

---

Subject: [PATCH 08/10] memcg/sl[au]b Track all the memcg children of a kmem\_cache.

Posted by [Glauber Costa](#) on Wed, 25 Jul 2012 14:38:19 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

This enables us to remove all the children of a kmem\_cache being destroyed, if for example the kernel module it's being used in gets unloaded. Otherwise, the children will still point to the destroyed parent.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>

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 <jannes@cmpxchg.org>

---

```
include/linux/memcontrol.h |  1 +
include/linux/slab.h      |  1 +
mm/memcontrol.c          | 16 ++++++
mm/slab_common.c         | 31 ++++++++++++++++++++++++
4 files changed, 48 insertions(+), 1 deletion(-)
```

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h

index 247019f..f8d63f2 100644

--- a/include/linux/memcontrol.h

+++ b/include/linux/memcontrol.h

```
@@ -426,6 +426,7 @@ void memcg_release_cache(struct kmem_cache *cachep);
 struct kmem_cache *
 __memcg_kmem_get_cache(struct kmem_cache *cachep, gfp_t gfp);
 void mem_cgroup_destroy_cache(struct kmem_cache *cachep);
+void mem_cgroup_remove_child_kmem_cache(struct kmem_cache *cachep, int id);
#else
 static inline void memcg_register_cache(struct mem_cgroup *memcg,
     struct kmem_cache *s)
```

diff --git a/include/linux/slab.h b/include/linux/slab.h

index b9310a4..3cfcd784 100644

--- a/include/linux/slab.h

+++ b/include/linux/slab.h

```
@@ -189,6 +189,7 @@ struct mem_cgroup_cache_params {
     bool dead;
     atomic_t nr_pages;
     struct list_head destroyed_list; /* Used when deleting memcg cache */
+    struct list_head sibling_list;
 };
#endif
```

```

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 1231d86..d92d963 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -586,8 +586,11 @@ static struct kmem_cache *kmem_cache_dup(struct mem_cgroup
 *memcg,
new = kmem_cache_create_memcg(memcg, name, s->object_size, s->align,
                               (s->flags & ~SLAB_PANIC), s->ctor, s);
- if (new)
+ if (new) {
    new->allocflags |= __GFP_KMEMCG;
+ list_add(&new->memcg_params.sibling_list,
+ &s->memcg_params.sibling_list);
+ }

kfree(name);
return new;
@@ -600,6 +603,7 @@ void memcg_register_cache(struct mem_cgroup *memcg, struct
kmem_cache *cachep)
int id = -1;

INIT_LIST_HEAD(&cachep->memcg_params.destroyed_list);
+ INIT_LIST_HEAD(&cachep->memcg_params.sibling_list);

if (!memcg)
    id = ida_simple_get(&cache_types, 0, MAX_KMEM_CACHE_TYPES,
@@ -611,6 +615,9 @@ void memcg_release_cache(struct kmem_cache *cachep)
{
    if (cachep->memcg_params.id != -1)
        ida_simple_remove(&cache_types, cachep->memcg_params.id);
+ else
+ list_del(&cachep->memcg_params.sibling_list);
+
}

/*
@@ -944,6 +951,13 @@ static void memcg_create_cache_enqueue(struct mem_cgroup
*memcg,
    schedule_work(&memcg_create_cache_work);
}

+void mem_cgroup_remove_child_kmem_cache(struct kmem_cache *cachep, int id)
+{
+ mutex_lock(&memcg_cache_mutex);
+ cachep->memcg_params.memcg->slabs[id] = NULL;
+ mutex_unlock(&memcg_cache_mutex);
+}

```

```

+
/*
 * Return the kmem_cache we're supposed to use for a slab allocation.
 * We try to use the current memcg's version of the cache.
diff --git a/mm/slab_common.c b/mm/slab_common.c
index 562146b..6504557 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@@ -189,8 +189,39 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t
size, size_t align
}
EXPORT_SYMBOL(kmem_cache_create);

+static void kmem_cache_destroy_memcg_children(struct kmem_cache *s)
+{
+ifdef CONFIG_MEMCG_KMEM
+ struct kmem_cache *c;
+ struct mem_cgroup_cache_params *p, *tmp;
+ int id = s->memcg_params.id;
+
+ if (id == -1)
+ return;
+
+ mutex_lock(&slab_mutex);
+ list_for_each_entry_safe(p, tmp,
+ &s->memcg_params.sibling_list, sibling_list) {
+ c = container_of(p, struct kmem_cache, memcg_params);
+ if (WARN_ON(c == s))
+ continue;
+
+ mutex_unlock(&slab_mutex);
+ BUG_ON(c->memcg_params.id != -1);
+ mem_cgroup_remove_child_kmem_cache(c, id);
+ kmem_cache_destroy(c);
+ mutex_lock(&slab_mutex);
+ }
+ mutex_unlock(&slab_mutex);
+endif /* CONFIG_MEMCG_KMEM */
+}
+
void kmem_cache_destroy(struct kmem_cache *s)
{
+
/* Destroy all the children caches if we aren't a memcg cache */
+ kmem_cache_destroy_memcg_children(s);
+
get_online_cpus();
mutex_lock(&slab_mutex);

```

```
list_del(&s->list);
```

```
--  
1.7.10.4
```

---