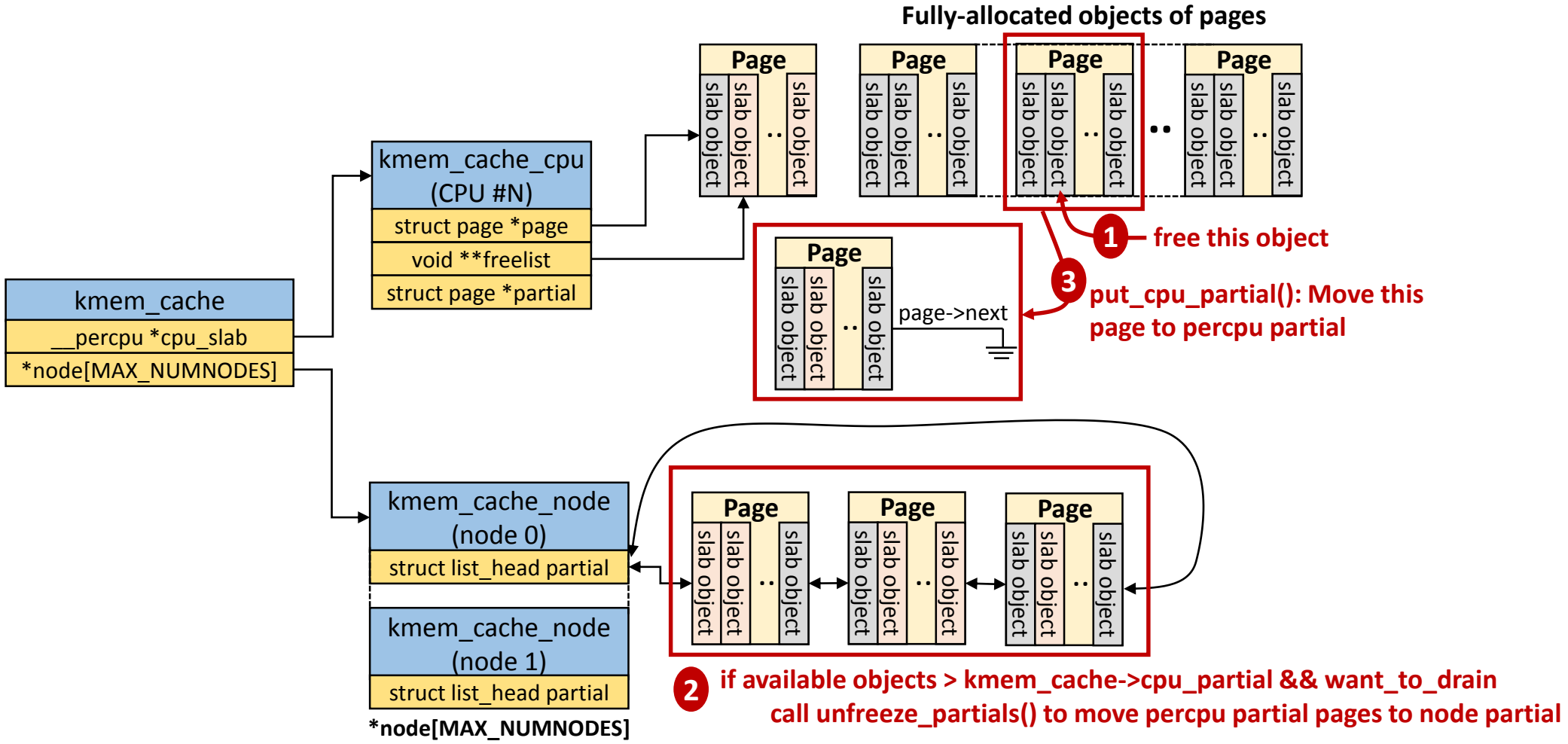




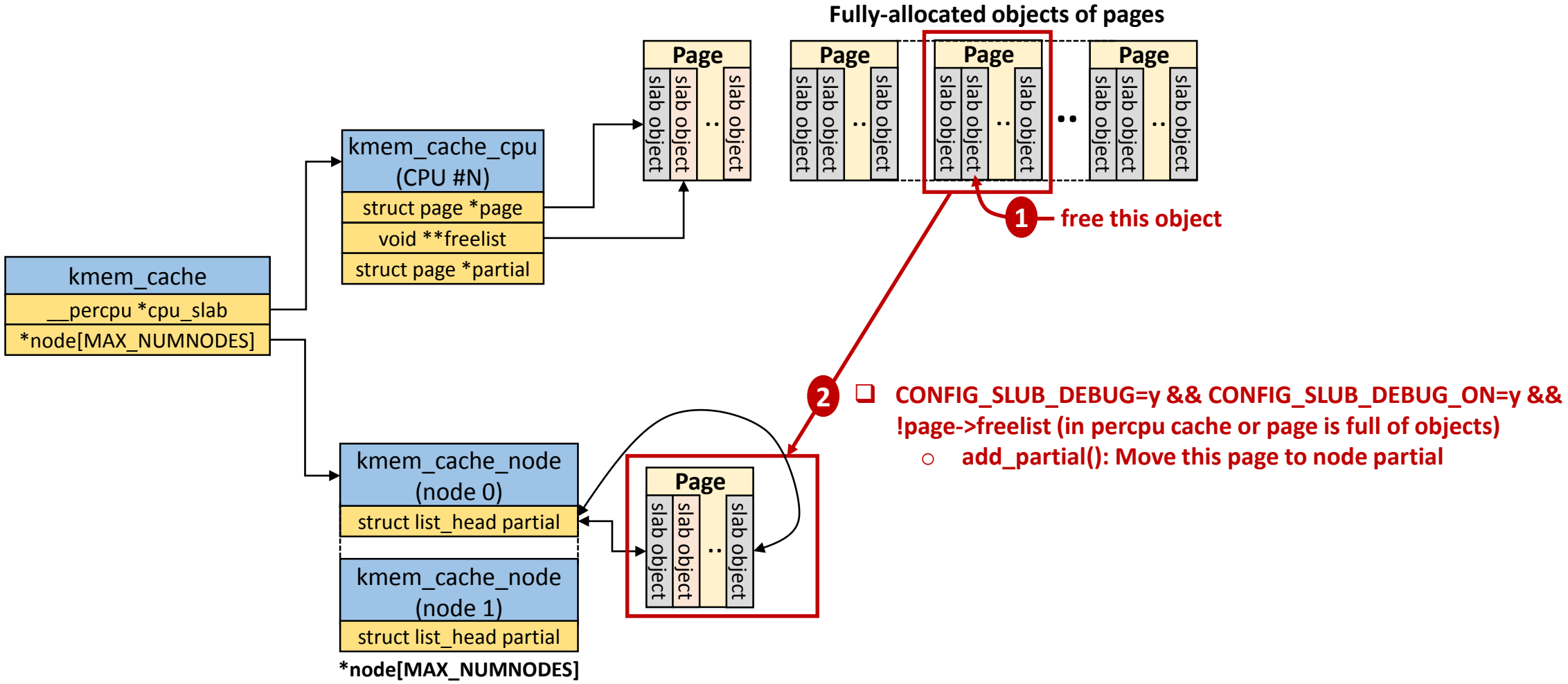
slowpath #1:put_cpu_partial() – Case 2 (1/2)



