
Subject: [PATCH] memory cgroup enhancements [1/5] force_empty for memory cgroup

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 16 Oct 2007 10:23:41 GMT

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

This patch adds an interface "memory.force_empty".
Any write to this file will drop all charges in this cgroup if there is no task under.

```
%echo 1 > /...../memory.force_empty
```

will drop all charges of memory cgroup if cgroup's tasks is empty.

This is useful to invoke rmdir() against memory cgroup successfully.

Tested and worked well on x86_64/fake-NUMA system.

Changelog v3 -> v4:

- adjusted to 2.6.23-mm1
- fixed typo
- changes buf[2]="0" to static const

Changelog v2 -> v3:

- changed the name from force_reclaim to force_empty.

Changelog v1 -> v2:

- added a new interface force_reclaim.
- changes spin_lock to spin_lock_irqsave().

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

```
mm/memcontrol.c | 102 ++++++-----
1 file changed, 95 insertions(+), 7 deletions(-)
```

Index: devel-2.6.23-mm1/mm/memcontrol.c

```
=====
--- devel-2.6.23-mm1.orig/mm/memcontrol.c
+++ devel-2.6.23-mm1/mm/memcontrol.c
@@ -480,6 +480,7 @@ void mem_cgroup_uncharge(struct page_cgr
    page = pc->page;
    /*
     * get page->cgroup and clear it under lock.
+ * force_empty can drop page->cgroup without checking refcnt.
     */
    if (clear_page_cgroup(page, pc) == pc) {
        mem = pc->mem_cgroup;
@@ -489,13 +490,6 @@ void mem_cgroup_uncharge(struct page_cgr
    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();
- }
- }
- }
@@ -543,6 +537,70 @@ retry:
    return;
}

+/*
+ * This routine traverse page_cgroup in given list and drop them all.
+ * This routine ignores page_cgroup->ref_cnt.
+ * *And* this routine doesn't reclaim page itself, just removes page_cgroup.
+ */
+static void
+mem_cgroup_force_empty_list(struct mem_cgroup *mem, struct list_head *list)
+{
+ struct page_cgroup *pc;
+ struct page *page;
+ int count = SWAP_CLUSTER_MAX;
+ unsigned long flags;
+
+ spin_lock_irqsave(&mem->lru_lock, flags);
+
+ while (!list_empty(list)) {
+ pc = list_entry(list->prev, struct page_cgroup, lru);
+ page = pc->page;
+ /* Avoid race with charge */
+ atomic_set(&pc->ref_cnt, 0);
+ if (clear_page_cgroup(page, pc) == pc) {
+ css_put(&mem->css);
+ res_counter_uncharge(&mem->res, PAGE_SIZE);
+ list_del_init(&pc->lru);
+ kfree(pc);
+ } else
+ count = 1; /* being uncharged ? ...do relax */
+
+ if (--count == 0) {
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+ cond_resched();
+ spin_lock_irqsave(&mem->lru_lock, flags);

```

```

+ count = SWAP_CLUSTER_MAX;
+ }
+ }
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+}
+
+/*
+ * make mem_cgroup's charge to be 0 if there is no binded task.
+ * This enables deleting this mem_cgroup.
+ */
+
+int mem_cgroup_force_empty(struct mem_cgroup *mem)
+{
+ int ret = -EBUSY;
+ css_get(&mem->css);
+ while (!list_empty(&mem->active_list) ||
+        !list_empty(&mem->inactive_list)) {
+ if (atomic_read(&mem->css.cgroup->count) > 0)
+ goto out;
+ /* drop all page_cgroup in active_list */
+ mem_cgroup_force_empty_list(mem, &mem->active_list);
+ /* drop all page_cgroup in inactive_list */
+ mem_cgroup_force_empty_list(mem, &mem->inactive_list);
+ }
+ ret = 0;
+out:
+ css_put(&mem->css);
+ return ret;
+}
+
+
+
+int mem_cgroup_write_strategy(char *buf, unsigned long long *tmp)
+{
+ *tmp = memparse(buf, &buf);
@@ -628,6 +686,31 @@ static ssize_t mem_control_type_read(str
+ ppos, buf, s - buf);
+ }
+
+
+static ssize_t mem_force_empty_write(struct cgroup *cont,
+ struct cftype *cft, struct file *file,
+ const char __user *userbuf,
+ size_t nbytes, loff_t *ppos)
+{
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+ int ret;
+ ret = mem_cgroup_force_empty(mem);

```

```

+ if (!ret)
+   ret = nbytes;
+ return ret;
+}
+
+static ssize_t mem_force_empty_read(struct cgroup *cont,
+   struct cftype *cft,
+   struct file *file, char __user *userbuf,
+   size_t nbytes, loff_t *ppos)
+{
+   static const char buf[2] = "0";
+   return simple_read_from_buffer((void __user *)userbuf, nbytes,
+   +   ppos, buf, strlen(buf));
+}
+
+
+static struct cftype mem_cgroup_files[] = {
+   {
+       .name = "usage_in_bytes",
@@ -650,6 +733,11 @@ static struct cftype mem_cgroup_files[]
+       .write = mem_control_type_write,
+       .read = mem_control_type_read,
+   },
+   {
+       .name = "force_empty",
+       .write = mem_force_empty_write,
+       .read = mem_force_empty_read,
+   },
+};
+
+static struct mem_cgroup init_mem_cgroup;

```

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