Subject: [BUGFIX][RFC][PATCH][only -mm] FIX memory leak in memory cgroup vs. page migration [2/1] additional
Posted by KAMEZAWA Hiroyuki on Tue, 02 Oct 2007 10:12:17 GMT
View Forum Message <> Reply to Message

The patch I sent needs following fix, sorry.
Anyway, I'll repost good-version with reflected comments again.

Thanks,
 -Kame
=
fix-page-migration-under-memory-controller patch needs this fix..sorry.

 - We should uncharge page by mem_cgroup_end_migration() only if
   page is not charged by mem_cgroup_prepare_migration().
 - move mem_cgroup_prepare_migration() after goto:

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

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
@@ -445,7 +445,7 @@ retry:
  }
 }

-void mem_cgroup_prepare_migration(struct page *page)
+int mem_cgroup_prepare_migration(struct page *page)
 {
  struct page_cgroup *pc;
  lock_page_cgroup(page);
@@ -453,7 +453,7 @@ void mem_cgroup_prepare_migration(struct
  if (pc)
   atomic_inc(&pc->ref_cnt);
  unlock_page_cgroup(page);
- return;
+ return (pc != NULL);
 }

 void mem_cgroup_end_migration(struct page *page)
Index: linux-2.6.23-rc8-mm2/include/linux/memcontrol.h
===
--- linux-2.6.23-rc8-mm2.orig/include/linux/memcontrol.h
+++ linux-2.6.23-rc8-mm2/include/linux/memcontrol.h
@@ -47,7 +47,7 @@ extern void mem_cgroup_out_of_memory(str
 extern int mem_cgroup_cache_charge(struct page *page, struct mm_struct *mm,
      gfp_t gfp_mask);

```
 /* For handling page migration in proper way */
-extern void mem_cgroup_prepare_migration(struct page *page);
+extern int mem_cgroup_prepare_migration(struct page *page);
 extern void mem_cgroup_end_migration(struct page *page);
 extern void mem_cgroup_page_migration(struct page *newpage, struct page *page);

@@ -114,9 +114,9 @@ static inline struct mem_cgroup *mm_cgro
 }

 /* For page migration */
-static inline void mem_cgroup_prepare_migration(struct page *page)
+static inline int mem_cgroup_prepare_migration(struct page *page)
 {
- return;
+ return 0;
 }

 static inline void mem_cgroup_end_migration(struct mem_cgroup *cgroup)
Index: linux-2.6.23-rc8-mm2/mm/migrate.c
===================================================================
--- linux-2.6.23-rc8-mm2.orig/mm/migrate.c
+++ linux-2.6.23-rc8-mm2/mm/migrate.c
@@ -620,6 +620,7 @@ static int unmap_and_move(new_page_t get
  int *result = NULL;
  struct page *newpage = get_new_page(page, private, &result);
  int rcu_locked = 0;
+ int charge = 0;

  if (!newpage)
   return -ENOMEM;
@@ -652,8 +653,6 @@ static int unmap_and_move(new_page_t get
   rcu_read_lock();
   rcu_locked = 1;
  }
- mem_cgroup_prepare_migration(page);
-
  /*
   * This is a corner case handling.
   * When a new swap-cache is read into, it is linked to LRU
@@ -663,6 +662,8 @@ static int unmap_and_move(new_page_t get
   */
  if (!page->mapping)
   goto rcu_unlock;
+
+ charge = mem_cgroup_prepare_migration(page);
  /* Establish migration ptes or remove ptes */
  try_to_unmap(page, 1);
```

```
@@ -671,8 +672,9 @@ static int unmap_and_move(new_page_t get

  if (rc) {
   remove_migration_ptes(page, page);
-  mem_cgroup_end_migration(page);
- } else
+  if (charge)
+   mem_cgroup_end_migration(page);
+ } else if (charge)
   mem_cgroup_end_migration(newpage);

 rcu_unlock:
```

_____

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