slowpath #1:put_cpu_partial() – Case 2 (2/2)

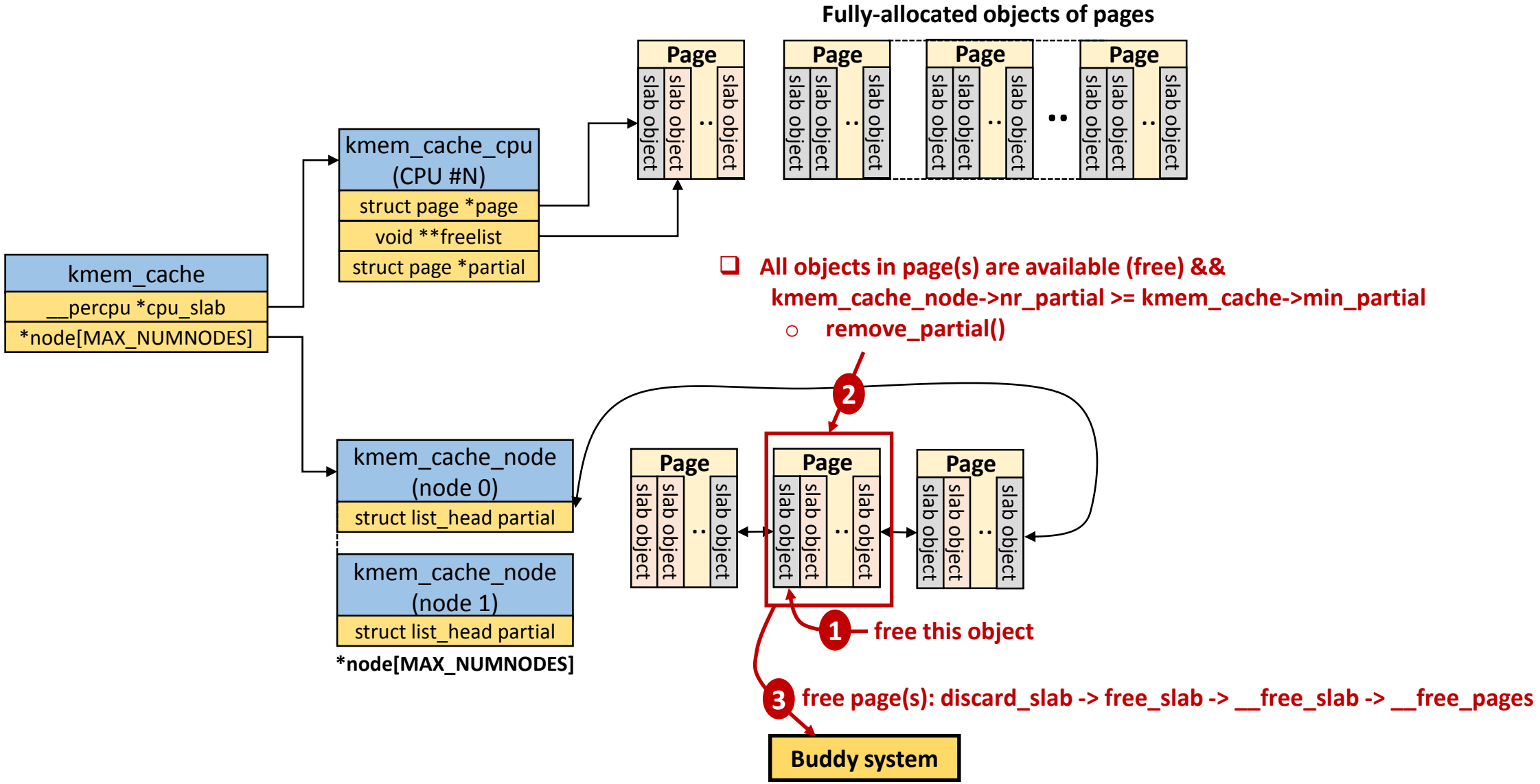


slowpath #2: add_partial() – Move a page to node partial



- CONFIG_SLUB_DEBUG=y && CONFIG_SLUB_DEBUG_ON=y && !page->freelist (in percpu cache or page is full of objects)
  - add_partial(): Move this page to node partial

