Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 15:49:51 GMT
View Forum Message <> Reply to Message

On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> Paul (??) Menage wrote:
> > On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> >> >
> >> > > mutex_lock(&container_mutex);
> >> > > set_bit(CONT_RELEASABLE, &cont->flags);
> >> > >- if (atomic_dec_and_test(&css->refcnt)) {
> >> > >- check_for_release(cont);
> >> > >- }
> >> > >+ check_for_release(cont);
> >> > > mutex_unlock(&container_mutex);
> >> > >
> > >
> > > I think that this isn't safe as it stands, without a synchronize_rcu()
> > > in container_diput() prior to the kfree(). Also, it will break if
> > > anyone tries to use a release agent on a hierarchy that has your
> > > memory controller bound to it.
> > >
> >
> >
> Isn't the code functionally the same as before? We still do atomic_test_and_dec()
> as before. We still set_bit() CONT_RELEASABLE, we take the container_mutex
> and check_for_release(). I am not sure I understand what changed?

Because as soon as you do the atomic_dec_and_test() on css->refcnt and the refcnt hits zero, then theoretically someone other thread (that already holds container_mutex) could check that the refcount is zero and free the container structure.

Adding a synchronize_rcu in container_diput() guarantees that the container structure won't be freed while someone may still be accessing it.

>
> Could you please elaborate as to why using a release agent is broken
> when the memory controller is attached to it?

Because then it will try to take container_mutex in css_put() if it drops the last reference to a container, which is the thing that you said you had to avoid since you called css_put() in contexts that couldn't sleep.

Paul

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers