Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 02:35:01 GMT
View Forum Message <> Reply to Message

On 7/16/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> - if (notify_on_release(cont)) {
> + if (atomic_dec_and_test(&css->refcnt) && notify_on_release(cont)) {

This seems like a good idea, as long as atomic_dec_and_test() isn't noticeably more expensive than atomic_dec(). I assume it shouldn't need to be, since the bus locking operations are presumably the same in each case.

> 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);
>
> That way we set the CONT_RELEASABLE bit only when the ref count drops
> to zero.
>

That's probably a good idea, in conjunction with another part of my patch for this that frees container objects under RCU - as soon as you do the atomic_dec_and_test(), then in theory some other thread could delete the container (since we're no longer going to be taking container_mutex in this function). But as long as the container object remains valid until synchronize_rcu() completes, then we can safely set the CONT_RELEASABLE bit on it.

>
> Yes, that is correct, the advantage is that with can_destroy() we
> don't need to go through release synchronization each time we do
> a css_put().

I think the amount of release synchronization *needed* is going to be the same whether you have the refcounting done in the subsystem or in the framework. But I agree that right now we're doing one more atomic op than we strictly need to, and can remove it.

Paul

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: Containers: css_put() dilemma
Posted by Balbir Singh on Tue, 17 Jul 2007 07:00:31 GMT
View Forum Message <> Reply to Message

> On 7/16/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> >
> >- if (notify_on_release(cont)) {
> >+ if (atomic_dec_and_test(&css->refcnt) && notify_on_release(cont)) {
>
> This seems like a good idea, as long as atomic_dec_and_test() isn't
> noticeably more expensive than atomic_dec(). I assume it shouldn't
> need to be, since the bus locking operations are presumably the same
> in each case.
>
> > 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);
> >
> >That way we set the CONT_RELEASABLE bit only when the ref count drops
> >to zero.
> >
>
> That's probably a good idea, in conjunction with another part of my
> patch for this that frees container objects under RCU - as soon as you
> do the atomic_dec_and_test(), then in theory some other thread could
> delete the container (since we're no longer going to be taking
> container_mutex in this function). But as long as the container object
> remains valid until synchronize_rcu() completes, then we can safely
> set the CONT_RELEASABLE bit on it.
>
> >
> >Yes, that is correct, the advantage is that with can_destroy() we
> >don't need to go through release synchronization each time we do
> >a css_put().
>
> I think the amount of release synchronization *needed* is going to be
> the same whether you have the refcounting done in the subsystem or in
> the framework. But I agree that right now we're doing one more atomic
> op than we strictly need to, and can remove it.
>

> Paul

Hi, Paul/Andrew

Would you accept this fix, while we wait for the complete solution.
It worked for me quite well.

Description

Stop checking if the container can be released every time we do css_put().
A better solution that avoids container_mutex has been suggested by
Paul, but meanwhile, to get containers working correctly, this fix
would be very useful.

Signed-off-by: <balbir@linux.vnet.ibm.com>
---

 kernel/container.c | 8 ++------
 1 file changed, 2 insertions(+), 6 deletions(-)

diff -puN kernel/container.c~container-css-put-on-refcount-zero kernel/container.c
--- linux-2.6.22-rc6/kernel/container.c~container-css-put-on-refcount-zero 2007-07-17
12:18:52.000000000 +0530
+++ linux-2.6.22-rc6-balbir/kernel/container.c 2007-07-17 12:23:29.000000000 +0530
@@ -2515,15 +2515,11 @@ static void check_for_release(struct con
 void css_put(struct container_subsys_state *css)
 {
  struct container *cont = css->container;
- if (notify_on_release(cont)) {
+ if (atomic_dec_and_test(&css->refcnt) && notify_on_release(cont)) {
   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);
- } else {
- atomic_dec(&css->refcnt);
  }
 }

_

--
 Warm Regards,
 Balbir Singh

Linux Technology Center
IBM, ISTL

_____

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