Subject: Re: [PATCH v2 10/11] memcg: allow a memcg with kmem charges to be destructed.
Posted by Michal Hocko on Tue, 21 Aug 2012 08:22:59 GMT
View Forum Message <> Reply to Message

On Thu 09-08-12 17:01:18, Glauber Costa wrote:
> Because the ultimate goal of the kmem tracking in memcg is to track slab
> pages as well, we can't guarantee that we'll always be able to point a
> page to a particular process, and migrate the charges along with it -
> since in the common case, a page will contain data belonging to multiple
> processes.
>
> Because of that, when we destroy a memcg, we only make sure the
> destruction will succeed by discounting the kmem charges from the user
> charges when we try to empty the cgroup.

This changes the semantic of memory.force_empty file because the usage should be 0 on success but it will show kmem usage in fact now. I guess it is inevitable with u+k accounting so you should be explicit about that and also update the documentation. If some tests (I am not 100% sure but I guess LTP) rely on that then they could be fixed by checking the kmem limit as well.

> Signed-off-by: Glauber Costa <glommer@parallels.com>
> Acked-by: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> CC: Christoph Lameter <cl@linux.com>
> CC: Pekka Enberg <penberg@cs.helsinki.fi>
> CC: Michal Hocko <mhocko@suse.cz>
> CC: Johannes Weiner <hannes@cmpxchg.org>
> CC: Suleiman Souhlal <suleiman@google.com>
> ---
> mm/memcontrol.c | 17 ++++++++++++++++-
> 1 file changed, 16 insertions(+), 1 deletion(-)
>
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index 3d30b79..7c1ea49 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -649,6 +649,11 @@ static void disarm_kmem_keys(struct mem_cgroup *memcg)
> {
>   if (test_bit(KMEM_ACCOUNTED_THIS, &memcg->kmem_accounted))
>   static_key_slow_dec(&memcg_kmem_enabled_key);
> + /*
> + * This check can't live in kmem destruction function,
> + * since the charges will outlive the cgroup
> + */
> + WARN_ON(res_counter_read_u64(&memcg->kmem, RES_USAGE) != 0);
> }

> #else
> static void disarm_kmem_keys(struct mem_cgroup *memcg)
> @@ -4005,6 +4010,7 @@ static int mem_cgroup_force_empty(struct mem_cgroup *memcg,
bool free_all)
>   int node, zid, shrink;
>   int nr_retries = MEM_CGROUP_RECLAIM_RETRIES;
>   struct cgroup *cgrp = memcg->css.cgroup;
> + u64 usage;
>
>   css_get(&memcg->css);
>
> @@ -4038,8 +4044,17 @@ move_account:
>   mem_cgroup_end_move(memcg);
>   memcg_oom_recover(memcg);
>   cond_resched();
> + /*
> +   * Kernel memory may not necessarily be trackable to a specific
> +   * process. So they are not migrated, and therefore we can't
> +   * expect their value to drop to 0 here.
> +   *
> +   * having res filled up with kmem only is enough
> +   */
> + usage = res_counter_read_u64(&memcg->res, RES_USAGE) -
> +   res_counter_read_u64(&memcg->kmem, RES_USAGE);
>   /* "ret" should also be checked to ensure all lists are empty. */
> - } while (res_counter_read_u64(&memcg->res, RES_USAGE) > 0 || ret);
> + } while (usage > 0 || ret);
> out:
>   css_put(&memcg->css);
>   return ret;
> --
> 1.7.11.2
>
> --
> To unsubscribe from this list: send the line "unsubscribe cgroups" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

--
Michal Hocko
SUSE Labs