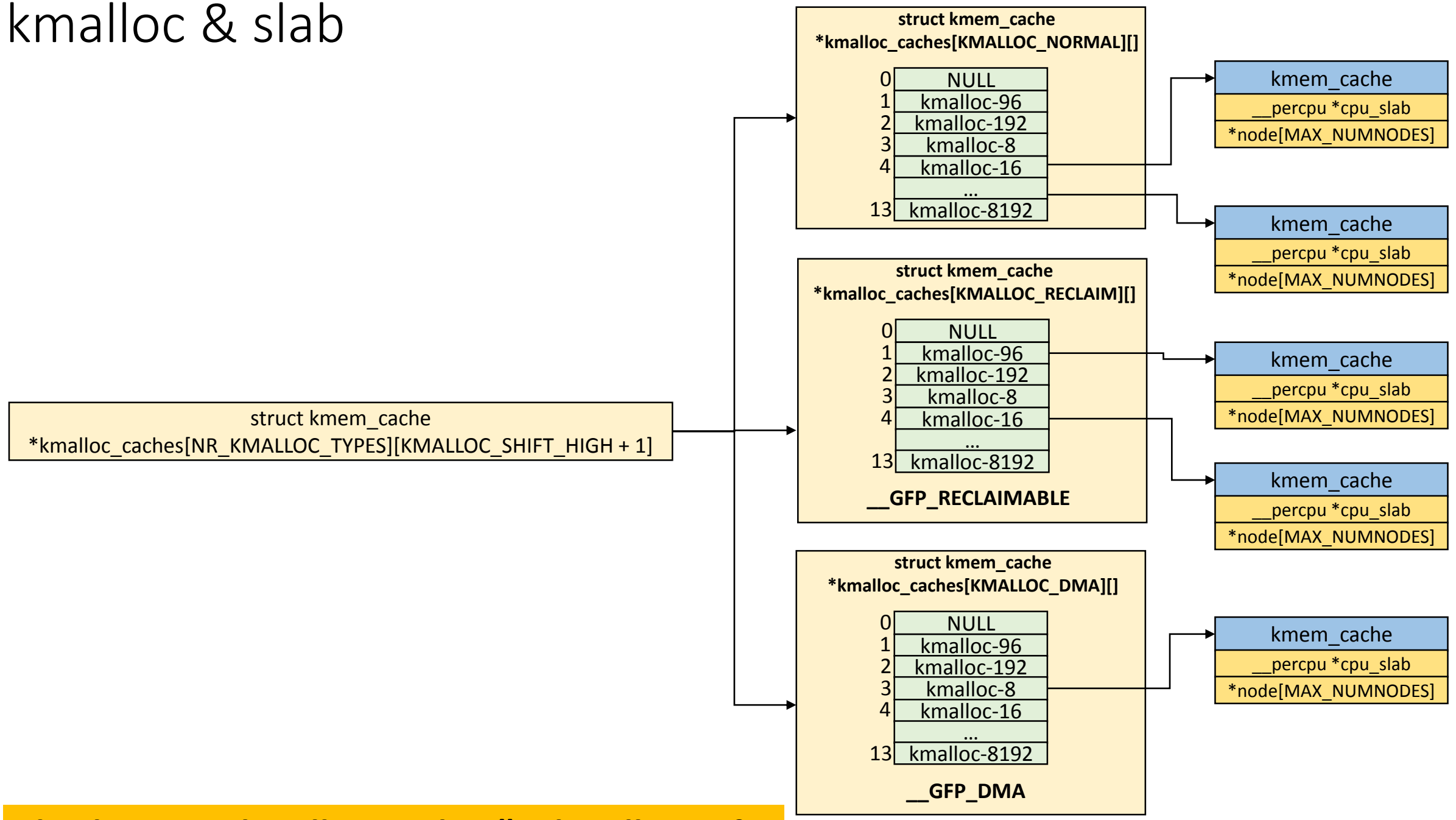
slowpath #3: remove_partial() and discard_slab()



kmalloc & slab

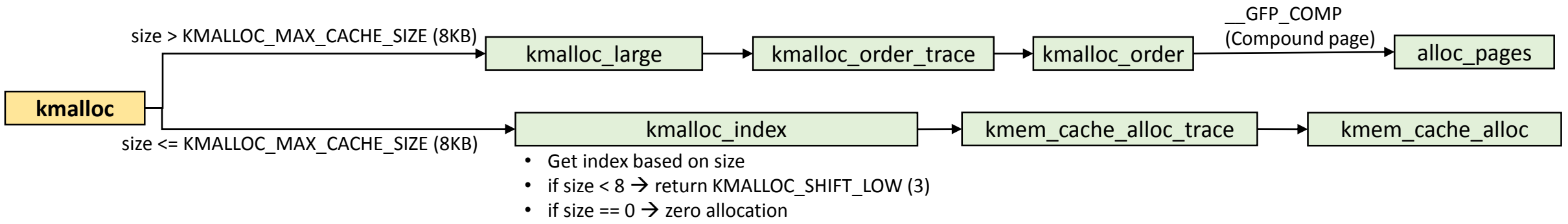
- Call path
- Compound page

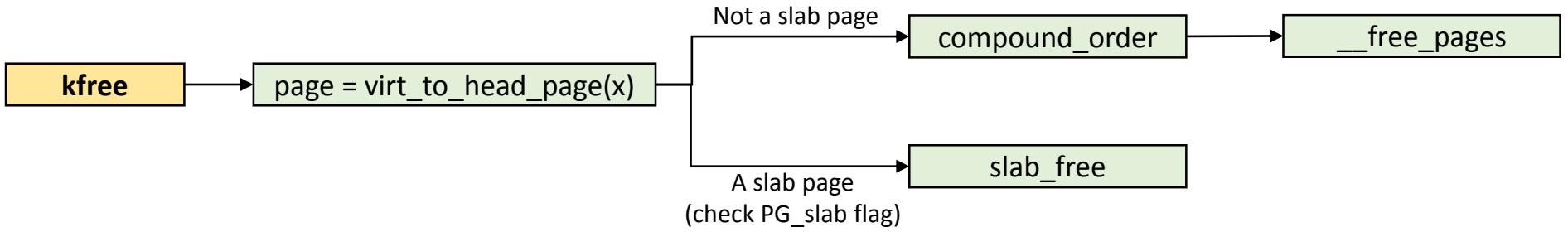
kmalloc & slab



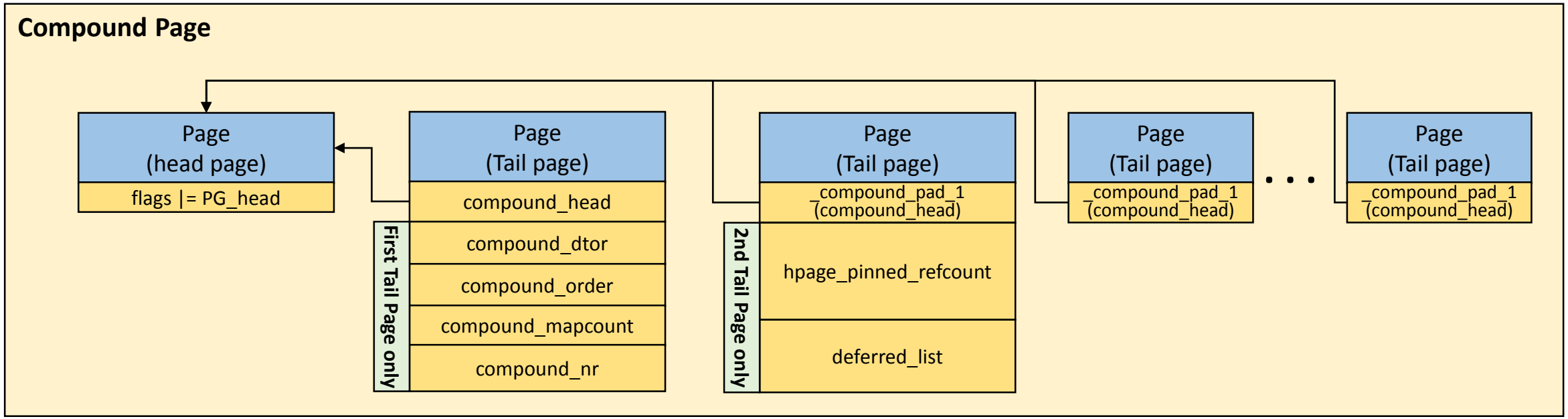
Check create_kmalloc_caches() &kmalloc_info

kmalloc() & kfree()





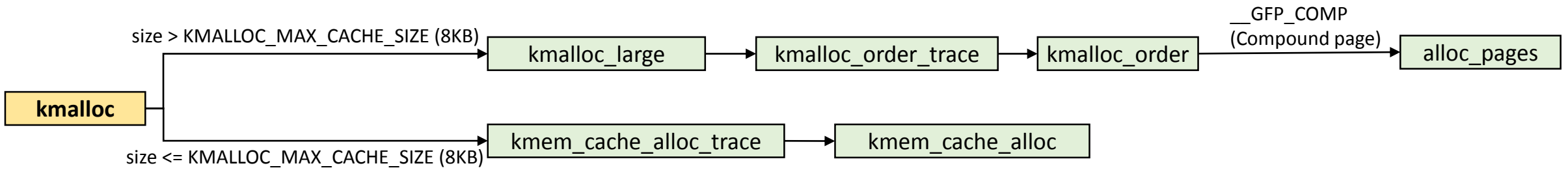
Compound page



Compound page – Use Cases

- Mainly used in huge page
- kmalloc: allocation size > 8192 bytes
  - Check kmalloc_order()

