

---

Subject: [PATCH v2 10/11] memcg: allow a memcg with kmem charges to be destructed.

Posted by [Glauber Costa](#) on Thu, 09 Aug 2012 13:01:18 GMT

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

---

Because the ultimate goal of the kmem tracking in memcg is to track slab pages as well, we can't guarantee that we'll always be able to point a page to a particular process, and migrate the charges along with it - since in the common case, a page will contain data belonging to multiple processes.

Because of that, when we destroy a memcg, we only make sure the destruction will succeed by discounting the kmem charges from the user charges when we try to empty the cgroup.

Signed-off-by: Glauber Costa <[glommer@parallels.com](mailto:glommer@parallels.com)>

Acked-by: Kamezawa Hiroyuki <[kamezawa.hiroyu@jp.fujitsu.com](mailto:kamezawa.hiroyu@jp.fujitsu.com)>

CC: Christoph Lameter <[cl@linux.com](mailto:cl@linux.com)>

CC: Pekka Enberg <[penberg@cs.helsinki.fi](mailto:penberg@cs.helsinki.fi)>

CC: Michal Hocko <[mhocko@suse.cz](mailto:mhocko@suse.cz)>

CC: Johannes Weiner <[hannes@cmpxchg.org](mailto:hannes@cmpxchg.org)>

CC: Suleiman Souhlal <[suleiman@google.com](mailto:suleiman@google.com)>

---

mm/memcontrol.c | 17 ++++++-----

1 file changed, 16 insertions(+), 1 deletion(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c

index 3d30b79..7c1ea49 100644

--- a/mm/memcontrol.c

+++ b/mm/memcontrol.c

@@ -649,6 +649,11 @@ static void disarm\_kmem\_keys(struct mem\_cgroup \*memcg)

{

  if (test\_bit(KMEM\_ACCOUNTED\_THIS, &memcg->kmem\_accounted))

    static\_key\_slow\_dec(&memcg\_kmem\_enabled\_key);

+ /\*

+ \* This check can't live in kmem destruction function,

+ \* since the charges will outlive the cgroup

+ \*/

+ WARN\_ON(res\_counter\_read\_u64(&memcg->kmem, RES\_USAGE) != 0);

}

#else

  static void disarm\_kmem\_keys(struct mem\_cgroup \*memcg)

@@ -4005,6 +4010,7 @@ static int mem\_cgroup\_force\_empty(struct mem\_cgroup \*memcg, bool free\_all)

  int node, zid, shrink;

  int nr\_retries = MEM\_CGROUP\_RECLAIM\_RETRIES;

  struct cgroup \*cgrp = memcg->css.cgroup;

+ u64 usage;

```
css_get(&memcg->css);

@@ -4038,8 +4044,17 @@ move_account:
    mem_cgroup_end_move(memcg);
    memcg_oom_recover(memcg);
    cond_resched();
+ /*
+ * Kernel memory may not necessarily be trackable to a specific
+ * process. So they are not migrated, and therefore we can't
+ * expect their value to drop to 0 here.
+ *
+ * having res filled up with kmem only is enough
+ */
+ usage = res_counter_read_u64(&memcg->res, RES_USAGE) -
+ res_counter_read_u64(&memcg->kmem, RES_USAGE);
/* "ret" should also be checked to ensure all lists are empty. */
- } while (res_counter_read_u64(&memcg->res, RES_USAGE) > 0 || ret);
+ } while (usage > 0 || ret);
out:
css_put(&memcg->css);
return ret;
--
```

## 1.7.11.2

---