

---

Subject: [PATCH][BUGFIX][for -mm] Misc fix for memory cgroup [3/5] --- add helper function for assignin page\_

Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 05:00:10 GMT

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

---

This patch adds following functions.

- clear\_page\_cgroup(page, pc)
- page\_cgroup\_assign\_new\_page\_group(page, pc)

Mainly for cleanup.

A manner "check page->cgroup again after lock\_page\_cgroup()" is implemented in straight way.

Changelog v1 -> v2:

- added comments.
- adder printk in mem\_cgroup\_uncharge() error path, but this is planned to be removed by other patch

Note:

- a comment in mem\_cgroup\_uncharge() will be removed by force-empty patch

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

mm/memcontrol.c | 100 ++++++-----  
1 file changed, 73 insertions(+), 27 deletions(-)

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

```
=====
--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c
+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c
@@ -162,6 +162,48 @@ static void __always_inline unlock_page_
    bit_spin_unlock(PAGE_CGROUP_LOCK_BIT, &page->page_cgroup);
}

+/*
+ * Tie new page_cgroup to struct page under lock_page_cgroup()
+ * This can fail if the page has been tied to a page_cgroup.
+ * If success, returns 0.
+ */
+static inline int
+page_cgroup_assign_new_page_cgroup(struct page *page, struct page_cgroup *pc)
+{
+    int ret = 0;
+
+    lock_page_cgroup(page);
+    if (!page_get_page_cgroup(page))
```

```

+ page_assign_page_cgroup(page, pc);
+ else /* A page is tied to other pc. */
+ ret = 1;
+ unlock_page_cgroup(page);
+ return ret;
+}
+
+/*
+ * Clear page->page_cgroup member under lock_page_cgroup().
+ * If given "pc" value is different from one page->page_cgroup,
+ * page->cgroup is not cleared.
+ * Returns a value of page->page_cgroup at lock taken.
+ * A can detect failure of clearing by following
+ * clear_page_cgroup(page, pc) == pc
+ */
+
+static inline struct page_cgroup *
+clear_page_cgroup(struct page *page, struct page_cgroup *pc)
+{
+ struct page_cgroup *ret;
+ /* lock and clear */
+ lock_page_cgroup(page);
+ ret = page_get_page_cgroup(page);
+ if (likely(ret == pc))
+ page_assign_page_cgroup(page, NULL);
+ unlock_page_cgroup(page);
+ return ret;
+}
+
+
+static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
+{
+ if (active)
@@ -260,7 +302,7 @@ int mem_cgroup_charge(struct page *page,
+ gfp_t gfp_mask)
+{
+ struct mem_cgroup *mem;
- struct page_cgroup *pc, *race_pc;
+ struct page_cgroup *pc;
+ unsigned long flags;
+ unsigned long nr_retries = MEM_CGROUP_RECLAIM_RETRIES;

@@ -354,24 +396,20 @@ noreclaim:
+ goto free_pc;
+}

- lock_page_cgroup(page);
- /*

```

```

- * Check if somebody else beat us to allocating the page_cgroup
- */
- race_pc = page_get_page_cgroup(page);
- if (race_pc) {
-     kfree(pc);
-     pc = race_pc;
-     atomic_inc(&pc->ref_cnt);
-     res_counter_uncharge(&mem->res, PAGE_SIZE);
-     css_put(&mem->css);
-     goto done;
- }
-
-     atomic_set(&pc->ref_cnt, 1);
-     pc->mem_cgroup = mem;
-     pc->page = page;
- page_assign_page_cgroup(page, pc);
+ if (page_cgroup_assign_new_page_cgroup(page, pc)) {
+     /*
+      * an another charge is added to this page already.
+      * we do take lock_page_cgroup(page) again and read
+      * page->cgroup, increment refcnt.... just retry is OK.
+      */
+     res_counter_uncharge(&mem->res, PAGE_SIZE);
+     css_put(&mem->css);
+     kfree(pc);
+     goto retry;
+ }

    spin_lock_irqsave(&mem->lru_lock, flags);
    list_add(&pc->lru, &mem->active_list);
@@ -422,17 +460,25 @@ void mem_cgroup_uncharge(struct page_cgr

    if (atomic_dec_and_test(&pc->ref_cnt)) {
        page = pc->page;
-     lock_page_cgroup(page);
-     mem = pc->mem_cgroup;
-     css_put(&mem->css);
-     page_assign_page_cgroup(page, NULL);
-     unlock_page_cgroup(page);
-     res_counter_uncharge(&mem->res, PAGE_SIZE);
-
-     spin_lock_irqsave(&mem->lru_lock, flags);
-     list_del_init(&pc->lru);
-     spin_unlock_irqrestore(&mem->lru_lock, flags);
-     kfree(pc);
+     /*
+      * get page->cgroup and clear it under lock.
+      */

```

```

+ if (clear_page_cgroup(page, pc) == pc) {
+   mem = pc->mem_cgroup;
+   css_put(&mem->css);
+   res_counter_uncharge(&mem->res, PAGE_SIZE);
+   spin_lock_irqsave(&mem->lru_lock, flags);
+   list_del_init(&pc->lru);
+   spin_unlock_irqrestore(&mem->lru_lock, flags);
+   kfree(pc);
+ } else {
+   /*
+    * Note: This will be removed when force-empty patch is
+    * applied. just show warning here.
+    */
+   printk(KERN_ERR "Race in mem_cgroup_uncharge() ?");
+   dump_stack();
+ }
+ }
+ }

```

---

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

---