Subject: Re: [Patch 1/3] Miscellaneous container fixes
Posted by Paul Menage on Fri, 01 Dec 2006 17:25:54 GMT
View Forum Message <> Reply to Message

On 12/1/06, Srivatsa Vaddagiri <vatsa@in.ibm.com> wrote:
> This patches fixes various bugs I hit in the recently posted container
> patches.
>
> 1. If a subsystem registers with fork/exit hook during bootup (much
>   before rcu is initialized), then the resulting synchronize_rcu() in
>   container_register_subsys() hangs. Avoid this by not calling
>   synchronize_rcu() if we arent fully booted yet.
>
> 2. If cpuset_create fails() for some reason, then the resulting
>   call to cpuset_destroy can trip. Avoid this by initializing
>   container->...->cpuset pointer to NULL in cpuset_create().
>
> 3. container_rmdir->cpuset_destroy->update_flag can deadlock on
>   container_lock(). Avoid this by introducing __update_flag, which
>   doesnt take container_lock().

Ah - this may be the lockup that PaulJ hit.

Thanks for these fixes.

Paul

Subject: Re: [Patch 1/3] Miscellaneous container fixes
Posted by Paul Jackson on Fri, 01 Dec 2006 20:31:34 GMT
View Forum Message <> Reply to Message

Paul M wrote:
> Ah - this may be the lockup that PaulJ hit.

Yes - looks like this fixes it. Thanks, Srivatsa.

And with that fix, it becomes obvious how to reproduce this problem:

```
mount -t cpuset cpuset /dev/cpuset # if not already mounted
cd /dev/cpuset
mkdir foo
echo 1 > foo/cpu_exclusive
rmdir foo    # hangs ...
```

However ...

Read the comment in kernel/cpuset.c for the routine cpuset_destroy().

It explains that update_flag() is called where it is (turning off
the cpu_exclusive flag, if it was set), to avoid the calling sequence:

  cpuset_destroy->update_flag->update_cpu_domains->lock_cpu_hotplug

while holding the callback_mutex, as that could ABBA deadlock with the
CPU hotplug code.

But with this container based rewrite of cpusets, it now seems that
cpuset_destroy -is- called holding the callback_mutex (though I don't
see any mention of that in the cpuset_destroy comment ;), so it would
seem that we once again are at risk for this ABBA deadlock.

I also notice that the comment for container_lock() in the file
kernel/container.c only mentions its use in the oom code. That is
no longer the only, or even primary, user of this lock routine.
The kernel/cpuset.c code uses it frequently (without comment ;),
and I wouldn't be surprised to see other future controllers calling
container_lock() as well.

Looks like its time to update those comments, and think about what
was written there before, as that might catch a bug or two, such as
the one Srivatsa just fixed for us.

Most of those long locking comments in kernel/cpuset.c are there
for a reason - recording the results of a lesson learned in the
school of hard knocks.

--

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

Subject: Re: [Patch 1/3] Miscellaneous container fixes
Posted by Paul Menage on Tue, 05 Dec 2006 12:04:56 GMT
View Forum Message <> Reply to Message

On 12/1/06, Paul Jackson <pj@sgi.com> wrote:
> Read the comment in kernel/cpuset.c for the routine cpuset_destroy().
> It explains that update_flag() is called where it is (turning off
> the cpu_exclusive flag, if it was set), to avoid the calling sequence:
>
>   cpuset_destroy->update_flag->update_cpu_domains->lock_cpu_hotplug
>
> while holding the callback_mutex, as that could ABBA deadlock with the
> CPU hotplug code.

This particular race is gone in the -mm2 kernel since cpus_exclusive
no longer drives sched_domains - can we assume that this will be
reaching mainline some time soon?

>
> But with this container based rewrite of cpusets, it now seems that
> cpuset_destroy -is- called holding the callback_mutex (though I don't
> see any mention of that in the cpuset_destroy comment ;), so it would

And in fact I explicitly documented it as only holding manage_mutex,
not callback_mutex in Documentation/containers.txt. I think maybe this
slipped in during the multi-hierarchy rewrite. :-(

Looking at the various *_destroy() functions in the container
subsystems in my patch set, I think that it should be OK to call the
destructors prior to taking callback_mutex for the unlinking of the
container from its parents.

>
> I also notice that the comment for container_lock() in the file
> kernel/container.c only mentions its use in the oom code. That is
> no longer the only, or even primary, user of this lock routine.
> The kernel/cpuset.c code uses it frequently (without comment ;),
> and I wouldn't be surprised to see other future controllers calling
> container_lock() as well.

As was pointed out by Chandra Seetharaman, it would be nice if we
could avoid having all the container subsystems relying on
callback_mutex for their locking needs - particularly since that's
likely to be acquired at performance-sensitive times.

The cpu_acct and beancounters subsystems that I included in my patch
set both use their own per-container locks for synchronization, so
it's not completely necessary to use the central locks. There's
probably a happy medium between "one big lock" and "way too many small
locks".

Paul