Subject: Containers: css_put() dilemma
Posted by Balbir Singh on Mon, 16 Jul 2007 18:50:40 GMT
View Forum Message <> Reply to Message

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()).

--
 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

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

Subject: Re: Containers: css_put() dilemma
Posted by Dave Hansen on Tue, 17 Jul 2007 16:02:11 GMT
View Forum Message <> Reply to Message

> 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.

Then that other task had a reference and itself should have bumped the
count, and the other user would never have seen it hit zero.

Even if there are still pages attached to the container, why not just
have those take a reference, and don't bother actually freeing the
container until the last true reference is dropped?

Does it matter if the destruction callbacks don't happen until well
after an attempt to destroy the container is made?

-- Dave

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

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 16:15:58 GMT
View Forum Message <> Reply to Message

On 7/17/07, Dave Hansen <haveblue@us.ibm.com> wrote:
> On Tue, 2007-07-17 at 08:49 -0700, Paul ($BJuN\ (B) Menage wrote:
> > 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.
>
> Then that other task had a reference and itself should have bumped the
> count, and the other user would never have seen it hit zero.

Nope. The container could have been empty (of tasks) and hence had a zero count.

The liveness model used by containers is that when the refcount hits
zero, the container isn't immediately destroyed (since it can contain
useful historical usage data, etc) but simply becomes eligible for
destruction by userspace via an rmdir().

>
> Even if there are still pages attached to the container, why not just
> have those take a reference, and don't bother actually freeing the
> container until the last true reference is dropped?

Yes, we could potentially just use the main count variable rather than
having separate per-subsystem extra refcounts. The main reasons to do

it this way are:

- the root subsystem state for a subsystem can shift between different "struct container" objects if it was previously inactive and gets mounted as part of a hierarchy (or similarly, gets unmounted and goes inactive). Possibly we could get around this by simply saying not doing refcounting on the subsys states attached to root containers since they can never be freed anyway.

- At some point I'd like to be able to support shifting subsystems between active hierarchies, at least in limited cases such as where the hierarchies are isomorphic, or binding/unbinding subsystems to/from active hierarchies. At that point we definitely need to be able to split out the different subsystem state refcounts from one another in the same hierarchy.

- we'd still have the issue that Balbir wants to be able to drop a reference in a non-sleeping context, and we want to avoid doing excessive synchronization in the normal case when the css_put() doesn't put the reference count to zero.

>
> Does it matter if the destruction callbacks don't happen until well
> after an attempt to destroy the container is made?

Well that's sort of the point of putting a synchronize_rcu() in container_diput() - it ensures that the actual destruction of the object doesn't occur until sufficiently after the destruction attempt is initiated that no one is still using the reference.

The alternative would be something that polls to spot whether refcounts have reached zero and if so runs the userspace helper. That doesn't seem particularly palatable when you have large numbers of containers, if we can avoid it easily.

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Paul Jackson on Tue, 17 Jul 2007 17:23:27 GMT

View Forum Message <> Reply to Message

> 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.

Not just theory ... I've debugged crashes from live customer systems
that were basically this race.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: Containers: css_put() dilemma
Posted by Balbir Singh on Tue, 17 Jul 2007 17:40:59 GMT

View Forum Message <> Reply to Message

Paul (??) Menage wrote:
> 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.
>

Hi, Paul,

That sounds correct. I wonder now if the solution should be some form
of delegation for deletion of unreferenced containers (HINT: work queue
or kernel threads).

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

Do we take rcu_read_lock() in css_put() path or use call_rcu() to
free the container?

>>
>> 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

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 17:44:08 GMT
View Forum Message <> Reply to Message

On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> That sounds correct. I wonder now if the solution should be some form
> of delegation for deletion of unreferenced containers (HINT: work queue
> or kernel threads).

What a great idea. In fact, that's exactly what the release agent
patch already does.

>
> > Adding a synchronize_rcu in container_diput() guarantees that the
> > container structure won't be freed while someone may still be
> > accessing it.
> >
>
> Do we take rcu_read_lock() in css_put() path or use call_rcu() to
> free the container?

Good point, we ought to add rcu_read_lock() (even though it doesn't
actually do anything on architectures other than alpha, right?)

Using call_rcu to do the container kfree rather than synchronize_rcu()
would be a possible future optimization, yes.