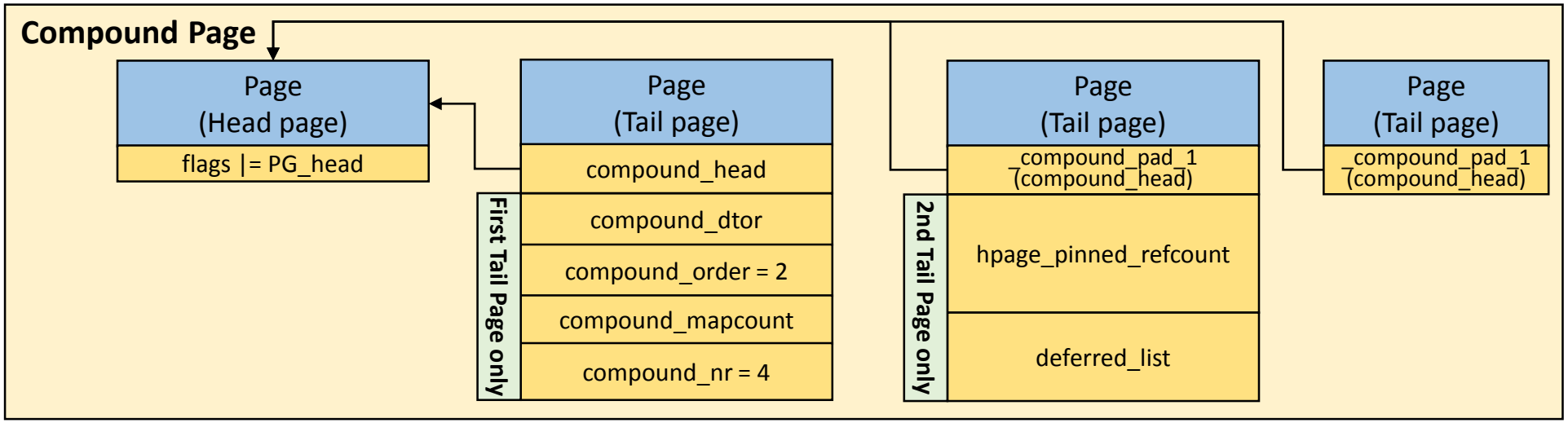
Compound page




```
(gdb) bt
#0  kmalloc_order (size=size@entry=8195, flags=<optimized out>,
    flags@entry=3264, order=order@entry=2)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slab_common.c
:838
#1  0xffffffff81b02315 in kmalloc_order_trace (order=2, flags=3264, size=8195)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/sl
ab.h:481
#2  kmalloc_large (flags=3264, size=8195)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/sl
ab.h:481
#3  kmalloc (flags=3264, size=8195)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/sl
ab.h:545
#4  unpack_to_rootfs (
    buf=0xffffffff81bfbf50 "070701000002D1000041ED", '0' <repeats 23 times>, "26
09E6D35", '0' <repeats 15 times>, "300000001", '0' <repeats 23 times>, "40000000
0dev", len=512)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/init/initramfs.c
:459
#5  0xffffffff81b02cd9 in populate_rootfs ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/init/initramfs.c
:608
```

Allocation size = 8195
order = 2 → 4 Pages

Compound page



Head Page

```
(gdb) bt 2
#0  kmalloc_order (size=size@entry=8195, flags=<optimized out>,
    flags@entry=3264, order=order@entry=2)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/slab_common.c
:838
#1  0xffffffff81b02315 in kmalloc_order_trace (order=2, flags=3264, size=8195)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/sl
ab.h:481
(More stack frames follow...)
(gdb) p page
$50 = (struct page *) 0xffffea0009035b00
(gdb) p /x page->flags
$51 = 0x5fffc000010000
(gdb) p /x PG_head
$52 = 0x10
```

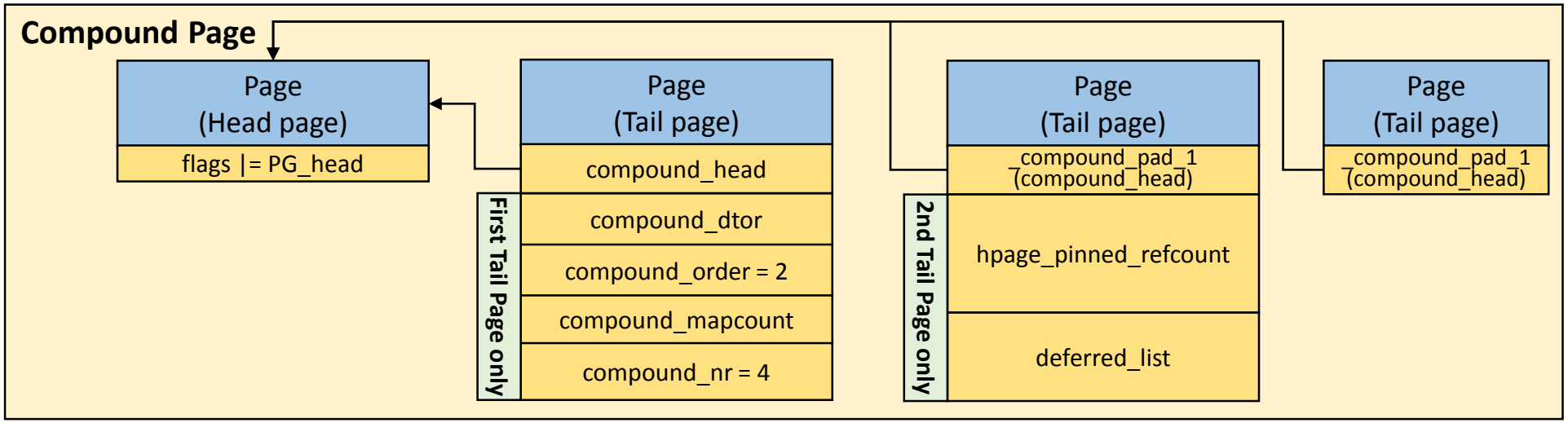
First Tail Page

```
(gdb) p /x (page + 1)->compound_head
$40 = 0xffffea0009035b01
(gdb) p /x (page + 1)->compound_order
$41 = 0x2
(gdb) p /x (page + 1)->compound_nr
$42 = 0x4
```

Second Tail Page

```
(gdb) p /x (page + 2)->compound_head
$44 = 0xffffea0009035b01
(gdb) p /x (page + 2)->_compound_pad_1
$45 = 0xffffea0009035b01
(gdb) p /x (page + 2)->hpage_pinned_refcount
$46 = {
  counter = 0x0
}
(gdb) p /x (page + 2)->deferred_list
$47 = {
  next = 0xdead000000000400,
  prev = 0x0
}
```

