

---

Subject: [PATCH 05/23] memcg: Reclaim when more than one page needed.  
Posted by [Glauber Costa](#) on Fri, 20 Apr 2012 21:49:02 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Suleiman Souhlal <ssouhlal@FreeBSD.org>

mem\_cgroup\_do\_charge() was written before slab accounting, and expects three cases: being called for 1 page, being called for a stock of 32 pages, or being called for a hugepage. If we call for 2 pages (and several slabs used in process creation are such, at least with the debug options I had), it assumed it's being called for stock and just retried without reclaiming.

Fix that by passing down a minsize argument in addition to the csize.

And what to do about that (csiz == PAGE\_SIZE && ret) retry? If it's needed at all (and presumably is since it's there, perhaps to handle races), then it should be extended to more than PAGE\_SIZE, yet how far? And should there be a retry count limit, of what? For now retry up to COSTLY\_ORDER (as page\_alloc.c does), stay safe with a cond\_resched(), and make sure not to do it if \_\_GFP\_NORETRY.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>

---

mm/memcontrol.c | 18 ++++++++-----  
1 files changed, 11 insertions(+), 7 deletions(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c  
index 4b94b2d..cbffc4c 100644

--- a/mm/memcontrol.c

+++ b/mm/memcontrol.c

@@ -2187,7 +2187,8 @@ enum {  
};

static int mem\_cgroup\_do\_charge(struct mem\_cgroup \*memcg, gfp\_t gfp\_mask,

- unsigned int nr\_pages, bool oom\_check)

+ unsigned int nr\_pages, unsigned int min\_pages,

+ bool oom\_check)

{  
unsigned long csize = nr\_pages \* PAGE\_SIZE;

struct mem\_cgroup \*mem\_over\_limit;

@@ -2210,18 +2211,18 @@ static int mem\_cgroup\_do\_charge(struct mem\_cgroup \*memcg,  
gfp\_t gfp\_mask,

} else

mem\_over\_limit = mem\_cgroup\_from\_res\_counter(fail\_res, res);

/\*

- \* nr\_pages can be either a huge page (HPAGE\_PMD\_NR), a batch

- \* of regular pages (CHARGE\_BATCH), or a single regular page (1).

- \*

```

    * Never reclaim on behalf of optional batching, retry with a
    * single page instead.
    */
- if (nr_pages == CHARGE_BATCH)
+ if (nr_pages > min_pages)
    return CHARGE_RETRY;

    if (!(gfp_mask & __GFP_WAIT))
        return CHARGE_WOULDBLOCK;

+ if (gfp_mask & __GFP_NORETRY)
+ return CHARGE_NOMEM;
+
    ret = mem_cgroup_reclaim(mem_over_limit, gfp_mask, flags);
    if (mem_cgroup_margin(mem_over_limit) >= nr_pages)
        return CHARGE_RETRY;
@@ -2234,8 +2235,10 @@ static int mem_cgroup_do_charge(struct mem_cgroup *memcg, gfp_t
gfp_mask,
    * unlikely to succeed so close to the limit, and we fall back
    * to regular pages anyway in case of failure.
    */
- if (nr_pages == 1 && ret)
+ if (nr_pages <= (PAGE_SIZE << PAGE_ALLOC_COSTLY_ORDER) && ret) {
+ cond_resched();
    return CHARGE_RETRY;
+ }

    /*
    * At task move, charge accounts can be doubly counted. So, it's
    @@ -2369,7 +2372,8 @@ again:
        nr_oom_retries = MEM_CGROUP_RECLAIM_RETRIES;
    }

- ret = mem_cgroup_do_charge(memcg, gfp_mask, batch, oom_check);
+ ret = mem_cgroup_do_charge(memcg, gfp_mask, batch, nr_pages,
+ oom_check);
    switch (ret) {
    case CHARGE_OK:
        break;
--
1.7.7.6

```

---