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Paul Jackson on Tue, 17 Jul 2007 17:53:41 GMT
View Forum Message <> Reply to Message

> That sounds correct. I wonder now if the solution should be some form
> of delegation for deletion of unreferenced containers (HINT: work queue
> or kernel threads).

At least for cpusets (the mother of all containers), notify on release
is part of the user visible API of cpusets. The kernel does not remove
cpusets; it runs a user program, /sbin/cpuset_release_agent. That
program might choose to rmdir the cpuset directory, and/or do other
actions, like notify a batch scheduler that one of its cpusets was
released.

That API is not as sacrosanct as some, but it is working, and I wouldn't
want to break it without good reason.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: Containers: css_put() dilemma
Posted by Paul Jackson on Tue, 17 Jul 2007 17:55:09 GMT
View Forum Message <> Reply to Message

Paul M wrote:
> In fact, that's exactly what the release agent
> patch already does.

I'm feeling lazy ;) What's the Subject of that
patch, for my easy searching?

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 17:55:52 GMT
View Forum Message <> Reply to Message

On 7/17/07, Paul Jackson <pj@sgi.com> wrote:
>
> At least for cpusets (the mother of all containers), notify on release
> is part of the user visible API of cpusets. The kernel does not remove
> cpusets; it runs a user program, /sbin/cpuset_release_agent. That
> program might choose to rmdir the cpuset directory, and/or do other
> actions, like notify a batch scheduler that one of its cpusets was
> released.

Right, that's what the release agent patch for process containers does
- there's a file in the hierarchy root called "release_agent" that
contains the path to the program to run if a notify_on_release
container goes idle. When you mount the "cpuset" filesystem, it
automatically populates that file with "/sbin/cpuset_release_agent". A
workqueue task is used to actually do the notifications so that
references can be dropped without having to potentially run a
userspace helper.

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 17:57:49 GMT
View Forum Message <> Reply to Message

On 7/17/07, Paul Jackson <pj@sgi.com> wrote:
> Paul M wrote:
> > In fact, that's exactly what the release agent
> > patch already does.
>
> I'm feeling lazy ;) What's the Subject of that
> patch, for my easy searching?

"Support for automatic userspace release agents"

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Paul Jackson on Tue, 17 Jul 2007 17:58:40 GMT
View Forum Message <> Reply to Message

Paul M wrote:
> Right, that's what the release agent patch for process containers does
> - there's a file in the hierarchy root called "release_agent" that
> contains the path to the program to run if a notify_on_release
> container goes idle. When you mount the "cpuset" filesystem, it
> automatically populates that file with "/sbin/cpuset_release_agent".

Cool -- thanks. Sounds good.

Nevermind my other "what's the Subject" query, if you haven't
already answered it. You just nicely answered my questions.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: Containers: css_put() dilemma
Posted by Balbir Singh on Tue, 17 Jul 2007 18:11:55 GMT
View Forum Message <> Reply to Message

Paul (??) Menage wrote:
> On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>>
>> That sounds correct. I wonder now if the solution should be some form
>> of delegation for deletion of unreferenced containers (HINT: work queue
>> or kernel threads).
>
> What a great idea. In fact, that's exactly what the release agent
> patch already does.

:-) I should have seen that. I am a little lost thinking that
container_rmdir() and the release agent check_for_release() work
without too much knowledge of each other. BTW, what are the semantics
of css_put() is it expected to free the container/run the release agent
when the reference count of the container_subsys_state drops to zero?

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Tue, 17 Jul 2007 18:26:03 GMT
View Forum Message <> Reply to Message

On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> without too much knowledge of each other. BTW, what are the semantics
> of css_put() is it expected to free the container/run the release agent
> when the reference count of the container_subsys_state drops to zero?
>

If you css_put() the last reference on a subsystem state object and
the associated container is marked as notify_on_release, then
check_for_release() is called which does a more full check of whether
the container is releasable. If it is, a workqueue task is scheduled
to run the userspace release agent, which can then do anything it
wants, including potentially deleting the empty container.

