Subject: Re: [PATCH v2 04/13] memcg: Make it possible to use the stock for more than one page.
Posted by Glauber Costa on Sun, 11 Mar 2012 10:49:52 GMT
View Forum Message <> Reply to Message

On 03/10/2012 12:39 AM, Suleiman Souhlal wrote:
> Signed-off-by: Suleiman Souhlal<suleiman@google.com>
> ---
> mm/memcontrol.c | 18 +++++++++---------
> 1 files changed, 9 insertions(+), 9 deletions(-)
>
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index 6fbb438..f605100 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -1965,19 +1965,19 @@ 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);
> @@ -2290,7 +2290,7 @@ 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 {
> @@ -2315,7 +2315,7 @@ 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

This patch itself is fine in what it wants to achieve.
But it made me think:

We'll jump into the stock code which makes user allocation faster.
but we're not getting the benefit of it when we're accounting kmem.
since we're allocating to both res_counters, we're actually defeating it
altogether, since we now have to go to the global poll *everytime* (for
memcg->kmem).

It would make a whole lot more sense to have the stock code moved to the
res_counter. We're now starting to have more users of that anyway, so
a common implementation makes sense.
