Subject: [PATCH 03/10] memcg: Reclaim when more than one page needed.
Posted by Suleiman Souhlal on Mon, 27 Feb 2012 22:58:46 GMT
View Forum Message <> Reply to Message

From: Hugh Dickins <hughd@google.com>

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 pass minsize to consume_stock() also, so that it can draw on stock
for higher order slabs, instead of accumulating an increasing surplus
of stock, as its "nr_pages == 1" tests previously caused.

And what to do about that (csize == 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: Hugh Dickins <hughd@google.com>
Signed-off-by: Suleiman Souhlal <suleiman@google.com>
---
 mm/memcontrol.c | 35 +++++++++++++++++++++++++----------
 1 files changed, 19 insertions(+), 16 deletions(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 6f44fcb..c82ca1c 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1928,19 +1928,19 @@ static DEFINE_PER_CPU(struct memcg_stock_pcp,
memcg_stock);
 static DEFINE_MUTEX(percpu_charge_mutex);

 /*
- * Try to consume stocked charge on this cpu. If success, one page is consumed
- * from local stock and true is returned. If the stock is 0 or charges from a
- * cgroup which is not current target, returns false. This stock will be
- * refilled.
+ * Try to consume stocked charge on this cpu. If success, nr_pages pages are
+ * consumed from local stock and true is returned. If the stock is 0 or
+ * charges from a cgroup which is not current target, returns false.
+ * This stock will be refilled.
  */

```
-static bool consume_stock(struct mem_cgroup *memcg)
+static bool consume_stock(struct mem_cgroup *memcg, int nr_pages)
 {
  struct memcg_stock_pcp *stock;
  bool ret = true;

  stock = &get_cpu_var(memcg_stock);
- if (memcg == stock->cached && stock->nr_pages)
-  stock->nr_pages--;
+ if (memcg == stock->cached && stock->nr_pages >= nr_pages)
+  stock->nr_pages -= nr_pages;
  else /* need to call res_counter_charge */
   ret = false;
  put_cpu_var(memcg_stock);
@@ -2131,7 +2131,7 @@ 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;
@@ -2154,18 +2154,18 @@ 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;
@@ -2178,8 +2178,10 @@ 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
@@ -2253,7 +2255,7 @@ again:
   VM_BUG_ON(css_is_removed(&memcg->css));
   if (mem_cgroup_is_root(memcg))
    goto done;
-  if (nr_pages == 1 && consume_stock(memcg))
+  if (consume_stock(memcg, nr_pages))
    goto done;
   css_get(&memcg->css);
  } else {
@@ -2278,7 +2280,7 @@ again:
    rcu_read_unlock();
    goto done;
   }
-  if (nr_pages == 1 && consume_stock(memcg)) {
+  if (consume_stock(memcg, nr_pages)) {
    /*
     * It seems dagerous to access memcg without css_get().
     * But considering how consume_stok works, it's not
@@ -2313,7 +2315,8 @@ 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;
--
1.7.7.3
```