Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by KOSAKI Motohiro on Sun, 22 Jun 2008 15:34:04 GMT
View Forum Message <> Reply to Message

CC'ed Paul Jackson

it seems typical ABBA deadlock.
I think cpuset use cgrou_lock() by mistake.

IMHO, cpuset_handle_cpuhp() sholdn't use cgroup_lock() and
shouldn't call rebuild_sched_domains().

```
 -> #1 (cgroup_mutex){--..}:
       [<c015a435>] __lock_acquire+0xf45/0x1040
       [<c015a5c8>] lock_acquire+0x98/0xd0
       [<c05416d1>] mutex_lock_nested+0xb1/0x300
       [<c0160e6f>] cgroup_lock+0xf/0x20      cgroup_lock
       [<c0164750>] cpuset_handle_cpuhp+0x20/0x180
       [<c014ea77>] notifier_call_chain+0x37/0x70
       [<c014eae9>] __raw_notifier_call_chain+0x19/0x20
       [<c051f8c8>] _cpu_down+0x78/0x240   cpu_hotplug.lock
       [<c051fabb>] cpu_down+0x2b/0x40      cpu_add_remove_lock
       [<c0520cd9>] store_online+0x39/0x80
       [<c02f627b>] sysdev_store+0x2b/0x40
       [<c01d3372>] sysfs_write_file+0xa2/0x100
       [<c0195486>] vfs_write+0x96/0x130
       [<c0195b4d>] sys_write+0x3d/0x70
       [<c010831b>] sysenter_past_esp+0x78/0xd1
       [<ffffffff>] 0xffffffff

 -> #0 (&cpu_hotplug.lock){--..}:
       [<c0159fe5>] __lock_acquire+0xaf5/0x1040
       [<c015a5c8>] lock_acquire+0x98/0xd0
       [<c05416d1>] mutex_lock_nested+0xb1/0x300
       [<c015efbc>] get_online_cpus+0x2c/0x40         cpu_hotplug.lock
       [<c0163e6d>] rebuild_sched_domains+0x7d/0x3a0
       [<c01653a4>] cpuset_common_file_write+0x204/0x440 cgroup_lock
       [<c0162bc7>] cgroup_file_write+0x67/0x130
       [<c0195486>] vfs_write+0x96/0x130
       [<c0195b4d>] sys_write+0x3d/0x70
       [<c010831b>] sysenter_past_esp+0x78/0xd1
       [<ffffffff>] 0xffffffff
```

> Hi,
>
> I decided to see what cgroups is all about, and followed the instructions

> in Documentation/cgroups.txt :-) It happened when I did this:
>
> [root@damson /dev/cgroup/Vegard 0]
> # echo 1 > cpuset.cpus
>
> I can also provide the kernel config if necessary.
>
>
> Vegard
>
>
> ===
> [INFO: possible circular locking dependency detected]
> 2.6.26-rc7 #25
> ---
> bash/10032 is trying to acquire lock:
> (&cpu_hotplug.lock){--..}, at: [<c015efbc>] get_online_cpus+0x2c/0x40
>
> but task is already holding lock:
> (cgroup_mutex){--..}, at: [<c0160e6f>] cgroup_lock+0xf/0x20
>
> which lock already depends on the new lock.

Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by Peter Zijlstra on Sun, 22 Jun 2008 15:50:21 GMT
View Forum Message <> Reply to Message

On Mon, 2008-06-23 at 00:34 +0900, KOSAKI Motohiro wrote:
> CC'ed Paul Jackson
>
> it seems typical ABBA deadlock.
> I think cpuset use cgrou_lock() by mistake.
>
> IMHO, cpuset_handle_cpuhp() sholdn't use cgroup_lock() and
> shouldn't call rebuild_sched_domains().

Looks like Max forgot to test with lockdep enabled...

Well, someone should when you change the online map.

Max, Paul, can we handle this in update_sched_domains() instead?

