

---

Subject: [PATCH 02/11] memcg: Reclaim when more than one page needed.  
Posted by [Glauber Costa](#) on Mon, 25 Jun 2012 14:15:19 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 or 3 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.

[v4: fixed nr pages calculation pointed out by Christoph Lameter ]

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

Signed-off-by: Glauber Costa <glommer@parallels.com>

Reviewed-by: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

---

mm/memcontrol.c | 23 ++++++-----  
1 file changed, 16 insertions(+), 7 deletions(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c

index 9304db2..8e601e8 100644

--- a/mm/memcontrol.c

+++ b/mm/memcontrol.c

@ @ -2158,8 +2158,16 @ @ enum {

CHARGE\_OOM\_DIE, /\* the current is killed because of OOM \*/  
};

+/\*

+ \* We need a number that is small enough to be likely to have been

+ \* reclaimed even under pressure, but not too big to trigger unnecessary

+ \* retries

+ \*/

+#define NR\_PAGES\_TO\_RETRY 2

+

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;
@@ -2182,18 +2190,18 @@ 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;
@@ -2206,7 +2214,7 @@ 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 <= NR_PAGES_TO_RETRY && ret)
    return CHARGE_RETRY;

/*
@@ -2341,7 +2349,8 @@ 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;

```

--

1.7.10.2

---