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Balbir Singh on Wed, 18 Jul 2007 04:29:28 GMT
View Forum Message <> Reply to Message

Paul (??) Menage wrote:
> On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>> without too much knowledge of each other. BTW, what are the semantics
>> of css_put() is it expected to free the container/run the release agent
>> when the reference count of the container_subsys_state drops to zero?
>>
>
> If you css_put() the last reference on a subsystem state object and
> the associated container is marked as notify_on_release, then
> check_for_release() is called which does a more full check of whether
> the container is releasable. If it is, a workqueue task is scheduled
> to run the userspace release agent, which can then do anything it
> wants, including potentially deleting the empty container.
>

Ok.. so my problem still remains, how do I get a non-blocking atomic
reference increment/decrement routine, that would prevent my
container from being deleted?

I don't find cpusets using css_put(). I was hoping that we could
alter css_* would provide the functionality I need.

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL

Subject: Re: Containers: css_put() dilemma
Posted by Balbir Singh on Wed, 18 Jul 2007 05:30:39 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Paul (??) Menage wrote:
>> On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>>> without too much knowledge of each other. BTW, what are the semantics
>>> of css_put() is it expected to free the container/run the release agent
>>> when the reference count of the container_subsys_state drops to zero?
>>>
>> If you css_put() the last reference on a subsystem state object and
>> the associated container is marked as notify_on_release, then
>> check_for_release() is called which does a more full check of whether

>> the container is releasable. If it is, a workqueue task is scheduled
>> to run the userspace release agent, which can then do anything it
>> wants, including potentially deleting the empty container.
>>
>
> Ok.. so my problem still remains, how do I get a non-blocking atomic
> reference increment/decrement routine, that would prevent my
> container from being deleted?
>
> I don't find cpusets using css_put(). I was hoping that we could
> alter css_* would provide the functionality I need.
>
>

Thinking out loud again, can we add can_destory() callbacks?

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL

Subject: Re: Containers: css_put() dilemma
Posted by Srivatsa Vaddagiri on Wed, 18 Jul 2007 05:52:18 GMT

View Forum Message <> Reply to Message

On Wed, Jul 18, 2007 at 11:00:39AM +0530, Balbir Singh wrote:
> Thinking out loud again, can we add can_destory() callbacks?

I remember suggesting such a callback long before :

 http://marc.info/?l=linux-kernel&m=117576241131788&w=2

--
Regards,
vatsa

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Wed, 18 Jul 2007 06:07:09 GMT
View Forum Message <> Reply to Message

On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> Ok.. so my problem still remains, how do I get a non-blocking atomic
> reference increment/decrement routine, that would prevent my
> container from being deleted?

css_put() in my new patchset will be non-blocking.

>
> I don't find cpusets using css_put().

Cpusets never needs to keep a non-process reference on a cpuset
object, so the framework handles everything.

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Paul Menage on Wed, 18 Jul 2007 23:15:38 GMT
View Forum Message <> Reply to Message

On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> Thinking out loud again, can we add can_destroy() callbacks?
>

What would the exact semantics of such a callback be?

Since for proper interaction with release agents we need the subsystem
to notify the framework when a subsystem object becomes releasable
(currently as part of css_put()), what would a can_destroy() callback
let you do that you couldn't do just by taking an extra css refcount
to prevent destruction and releasing that refcount to allow
destruction?

Paul

Subject: Re: Containers: css_put() dilemma
Posted by Balbir Singh on Thu, 19 Jul 2007 03:44:28 GMT
View Forum Message <> Reply to Message

On 7/19/07, Paul Menage <menage@google.com> wrote:
> On 7/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> >
> > Thinking out loud again, can we add can_destroy() callbacks?
> >
>
> What would the exact semantics of such a callback be?
>
> Since for proper interaction with release agents we need the subsystem
> to notify the framework when a subsystem object becomes releasable
> (currently as part of css_put()), what would a can_destroy() callback
> let you do that you couldn't do just by taking an extra css refcount
> to prevent destruction and releasing that refcount to allow
> destruction?

I was thinking along those lines before you mentioned that the next
version of css_put() will not block. The advantage I see of
can_destory() is that it allows subsystems to do their own reference
counting and decide whether they are ready to be deleted or not. The
other advantage I see is that it can act like a prepare to be deleted
phase for the controller, the controller might decide to take some
action in the can_destroy() phase, like the memory controller could
decide to start reclaiming all the remaining page cache pages.

BTW, do you know upfront as to when the next set of container
enhancement patches will be ready? The css_put() issue is blocking us
currently.

Balbir