> -> #1 (cgroup_mutex){--..}:

>   [<c015a435>] __lock_acquire+0xf45/0x1040
>   [<c015a5c8>] lock_acquire+0x98/0xd0
>   [<c05416d1>] mutex_lock_nested+0xb1/0x300
>   [<c0160e6f>] cgroup_lock+0xf/0x20   cgroup_lock
>   [<c0164750>] cpuset_handle_cpuhp+0x20/0x180
>   [<c014ea77>] notifier_call_chain+0x37/0x70
>   [<c014eae9>] __raw_notifier_call_chain+0x19/0x20
>   [<c051f8c8>] _cpu_down+0x78/0x240   cpu_hotplug.lock
>   [<c051fabb>] cpu_down+0x2b/0x40   cpu_add_remove_lock
>   [<c0520cd9>] store_online+0x39/0x80
>   [<c02f627b>] sysdev_store+0x2b/0x40
>   [<c01d3372>] sysfs_write_file+0xa2/0x100
>   [<c0195486>] vfs_write+0x96/0x130
>   [<c0195b4d>] sys_write+0x3d/0x70
>   [<c010831b>] sysenter_past_esp+0x78/0xd1
>   [<ffffffff>] 0xffffffff
>
> -> #0 (&cpu_hotplug.lock){--..}:
>   [<c0159fe5>] __lock_acquire+0xaf5/0x1040
>   [<c015a5c8>] lock_acquire+0x98/0xd0
>   [<c05416d1>] mutex_lock_nested+0xb1/0x300
>   [<c015efbc>] get_online_cpus+0x2c/0x40   cpu_hotplug.lock
>   [<c0163e6d>] rebuild_sched_domains+0x7d/0x3a0
>   [<c01653a4>] cpuset_common_file_write+0x204/0x440 cgroup_lock
>   [<c0162bc7>] cgroup_file_write+0x67/0x130
>   [<c0195486>] vfs_write+0x96/0x130
>   [<c0195b4d>] sys_write+0x3d/0x70
>   [<c010831b>] sysenter_past_esp+0x78/0xd1
>   [<ffffffff>] 0xffffffff
>
>
> > Hi,
> >
> > I decided to see what cgroups is all about, and followed the instructions
> > in Documentation/cgroups.txt :-) It happened when I did this:
> >
> >   [root@damson /dev/cgroup/Vegard 0]
> >   # echo 1 > cpuset.cpus
> >
> > I can also provide the kernel config if necessary.
> >
> >
> > Vegard
> >
> >
> > ===
> > [INFO: possible circular locking dependency detected]
> > 2.6.26-rc7 #25

> > ---
> > bash/10032 is trying to acquire lock:
> > (&cpu_hotplug.lock){--..}, at: [<c015efbc>] get_online_cpus+0x2c/0x40
> >
> > but task is already holding lock:
> > (cgroup_mutex){--..}, at: [<c0160e6f>] cgroup_lock+0xf/0x20
> >
> > which lock already depends on the new lock.
> --
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at http://www.tux.org/lkml/

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

---

Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by Cyrill Gorcunov on Sun, 22 Jun 2008 16:02:02 GMT
View Forum Message <> Reply to Message

[KOSAKI Motohiro - Mon, Jun 23, 2008 at 12:34:04AM +0900]
| CC'ed Paul Jackson
|
| it seems typical ABBA deadlock.
| I think cpuset use cgrou_lock() by mistake.
|
| IMHO, cpuset_handle_cpuhp() sholdn't use cgroup_lock() and
| shouldn't call rebuild_sched_domains().
|
|
| -> #1 (cgroup_mutex){--..}:
| [<c015a435>] __lock_acquire+0xf45/0x1040
| [<c015a5c8>] lock_acquire+0x98/0xd0
| [<c05416d1>] mutex_lock_nested+0xb1/0x300
| [<c0160e6f>] cgroup_lock+0xf/0x20   cgroup_lock
| [<c0164750>] cpuset_handle_cpuhp+0x20/0x180
| [<c014ea77>] notifier_call_chain+0x37/0x70
| [<c014eae9>] __raw_notifier_call_chain+0x19/0x20
| [<c051f8c8>] _cpu_down+0x78/0x240   cpu_hotplug.lock
| [<c051fabb>] cpu_down+0x2b/0x40   cpu_add_remove_lock
| [<c0520cd9>] store_online+0x39/0x80
| [<c02f627b>] sysdev_store+0x2b/0x40
| [<c01d3372>] sysfs_write_file+0xa2/0x100
| [<c0195486>] vfs_write+0x96/0x130

