Subject: [PATCH v5 04/18] slab: don't preemptively remove element from list in cache destroy
Posted by Glauber Costa on Fri, 19 Oct 2012 14:20:28 GMT
View Forum Message <> Reply to Message

After the slab/slub/slob merge, we are deleting the element from the
slab_cache lists, and then if the destruction fail, we add it back
again. This behavior was present in some caches, but not in others, if
my memory doesn't fail me.

I, however, see no reason why we need to do so, since we are now locked
during the whole deletion (which wasn't necessarily true before). I
propose a simplification in which we delete it only when there is no
more going back, so we don't need to add it again.

Signed-off-by: Glauber Costa <glommer@parallels.com>
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>

```
 mm/slab_common.c | 5 ++---
 1 file changed, 2 insertions(+), 3 deletions(-)

diff --git a/mm/slab_common.c b/mm/slab_common.c
index 1ee1d6f..bf4b4f1 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -174,16 +174,15 @@ void kmem_cache_destroy(struct kmem_cache *s)
  mutex_lock(&slab_mutex);
  s->refcount--;
  if (!s->refcount) {
-  list_del(&s->list);
-
   if (!__kmem_cache_shutdown(s)) {
    if (s->flags & SLAB_DESTROY_BY_RCU)
     rcu_barrier();

+   list_del(&s->list);
+
    kfree(s->name);
    kmem_cache_free(kmem_cache, s);
   } else {
-   list_add(&s->list, &slab_caches);
    printk(KERN_ERR "kmem_cache_destroy %s: Slab cache still has objects\n",
     s->name);
    dump_stack();
--
1.7.11.7
```