Compound page



First Tail Page

```
(gdb) p /x (page + 1)->compound_head
$40 = 0xffffea0009035b01
(gdb) p /x (page + 1)->compound_order
$41 = 0x2
(gdb) p /x (page + 1)->compound_nr
$42 = 0x4
```

Second Tail Page

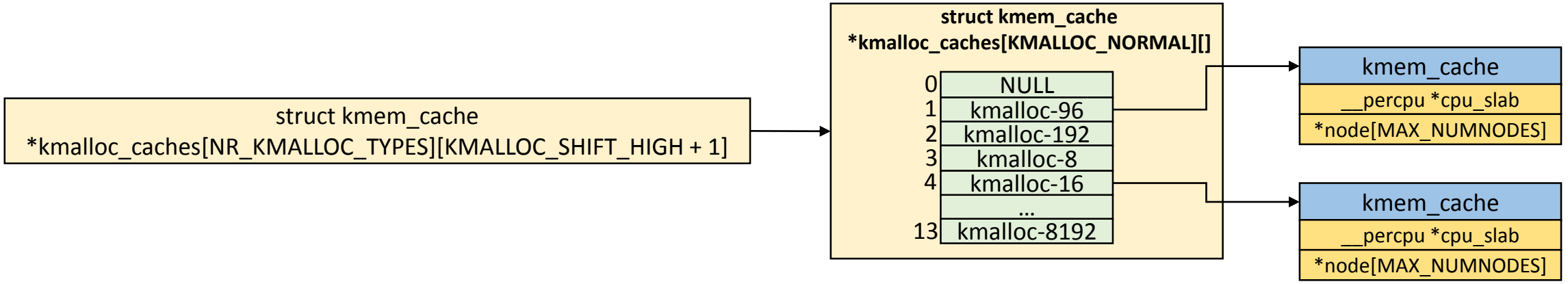
```
(gdb) p /x (page + 2)->compound_head
$44 = 0xffffea0009035b01
(gdb) p /x (page + 2)->_compound_pad_1
$45 = 0xffffea0009035b01
(gdb) p /x (page + 2)->hpage_pinned_refcount
$46 = {
  counter = 0x0
}
(gdb) p /x (page + 2)->deferred_list
$47 = {
  next = 0xdead000000000400,
  prev = 0x0
}
```

bit 0: purpose

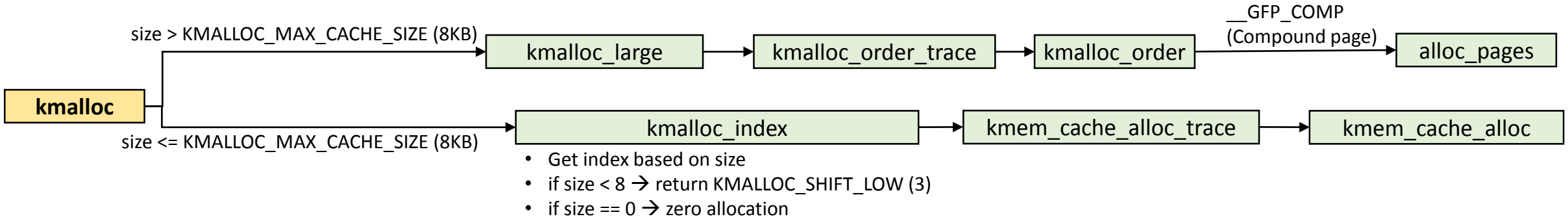
```
static inline struct page *compound_head(struct page *page)
{
        unsigned long head = READ_ONCE(page->compound_head);

        if (unlikely(head & 1))
                return (struct page *) (head - 1);
        return page;
}

static __always_inline int PageTail(struct page *page)
{
        return READ_ONCE(page->compound_head) & 1;
}
```

include/linux/page-flags.h

Q1: [SLUB] kmalloc: allocation size = 0, 1, 2, or 4 → workable?



