Subject: [PATCH v2 11/13] memcg: Handle bypassed kernel memory charges.
Posted by Suleiman Souhlal on Fri, 09 Mar 2012 20:39:14 GMT
View Forum Message <> Reply to Message

When __mem_cgroup_try_charge() decides to bypass a slab charge
(because we are getting OOM killed or have a fatal signal pending),
we may end up with a slab that belongs to a memcg, but wasn't
charged to it. When we free such a slab page, we end up uncharging
it from the memcg, even though it was never charged, which may
lead to res_counter underflows.

To avoid this, when a charge is bypassed, we force the charge,
without checking for the bypass conditions or doing any reclaim.
This may cause the cgroup's usage to temporarily go above its limit.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>
---
 mm/memcontrol.c | 15 +++++++++++++--
 1 files changed, 13 insertions(+), 2 deletions(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 72e83af..9f5e9d8 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -5672,16 +5672,27 @@ memcg_charge_kmem(struct mem_cgroup *memcg, gfp_t gfp, long long delta)

  ret = 0;

- _memcg = memcg;
  if (memcg && !mem_cgroup_test_flag(memcg,
      MEMCG_INDEPENDENT_KMEM_LIMIT)) {
+ _memcg = memcg;
   ret = __mem_cgroup_try_charge(NULL, gfp, delta / PAGE_SIZE,
       &_memcg, may_oom);
   if (ret == -ENOMEM)
    return ret;
+ else if (ret == -EINTR) {
+   /*
+   * __mem_cgroup_try_charge() chose to bypass to root due
+   * to OOM kill or fatal signal.
+   * Since our only options are to either fail the
+   * allocation or charge it to this cgroup, force the
+   * change, going above the limit if needed.
+   */
+   ret = res_counter_charge_nofail(&memcg->res, delta,
+   &fail_res);
+ }

```
  }

- if (memcg && _memcg == memcg)
+ if (memcg)
    ret = res_counter_charge(&memcg->kmem, delta, &fail_res);

  return ret;
--
1.7.7.3
```