

---

Subject: Re: Containers: css\_put() dilemma

Posted by [Balbir Singh](#) on Tue, 17 Jul 2007 02:21:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Paul (??) Menage wrote:

> On 7/16/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>> Hi, Paul,

>>

>> I've run into a strange problem with css\_put(). After the changes for  
>> notify\_on\_release(), the css\_put() routine can now block and it blocks on  
>> the container\_mutex. This implies that css\_put() cannot be called if

>>

>> 1. We cannot block

>> 2. We already hold the container\_mutex

>>

>> The problem I have is that of preventing the destruction of my container  
>> (when the user does rmdir). If the user migrates away all tasks and does  
>> an rmdir, the only way to prevent the container from going away is  
>> through

>> css\_get() references. In my case, some pages have been allocated from the  
>> container and hence I do not want it to go away, until all the pages  
>> charged to it are freed. When I use css\_get/put() to prevent destruction  
>> I am blocked by the limitations of css\_put() listed above.

>>

>> Do you have any recommendations for a cleaner solution? I suspect we'll  
>> need can\_destroy() callbacks (similar to can\_attach()).

>

> I think moving the release\_list synchronization inside a separate  
> spinlock, and thus not requiring container\_mutex to be held for  
> check\_for\_release(), is the simplest solution. I'll do that. I'm  
> hoping to get a new set of patches to Andrew today or tomorrow.  
>

That sounds good to me. But I worry about having to do release synchronization on every css\_put(). The current patch I have, but does not work 100% does the following (WARNING: white spaces ahead, do not use the patch directly)

```
-   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);
```

That way we set the CONT\_RELEASABLE bit only when the ref count drops to zero.

> Adding a can\_destroy() callback is possible, but since I envisage that  
> most subsystems that would want to implement it would basically be  
> doing reference counting anyway, it seems worth having a generic  
> reference counting mechanism in the framework. In particular, since  
> once the container does become releasable due to all the  
> subsystem-specific refcounts being released, we want to be able to  
> invoke the release agent, we'll end up with the same synchronization  
> problems that we have now if we just pushed everything into a  
> can\_destroy() method. (Unless the framework polled all can\_destroy()  
> methods for potentially-removable containers, which seems a bit  
> nasty).  
>  
> We can add can\_destroy() if we encounter a situation that can't be  
> handled by generic reference counting.  
>

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(). May be the patch above will fix the problem along with your release locking proposal.

> Paul  
> \_\_\_\_\_  
> Containers mailing list  
> Containers@lists.linux-foundation.org  
> <https://lists.linux-foundation.org/mailman/listinfo/containers>

--

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>

---