Subject: Re: [PATCH v3 16/28] memcg: kmem controller charge/uncharge infrastructure
Posted by Glauber Costa on Wed, 30 May 2012 13:06:22 GMT
View Forum Message <> Reply to Message

On 05/30/2012 05:04 PM, Frederic Weisbecker wrote:
> Do you think it's possible that this memcg can be destroyed (like ss->destroy())
> concurrently?
>
> Probably not because there is a synchronize_rcu() in cgroup_diput() so as long
> as we are in rcu_read_lock() we are fine.
>
> OTOH current->mm->owner can exit() right after we fetched its memcg and thus the css_set
> can be freed concurrently? And then the cgroup itself after we call rcu_read_unlock()
> due to cgroup_diput().
> And yet we are doing the mem_cgroup_get() below unconditionally assuming it's
> always fine to get a reference to it.
>
> May be I'm missing something?
When a cache is created, we grab a reference to the memcg. So after the
cache is created, no.

When destroy is called, we flush the create queue, so if the cache is
not created yet, it will just disappear.

I think the only problem that might happen is in the following scenario:

* cache gets created, but ref count is not yet taken
* memcg disappears
* we try to inc refcount for a non-existent memcg, and crash.

This would be trivially solvable by grabing the reference earlier.
But even then, I need to audit this further to make sure it is really an
issue.