Subject: [PATCH v5 01/14] memcg: Make it possible to use the stock for more than one page.
Posted by Glauber Costa on Tue, 16 Oct 2012 10:16:38 GMT
View Forum Message <> Reply to Message

From: Suleiman Souhlal <ssouhlal@FreeBSD.org>

We currently have a percpu stock cache scheme that charges one page at a time from memcg->res, the user counter. When the kernel memory controller comes into play, we'll need to charge more than that.

This is because kernel memory allocations will also draw from the user counter, and can be bigger than a single page, as it is the case with the stack (usually 2 pages) or some higher order slabs.

[glommer@parallels.com: added a changelog]

Signed-off-by: Suleiman Souhlal <suleiman@google.com>
Signed-off-by: Glauber Costa <glommer@parallels.com>
Acked-by: David Rientjes <rientjes@google.com>
Acked-by: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
Acked-by: Michal Hocko <mhocko@suse.cz>
Acked-by: Johannes Weiner <hannes@cmpxchg.org>
CC: Tejun Heo <tj@kernel.org>
---
 mm/memcontrol.c | 28 ++++++++++++++++++++----------
 1 file changed, 18 insertions(+), 10 deletions(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 7acf43b..47cb019 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -2028,20 +2028,28 @@ struct memcg_stock_pcp {
 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.
+/**
+ * consume_stock: Try to consume stocked charge on this cpu.
+ * @memcg: memcg to consume from.
+ * @nr_pages: how many pages to charge.
+ *
+ * The charges will only happen if @memcg matches the current cpu's memcg
+ * stock, and at least @nr_pages are available in that stock. Failure to

```
+ * service an allocation will refill the stock.
+ *
+ * returns true if succesfull, false otherwise.
  */
-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;

+ if (nr_pages > CHARGE_BATCH)
+  return false;
+
  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);
@@ -2340,7 +2348,7 @@ 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 {
@@ -2365,7 +2373,7 @@ 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
--
1.7.11.7
```