Subject: [PATCH][for -mm] Fix and Enhancements for memory cgroup [1/6] fix refcnt race in charge/uncharge
Posted by KAMEZAWA Hiroyuki on Tue, 09 Oct 2007 09:49:25 GMT
View Forum Message <> Reply to Message

The logic of uncharging is
 - decrement refcnt -> lock page cgroup -> remove page cgroup.
But the logic of charging is
 - lock page cgroup -> increment refcnt -> return.

Then, one charge will be added to a page_cgroup under being removed.
This makes no big trouble (like panic) but one charge is lost.

This patch add a test at charging to verify page_cgroup's refcnt is
greater than 0. If not, unlock and retry.

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

 mm/memcontrol.c | 9 +++++++--
 1 file changed, 7 insertions(+), 2 deletions(-)

Index: linux-2.6.23-rc8-mm2/mm/memcontrol.c
===
--- linux-2.6.23-rc8-mm2.orig/mm/memcontrol.c
+++ linux-2.6.23-rc8-mm2/mm/memcontrol.c
@@ -271,14 +271,19 @@ int mem_cgroup_charge(struct page *page,
   * to see if the cgroup page already has a page_cgroup associated
   * with it
   */
+retry:
  lock_page_cgroup(page);
  pc = page_get_page_cgroup(page);
  /*
   * The page_cgroup exists and the page has already been accounted
   */
  if (pc) {
- atomic_inc(&pc->ref_cnt);
- goto done;
+ if (unlikely(!atomic_inc_not_zero(&pc->ref_cnt))) {
+   /* this page is under being uncharge ? */
+   unlock_page_cgroup(page);
+   goto retry;
+ } else
+   goto done;
  }

  unlock_page_cgroup(page);

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers