
Subject: [PATCH] memory cgroup enhancements take 4 [3/8] remember "a page is charged as page cache"

Posted by [KAMEZAWA Hiroyuki](#) on Wed, 31 Oct 2007 10:28:48 GMT

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

Add a flag to page_cgroup to remember "this page is charged as cache."

cache here includes page caches and swap cache.

This is useful for implementing precise accounting in memory cgroup.

TODO:

- distinguish page-cache and swap-cache

Changelog v2 -> v3

- added enum for mem_cgroup_charge_type_common(...charge_type)
- renamed #define PCGF_XXX_XXX to PAGE_CGROUP_FLAG_XXX

Changelog v1 -> v2

- moved #define to out-side of struct definition

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

Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

mm/memcontrol.c | 24 ++++++

1 file changed, 21 insertions(+), 3 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
@@ -83,7 +83,9 @@ struct page_cgroup {
    struct mem_cgroup *mem_cgroup;
    atomic_t ref_cnt; /* Helpful when pages move b/w */
    /* mapped and cached states */
+ int flags;
};
+#define PAGE_CGROUP_FLAG_CACHE (0x1) /* charged as cache */

enum {
    MEM_CGROUP_TYPE_UNSPEC = 0,
@@ -93,6 +95,11 @@ enum {
    MEM_CGROUP_TYPE_MAX,
};

+enum charge_type {
+ MEM_CGROUP_CHARGE_TYPE_CACHE = 0,
+ MEM_CGROUP_CHARGE_TYPE_MAPPED,
+};
+
```

```

static struct mem_cgroup init_mem_cgroup;

static inline
@@ -315,8 +322,8 @@ unsigned long mem_cgroup_isolate_pages(u
 * 0 if the charge was successful
 * < 0 if the cgroup is over its limit
 */
-int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
- gfp_t gfp_mask)
+static int mem_cgroup_charge_common(struct page *page, struct mm_struct *mm,
+ gfp_t gfp_mask, enum charge_type ctype)
{
    struct mem_cgroup *mem;
    struct page_cgroup *pc;
@@ -418,6 +425,9 @@ noreclaim:
    atomic_set(&pc->ref_cnt, 1);
    pc->mem_cgroup = mem;
    pc->page = page;
+ pc->flags = 0;
+ if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
+ pc->flags |= PAGE_CGROUP_FLAG_CACHE;
    if (page_cgroup_assign_new_page_cgroup(page, pc)) {
/*
 * an another charge is added to this page already.
@@ -442,6 +452,13 @@ err:
    return -ENOMEM;
}

+int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
+ gfp_t gfp_mask)
+{
+ return mem_cgroup_charge_common(page, mm, gfp_mask,
+ MEM_CGROUP_CHARGE_TYPE_MAPPED);
+}
+
/*
 * See if the cached pages should be charged at all?
 */
@@ -454,7 +471,8 @@ int mem_cgroup_cache_charge(struct page

    mem = rcu_dereference(mm->mem_cgroup);
    if (mem->control_type == MEM_CGROUP_TYPE_ALL)
- return mem_cgroup_charge(page, mm, gfp_mask);
+ return mem_cgroup_charge_common(page, mm, gfp_mask,
+ MEM_CGROUP_CHARGE_TYPE_CACHE);
    else
        return 0;
}

```

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