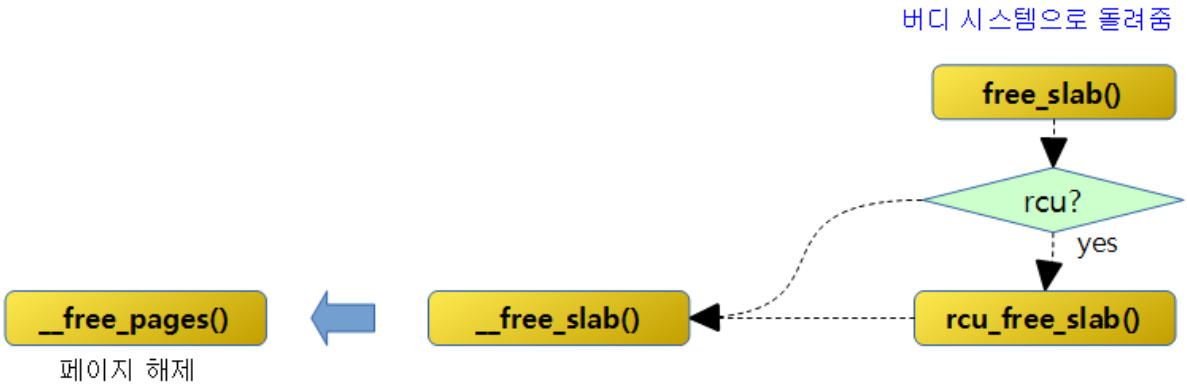
Slub Memory Allocator -10-

2016-06-07 (http://jake.dothome.co.kr/slub-slub-free/) Moon Young-il (http://jake.dothome.co.kr/author/admin/) Linux Kernel (http://jake.dothome.co.kr/category/linux/)

<kernel v5.0>

Release slap pages

The following diagram shows the flow of the slub page to be released from the free_slab() function to return to the buddy system.



(http://jake.dothome.co.kr/wp-content/uploads/2016/06/free_slab-1.png)

free_slab()

mm/slub.c

```
1 static void free_slab(struct kmem_cache *s, struct page *page)
2 {
3         if (unlikely(s->flags & SLAB_TYPESAFE_BY_RCU)) {
4                 call_rcu(&page->rcu_head, rcu_free_slab);
5         } else
6                 __free_slab(s, page);
7 }
```

Dismiss the slap page and return it to the buddy system.

- In line 3~4 of code, there is a small chance that lock-less RCU will be used to release the slap and reclaim it to the buddy system.
- In line 5~6 of the code, immediately reclaim the slap page to the buddy system.

rcu_free_slab()

mm/slub.c

```
1 static void rcu_free_slab(struct rcu_head *h)
2 {
```

```
3           struct page *page = container_of(h, struct page, rcu_head);
4
5           __free_slab(page->slab_cache, page);
6   }
```

Reclaim the slap page to the buddy system. This function is called through the RCU.

__free_slab()

mm/slub.c

```
01  static void __free_slab(struct kmem_cache *s, struct page *page)
02  {
03          int order = compound_order(page);
04          int pages = 1 << order;
05
06          if (s->flags & SLAB_CONSISTENCY_CHECKS) {
07                  void *p;
08
09                  slab_pad_check(s, page);
10                  for_each_object(p, s, page_address(page),
11                                                  page->objects)
12                          check_object(s, page, p, SLUB_RED_INACTIVE);
13          }
14
15          mod_lruvec_page_state(page,
16                  (s->flags & SLAB_RECLAIM_ACCOUNT) ?
17                  NR_SLAB_RECLAIMABLE : NR_SLAB_UNRECLAIMABLE,
18                  -pages);
19
20          __ClearPageSlabPfmemalloc(page);
21          __ClearPageSlab(page);
22
23          page->mapping = NULL;
24          if (current->reclaim_state)
25                  current->reclaim_state->reclaimed_slab += pages;
26          memcg_uncharge_slab(page, order, s);
27          __free_pages(page, order);
28  }
```

Dismiss the slap page and retrieve it to the buddy system.

- In line 3~4 of code, the slab page can be configured as a compound page. Therefore, we know the corresponding order value and the number of pages.
- If you used the SLAB_CONSISTENCY_CHECKS flag in lines 6~13 of the code, perform the consistency check on the slab page as follows:
  - Check the padding for the slab page.
  - Check the red-zone of each slab object to see if it is SLUB_RED_INACTIVE (0xbb).
- In code lines 15~18, decrement the NR_SLAB_RECLAIMABLE counter if it is a reclaimable slap cache, and the NR_SLAB_UNRECLAIMABLE counter by the number of pages if it is not.
- Clear PG_active and PG_slab on lines 20~23 of the code and unlink the page mapping.
- If the reclamation status of the current task is set in code lines 24~25, add the reclaimed_slab by the number of pages.
- In line 26 of the code, the memory cgroup tells us that the slab page has been recalled, so that it is reduced in counting.
- At line 27 of code, the buddy system retrieves the number of pages in order.

memcg_uncharge_slab()

mm/slab.h

```
static __always_inline void memcg_uncharge_slab(struct page *page, int o
rder,
                                                struct kmem_cache *s)
{
        if (!memcg_kmem_enabled())
                return;
        memcg_kmem_uncharge(page, order);
}
```

If memcg_kmem is enabled while using the CONFIG_MEMCG_KMEM kernel options, the memory cgroup will notify you that the slab page has been recalled and reduce it from counting. However, if it is a root cache, it will not be processed and will exit the routine.

consultation

- Slab Memory Allocator -1- (Structure) (http://jake.dothome.co.kr/slub/) | Qc
- Slab Memory Allocator -2- (Initialize Cache) (http://jake.dothome.co.kr/kmem_cache_init) | Qc
- Slub Memory Allocator -3- (Create Cache) (http://jake.dothome.co.kr/slub-cache-create) | Qc
- Slub Memory Allocator -4- (Calculate Order) (http://jake.dothome.co.kr/slub-order) | Qc
- Slub Memory Allocator -5- | (http://jake.dothome.co.kr/slub-slub-alloc) Qc
- Slub Memory Allocator -6- (Assign Object) (http://jake.dothome.co.kr/slub-object-alloc) | Qc
- Slub Memory Allocator -7- (Object Unlocked) (http://jake.dothome.co.kr/slub-object-free) | Qc
- Slub Memory Allocator -8- (Drain/Flash Cache) (http://jake.dothome.co.kr/slub-drain-flush-cache) | Qc
- Slub Memory Allocator -9- (Cache Shrink) (http://jake.dothome.co.kr/slub-cache-shrink) | Qc
- Slub Memory Allocator -10- | (http://jake.dothome.co.kr/slub-slub-free) Sentence C – Current post
- Slub Memory Allocator -11- (Clear Cache (http://jake.dothome.co.kr/slub-cache-destroy)) | Qc
- Slub Memory Allocator -12- (Debugging Slub) (http://jake.dothome.co.kr/slub-debug) | Qc
- Slub Memory Allocator -13- (slabinfo) (http://jake.dothome.co.kr/slub-slabinfo) | 문c

---

LEAVE A COMMENT

Your email will not be published. Required fields are marked with *

Comments

name *

email *

Website

WRITE A COMMENT

‹ Slub Memory Allocator -6- (Assign Object) (http://jake.dothome.co.kr/slub-object-alloc/)

Slub Memory Allocator -7- (Object Unlocked) › (http://jake.dothome.co.kr/slub-object-free/)

Munc Blog (2015 ~ 2024)