Subject: [PATCH 3/4] swapcgroup: implement charge/uncharge
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:20:05 GMT
View Forum Message <> Reply to Message

This patch implements charge and uncharge of swapcgroup.

- what will be charged ?
  charge the number of swap entries in bytes.

- when to charge/uncharge ?
  charge at get_swap_entry(), and uncharge at swap_entry_free().

- to what group charge the swap entry ?
  To determine to what mem_cgroup the swap entry should be charged,
  I changed the argument of get_swap_entry() from (void) to
  (struct page *). As a result, get_swap_entry() can determine
  to what mem_cgroup it should charge the swap entry
  by referring to page->page_cgroup->mem_cgroup.

- from what group uncharge the swap entry ?
  I added to swap_info_struct a member 'struct swap_cgroup **',
  array of pointer to which swap_cgroup the swap entry is
  charged.

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

---
 include/linux/memcontrol.h | 21 ++++++++++++++++++++
 include/linux/swap.h | 4 +-
 mm/memcontrol.c | 47 ++
 mm/shmem.c | 2 +-
 mm/swap_state.c | 2 +-
 mm/swapfile.c | 14 +++++++++++-
 6 files changed, 85 insertions(+), 5 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index fdf3967..a7e6621 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -24,6 +24,7 @@ struct mem_cgroup;
 struct page_cgroup;
 struct page;
 struct mm_struct;
+struct swap_info_struct;

 #ifdef CONFIG_CGROUP_MEM_RES_CTLR

```
@@ -172,5 +173,25 @@ static inline long mem_cgroup_calc_reclaim_inactive(struct
mem_cgroup *mem,
 }
 #endif /* CONFIG_CGROUP_MEM_CONT */

+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+extern int swap_cgroup_charge(struct page *page,
+   struct swap_info_struct *si,
+   unsigned long offset);
+extern void swap_cgroup_uncharge(struct swap_info_struct *si,
+    unsigned long offset);
+#else /* CONFIG_CGROUP_SWAP_RES_CTLR */
+static inline int swap_cgroup_charge(struct page *page,
+   struct swap_info_struct *si,
+   unsigned long offset)
+{
+ return 0;
+}
+
+static inline void swap_cgroup_uncharge(struct swap_info_struct *si,
+    unsigned long offset)
+{
+}
+#endif /* CONFIG_CGROUP_SWAP_RES_CTLR */
+
 #endif /* _LINUX_MEMCONTROL_H */

diff --git a/include/linux/swap.h b/include/linux/swap.h
index 67de27b..18887f0 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h
@@ -241,7 +241,7 @@ extern struct page *swapin_readahead(swp_entry_t, gfp_t,
 /* linux/mm/swapfile.c */
 extern long total_swap_pages;
 extern void si_swapinfo(struct sysinfo *);
-extern swp_entry_t get_swap_page(void);
+extern swp_entry_t get_swap_page(struct page *);
 extern swp_entry_t get_swap_page_of_type(int);
 extern int swap_duplicate(swp_entry_t);
 extern int valid_swaphandles(swp_entry_t, unsigned long *);
@@ -342,7 +342,7 @@ static inline int remove_exclusive_swap_page(struct page *p)
  return 0;
 }

-static inline swp_entry_t get_swap_page(void)
+static inline swp_entry_t get_swap_page(struct page *page)
 {
  swp_entry_t entry;
```

```
  entry.val = 0;
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index a837215..84e803d 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1220,3 +1220,50 @@ struct cgroup_subsys mem_cgroup_subsys = {
  .attach = mem_cgroup_move_task,
  .early_init = 0,
 };
+
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+int swap_cgroup_charge(struct page *page,
+   struct swap_info_struct *si,
+   unsigned long offset)
+{
+ int ret;
+ struct page_cgroup *pc;
+ struct mem_cgroup *mem;
+
+ lock_page_cgroup(page);
+ pc = page_get_page_cgroup(page);
+ if (unlikely(!pc))
+  mem = &init_mem_cgroup;
+ else
+  mem = pc->mem_cgroup;
+ unlock_page_cgroup(page);
+
+ css_get(&mem->css);
+ ret = res_counter_charge(&mem->swap_res, PAGE_SIZE);
+ if (!ret)
+  si->memcg[offset] = mem;
+ else
+  css_put(&mem->css);
+
+ return ret;
+}
+
+void swap_cgroup_uncharge(struct swap_info_struct *si,
+    unsigned long offset)
+{
+ struct mem_cgroup *mem = si->memcg[offset];
+
+ /* "mem" would be NULL:
+  * 1. when get_swap_page() failed at charging swap_cgroup,
+  *    and called swap_entry_free().
+  * 2. when this swap entry had been assigned by
+  *    get_swap_page_of_type() (via SWSUSP?).
+  */
```

```
+ if (mem) {
+  res_counter_uncharge(&mem->swap_res, PAGE_SIZE);
+  si->memcg[offset] = NULL;
+  css_put(&mem->css);
+ }
+}
+#endif
+
diff --git a/mm/shmem.c b/mm/shmem.c
index 95b056d..69f8909 100644
--- a/mm/shmem.c
+++ b/mm/shmem.c
@@ -1029,7 +1029,7 @@ static int shmem_writepage(struct page *page, struct
writeback_control *wbc)
   * want to check if there's a redundant swappage to be discarded.
   */
  if (wbc->for_reclaim)
-  swap = get_swap_page();
+  swap = get_swap_page(page);
  else
   swap.val = 0;

diff --git a/mm/swap_state.c b/mm/swap_state.c
index 676e191..a78d617 100644
--- a/mm/swap_state.c
+++ b/mm/swap_state.c
@@ -130,7 +130,7 @@ int add_to_swap(struct page * page, gfp_t gfp_mask)
  BUG_ON(!PageUptodate(page));

  for (;;) {
-  entry = get_swap_page();
+  entry = get_swap_page(page);
   if (!entry.val)
    return 0;

diff --git a/mm/swapfile.c b/mm/swapfile.c
index 232bf20..682b71e 100644
--- a/mm/swapfile.c
+++ b/mm/swapfile.c
@@ -172,7 +172,10 @@ no_page:
  return 0;
 }

-swp_entry_t get_swap_page(void)
+/* get_swap_page() calls this */
+static int swap_entry_free(struct swap_info_struct *, unsigned long);
+
+swp_entry_t get_swap_page(struct page *page)
```

```
 {
  struct swap_info_struct *si;
  pgoff_t offset;
@@ -201,6 +204,14 @@ swp_entry_t get_swap_page(void)
   swap_list.next = next;
   offset = scan_swap_map(si);
   if (offset) {
+   /*
+    * This should be the first use of this swap entry.
+    * So, charge this swap entry here.
+    */
+   if (swap_cgroup_charge(page, si, offset)) {
+    swap_entry_free(si, offset);
+    goto noswap;
+   }
    spin_unlock(&swap_lock);
    return swp_entry(type, offset);
   }
@@ -285,6 +296,7 @@ static int swap_entry_free(struct swap_info_struct *p, unsigned long
offset)
     swap_list.next = p - swap_info;
    nr_swap_pages++;
    p->inuse_pages--;
+   swap_cgroup_uncharge(p, offset);
   }
  }
  return count;
```

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