minimum kmalloc cache size = 8 bytes



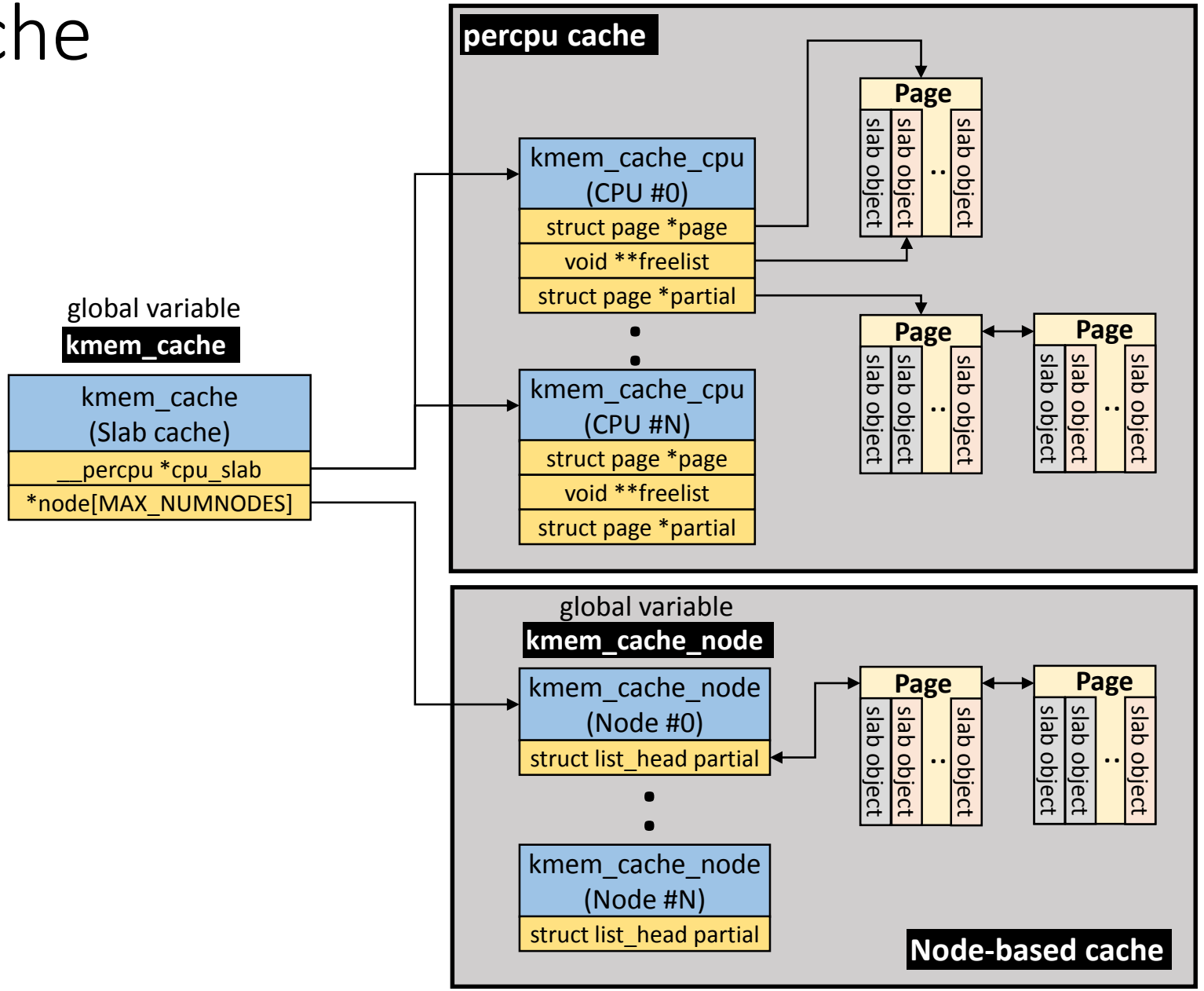
- if size < 8 → return KMALLOC_SHIFT_LOW (3)
- if size == 0 → zero allocation

