Subject: Re: [PATCH v3 06/13] memcg: kmem controller infrastructure
Posted by Michal Hocko on Mon, 01 Oct 2012 11:51:57 GMT
View Forum Message <> Reply to Message

On Mon 01-10-12 14:09:09, Glauber Costa wrote:
> On 10/01/2012 01:48 PM, Michal Hocko wrote:
> > On Fri 28-09-12 15:34:19, Glauber Costa wrote:
> >> On 09/27/2012 05:44 PM, Michal Hocko wrote:
> >>>>> the reference count aquired by mem_cgroup_get will still prevent the
> >>>>> memcg from going away, no?
> >>> Yes but you are outside of the rcu now and we usually do css_get before
> >>> we rcu_unlock. mem_cgroup_get just makes sure the group doesn't get
> >>> deallocated but it could be gone before you call it. Or I am just
> >>> confused - these 2 levels of ref counting is really not nice.
> >>>
> >>> Anyway, I have just noticed that __mem_cgroup_try_charge does
> >>> VM_BUG_ON(css_is_removed(&memcg->css)) on a given memcg so you should
> >>> keep css ref count up as well.
> >>>
> >>
> >> IIRC, css_get will prevent the cgroup directory from being removed.
> >> Because some allocations are expected to outlive the cgroup, we
> >> specifically don't want that.
> >
> > Yes, but how do you guarantee that the above VM_BUG_ON doesn't trigger?
> > Task could have been moved to another group between mem_cgroup_from_task
> > and mem_cgroup_get, no?
> >
>
> Ok, after reading this again (and again), you seem to be right. It
> concerns me, however, that simply getting the css would lead us to a
> double get/put pair, since try_charge will have to do it anyway.

That happens only for !*ptr case and you provide a memcg here, don't
you.

> I considered just letting try_charge selecting the memcg, but that is
> not really what we want, since if that memcg will fail kmem allocations,
> we simply won't issue try charge, but return early.
>
> Any immediate suggestions on how to handle this ?

I would do the same thing __mem_cgroup_try_charge does.

```
retry:
 rcu_read_lock();
 p = rcu_dereference(mm->owner);
 if (!css_tryget(&memcg->css)) {
  rcu_read_unlock();
  goto retry;
 }
 rcu_read_unlock();
```

--
Michal Hocko
SUSE Labs