Subject: [PATCH][for -mm] Fix and Enhancements for memory cgroup [3/6] add helper function for page_cgroup
Posted by KAMEZAWA Hiroyuki on Tue, 09 Oct 2007 09:51:32 GMT
View Forum Message <> Reply to Message

This patch adds follwoing functions.
- clear_page_cgroup(page, pc)
- page_cgroup_assign_new_page_group(page, pc)

Mainly for cleaunp.

A manner "check page->cgroup again after lock_page_cgroup()" is implemented in straight way.

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

```
 mm/memcontrol.c |   76 +++++++++++++++++++++++++++++++++++++++++++----------------------
 1 file changed, 49 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 +162,35 @@ static void __always_inline unlock_page_
  bit_spin_unlock(PAGE_CGROUP_LOCK_BIT, &page->page_cgroup);
 }

+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
+  ret = 1;
+ unlock_page_cgroup(page);
+ return ret;
+}
+
+
+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 +289,7 @@ 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;

@@ -353,24 +382,16 @@ 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)) {
+  /* race ... undo and retry */
+  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);
@@ -421,17 +442,18 @@ 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);
+  /*
+   * Obetaion 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);
+  }
  }
 }
```

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