Reference

- https://www.cnblogs.com/LoyenWang/p/11922887.html

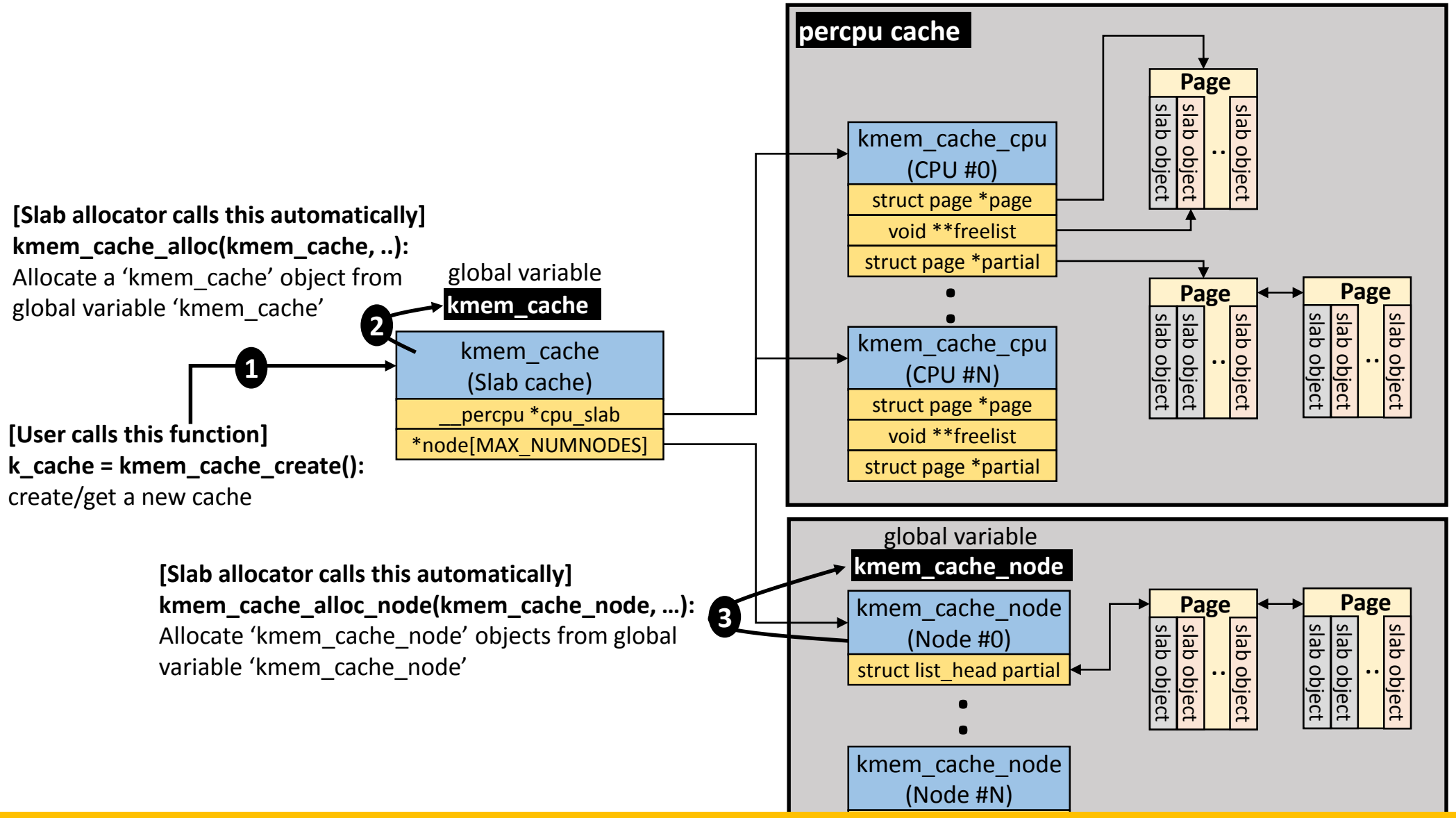
Backup

Slab Cache



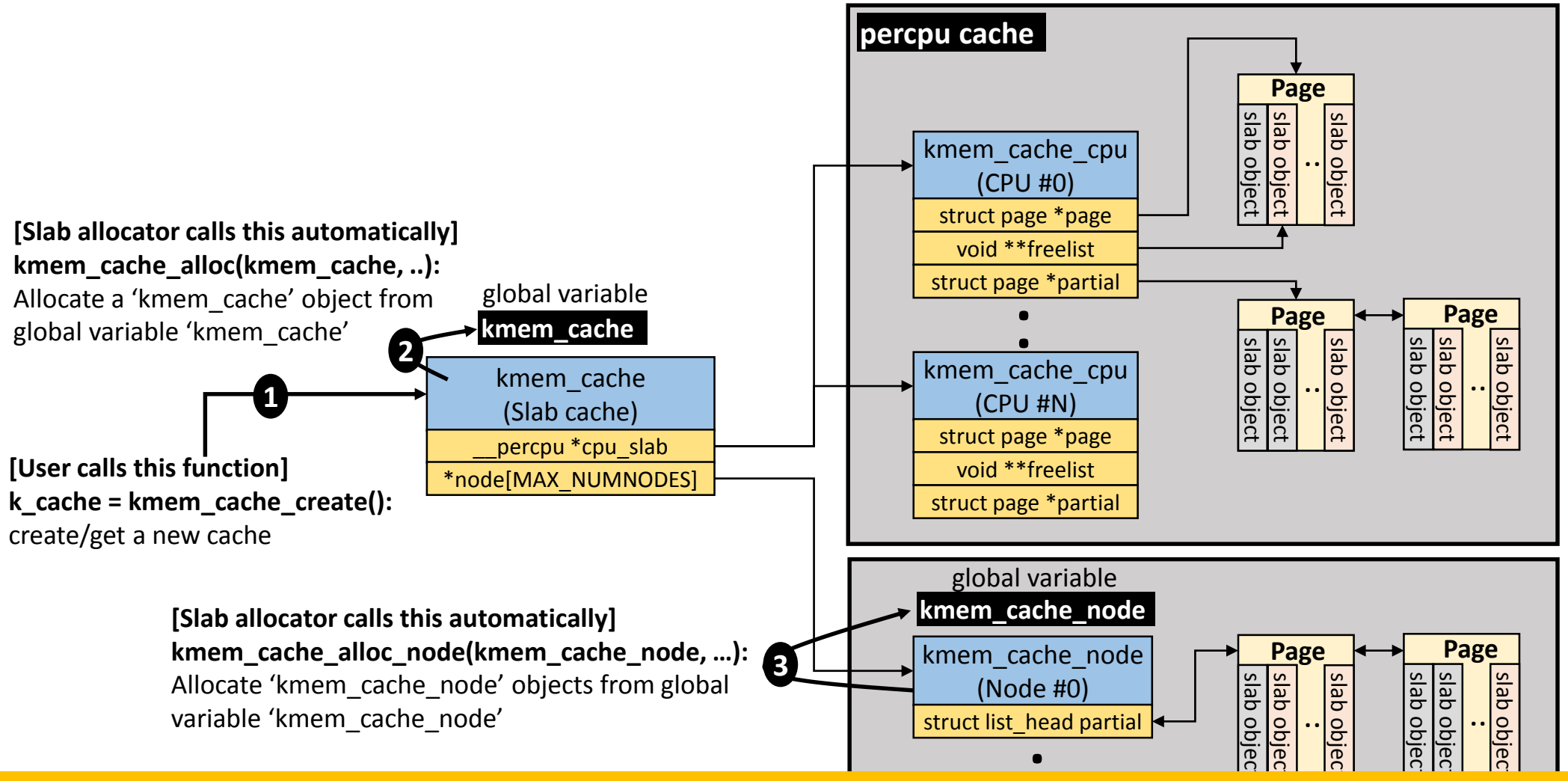
Node-based cache: freelist concept is implemented in page struct (not shown)

Slab Allocator Initialization: “Chicken or the egg” problem



Who initializes global variables ‘kmem_cache’ and ‘kmem_cache_node’?

Slab Allocator Initialization: “Chicken or the egg” problem



Slab allocator: statically initialized ‘kmem_cache_boot’
Slob allocator: statically initialized ‘kmem_cache_boot’
Slub allocator: static local variables ‘boot_kmem_cache’ and ‘boot_kmem_cache_node’

Slab Allocator Initialization: “Chicken or the egg” problem

```
void __init proc_caches_init(void)
{
        unsigned int mm_size;

        sighand_cachep = kmem_cache_create("sighand_cache",
                        sizeof(struct sighand_struct), 0,
                        SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_TYPESAFE_BY_RCU|
                        SLAB_ACCOUNT, sighand_ctor);
+-- 29 lines: signal_cachep = kmem_cache_create("signal_cache",-------------
}

kernel/fork.c                                                  2771,1
```