

---

Subject: [PATCH -mm 3/5] swapcgroup (v3): implement charge and uncharge  
Posted by [Daisuke Nishimura](#) on Fri, 04 Jul 2008 06:20:14 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 in previous patch to swap\_info\_struct a member 'struct swap\_cgroup \*\*', array of pointer to which swap\_cgroup the swap entry is charged.

#### Change log

v2->v3

- Rebased on 2.6.26-rc5-mm3
- make swap\_cgroup\_charge() fail when !pc.
- add check on mem\_cgroup\_subsys.disabled

v1->v2

- Rebased on 2.6.26-rc2-mm1
- Implemented as a add-on to memory cgroup.

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

---

```
include/linux/memcontrol.h | 17 ++++++++
include/linux/swap.h       |  4 +-
mm/memcontrol.c            | 53 +++++++++++++++++++++++++++++++++++++
mm/shmem.c                 |  2 +-
mm/swap_state.c           |  2 +-
mm/swapfile.c             | 13 +++++++-
6 files changed, 86 insertions(+), 5 deletions(-)
```

```
diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index b6ff509..f3c84f7 100644
```

```

--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -170,6 +170,11 @@ static inline long mem_cgroup_calc_reclaim(struct mem_cgroup *mem,
extern struct mem_cgroup **swap_info_clear_memcg(struct swap_info_struct *p);
extern int swap_info_alloc_memcg(struct swap_info_struct *p,
    unsigned long maxpages);
+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
static inline
struct mem_cgroup **swap_info_clear_memcg(struct swap_info_struct *p)
@@ -182,6 +187,18 @@ int swap_info_alloc_memcg(struct swap_info_struct *p, unsigned long
maxpages)
{
    return 0;
}
+
+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

#endif /* _LINUX_MEMCONTROL_H */
diff --git a/include/linux/swap.h b/include/linux/swap.h
index 6e1b03d..f80255b 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h
@@ -298,7 +298,7 @@ extern struct page *swpin_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 *);
@@ -405,7 +405,7 @@ static inline int remove_exclusive_swap_page_ref(struct page *page)

```

```

    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 81bb7fa..d16d0a5 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1243,58 @@ int swap_info_alloc_memcg(struct swap_info_struct *p, unsigned
long maxpages)
out:
    return ret;
}
+
+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;
+
+ if (mem_cgroup_subsys.disabled)
+ return 0;
+
+ lock_page_cgroup(page);
+ pc = page_get_page_cgroup(page);
+ if (unlikely(!pc)) {
+ /* make get_swap_page fail */
+ unlock_page_cgroup(page);
+ return -ENOMEM;
+ } else {
+ mem = pc->mem_cgroup;
+ css_get(&mem->css);
+ }
+ unlock_page_cgroup(page);
+
+ 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];
+
+ if (mem_cgroup_subsys.disabled)
+ return;
+
+ /* "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 bed732c..958e041 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 ca43e64..ad5d85c 100644

```

```

--- a/mm/swap_state.c
+++ b/mm/swap_state.c
@@ -138,7 +138,7 @@ 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 312c573..57798c5 100644
--- a/mm/swapfile.c
+++ b/mm/swapfile.c
@@ -172,7 +172,9 @@ 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 +203,14 @@ 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 +295,7 @@ 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>

---