
Subject: [PATCH v4 11/19] sl[au]b: always get the cache from its page in kfree
Posted by [Glauber Costa](#) on Fri, 12 Oct 2012 13:41:05 GMT
[View Forum Message](#) <> [Reply to Message](#)

struct page already have this information. If we start chaining caches, this information will always be more trustworthy than whatever is passed into the function

A parent pointer is added to the slab structure, so we can make sure the freeing comes from either the right slab, or from its rightful parent.

[v3: added parent testing with VM_BUG_ON]
[v4: make it faster when kmemcg not in use]

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: Michal Hocko <mhocko@suse.cz>
CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Johannes Weiner <hannes@cmpxchg.org>
CC: Suleiman Souhlal <suleiman@google.com>
CC: Tejun Heo <tj@kernel.org>

include/linux/memcontrol.h | 4 ++++
mm/slab.c | 17 ++++++++
mm/slab.h | 13 ++++++++
mm/slub.c | 14 ++++++++
4 files changed, 45 insertions(+), 3 deletions(-)

```
diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index c7886aa..4c94182 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -536,6 +536,10 @@ static inline void sock_release_memcg(struct sock *sk)
{
}

+static inline bool memcg_kmem_enabled(void)
+{
+ return false;
+}
static inline bool
memcg_kmem_newpage_charge(gfp_t gfp, struct mem_cgroup **memcg, int order)
{
diff --git a/mm/slab.c b/mm/slab.c
```

index 98b3460..6f22067 100644

--- a/mm/slab.c

+++ b/mm/slab.c

@@ -3911,9 +3911,24 @@ EXPORT_SYMBOL(__kmallocc);

* Free an object which was previously allocated from this

* cache.

*/

-void kmem_cache_free(struct kmem_cache *cachep, void *objp)

+void kmem_cache_free(struct kmem_cache *s, void *objp)

{

unsigned long flags;

+ struct kmem_cache *cachep;

+

+ /*

+ * When kmemcg is not being used, both assignments should return the

+ * same value. but we don't want to pay the assignment price in that

+ * case. If it is not compiled in, the compiler should be smart enough

+ * to not do even the assignment. In that case, slab_equal_or_root

+ * will also be a constant.

+ */

+ if (memcg_kmem_enabled()) {

+ cachep = virt_to_cache(objp);

+ VM_BUG_ON(!slab_equal_or_root(cachep, s));

+ } else

+ cachep = s;

+

local_irq_save(flags);

debug_check_no_locks_freed(objp, cachep->object_size);

diff --git a/mm/slab.h b/mm/slab.h

index c35ecce..b9b5f1f 100644

--- a/mm/slab.h

+++ b/mm/slab.h

@@ -108,6 +108,13 @@ static inline bool cache_match_memcg(struct kmem_cache *cachep,

return (is_root_cache(cachep) && !memcg) ||

(cachep->memcg_params->memcg == memcg);

}

+

+static inline bool slab_equal_or_root(struct kmem_cache *s,

+ struct kmem_cache *p)

+{

+ return (p == s) ||

+ (s->memcg_params && (p == s->memcg_params->root_cache));

+}

#else

static inline bool is_root_cache(struct kmem_cache *s)

{

@@ -119,5 +126,11 @@ static inline bool cache_match_memcg(struct kmem_cache *cachep,

```

{
    return true;
}
+
+static inline bool slab_equal_or_root(struct kmem_cache *s,
+    struct kmem_cache *p)
+{
+    return true;
+}
#endif
#endif
diff --git a/mm/slub.c b/mm/slub.c
index 05aefe2..6e1a90f 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -2609,9 +2609,19 @@ redo:

    void kmem_cache_free(struct kmem_cache *s, void *x)
    {
-    struct page *page;
+    struct page *page = virt_to_head_page(x);

-    page = virt_to_head_page(x);
+    /*
+     * When kmemcg is not being used, both assignments should return the
+     * same value. but we don't want to pay the assignment price in that
+     * case. If it is not compiled in, the compiler should be smart enough
+     * to not do even the assignment. In that case, slab_equal_or_root
+     * will also be a constant.
+     */
+    if (memcg_kmem_enabled()) {
+        VM_BUG_ON(!slab_equal_or_root(page->slab, s));
+        s = page->slab;
+    }

    if (kmem_cache_debug(s) && page->slab != s) {
        pr_err("kmem_cache_free: Wrong slab cache. %s but object"
--
1.7.11.4

```