| [<c0195b4d>] sys_write+0x3d/0x70
| [<c010831b>] sysenter_past_esp+0x78/0xd1
| [<ffffffff>] 0xffffffff
|
| -> #0 (&cpu_hotplug.lock){--..}:
| [<c0159fe5>] __lock_acquire+0xaf5/0x1040
| [<c015a5c8>] lock_acquire+0x98/0xd0
| [<c05416d1>] mutex_lock_nested+0xb1/0x300
| [<c015efbc>] get_online_cpus+0x2c/0x40 cpu_hotplug.lock
| [<c0163e6d>] rebuild_sched_domains+0x7d/0x3a0
| [<c01653a4>] cpuset_common_file_write+0x204/0x440 cgroup_lock
| [<c0162bc7>] cgroup_file_write+0x67/0x130
| [<c0195486>] vfs_write+0x96/0x130
| [<c0195b4d>] sys_write+0x3d/0x70
| [<c010831b>] sysenter_past_esp+0x78/0xd1
| [<ffffffff>] 0xffffffff
|
|
| > Hi,
| >
| > I decided to see what cgroups is all about, and followed the instructions
| > in Documentation/cgroups.txt :-) It happened when I did this:
| >
| > [root@damson /dev/cgroup/Vegard 0]
| > # echo 1 > cpuset.cpus
| >
| > I can also provide the kernel config if necessary.
| >
| >
| > Vegard
| >
| >
| > ===
| > [INFO: possible circular locking dependency detected]
| > 2.6.26-rc7 #25
| > ---
| > bash/10032 is trying to acquire lock:
| > (&cpu_hotplug.lock){--..}, at: [<c015efbc>] get_online_cpus+0x2c/0x40
| >
| > but task is already holding lock:
| > (cgroup_mutex){--..}, at: [<c0160e6f>] cgroup_lock+0xf/0x20
| >
| > which lock already depends on the new lock.
|

Thanks Kosaki!

- Cyrill -

Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by Paul Jackson on Mon, 23 Jun 2008 12:02:23 GMT
View Forum Message <> Reply to Message

CC'd Gautham R Shenoy <ego@in.ibm.com>.

I believe that we had the locking relation between what had been
cgroup_lock (global cgroup lock which can be held over large stretches
of non-performance critical code) and callback_mutex (global cpuset
specific lock which is held over shorter stretches of more performance
critical code - though still not on really hot code paths.) One can
nest callback_mutex inside cgroup_lock, but not vice versa.

The callback_mutex guarded some CPU masks and Node masks, which might
be multi-word and hence don't change atomically. Any low level code
that needs to read these these cpuset CPU and Node masks, needs to
hold callback_mutex briefly, to keep that mask from changing while
being read.

There is even a comment in kernel/cpuset.c, explaining how an ABBA
deadlock must be avoided when calling rebuild_sched_domains():

```
/*
 * rebuild_sched_domains()
 *
 * ...
 *
 * Call with cgroup_mutex held.  May take callback_mutex during
 * call due to the kfifo_alloc() and kmalloc() calls.  May nest
 * a call to the get_online_cpus()/put_online_cpus() pair.
 * Must not be called holding callback_mutex, because we must not
 * call get_online_cpus() while holding callback_mutex.  Elsewhere
 * the kernel nests callback_mutex inside get_online_cpus() calls.
 * So the reverse nesting would risk an ABBA deadlock.
```

This went into the kernel sometime around 2.6.18.

Then in October and November of 2007, Gautham R Shenoy submitted
"Refcount Based Cpu Hotplug" (http://lkml.org/lkml/2007/11/15/239)

This added cpu_hotplug.lock, which at first glance seems to fit into
the locking hierarchy about where callback_mutex did before, such as

being invocable from rebuild_sched_domains().

However ... the kernel/cpuset.c comments were not updated to describe
the intended locking hierarchy as it relates to cpu_hotplug.lock, and
it looks as if cpu_hotplug.lock can also be taken while invoking the
hotplug callbacks, such as the one here that is handling a CPU down
event for cpusets.

Gautham ... you there?

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.940.382.4214

_____

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

---

Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by Max Krasnyanskiy on Tue, 24 Jun 2008 06:29:21 GMT
View Forum Message <> Reply to Message

Peter Zijlstra wrote:
> On Mon, 2008-06-23 at 00:34 +0900, KOSAKI Motohiro wrote:
>> CC'ed Paul Jackson
>>
>> it seems typical ABBA deadlock.
>> I think cpuset use cgrou_lock() by mistake.
>>
>> IMHO, cpuset_handle_cpuhp() sholdn't use cgroup_lock() and
>> shouldn't call rebuild_sched_domains().
>
> Looks like Max forgot to test with lockdep enabled...
Hmm, I don't think I actually changed any lock nesting/dependencies. Did I ?
Oh, I see rebuild_sched_domains() is now called from cpuset hotplug handler.
I just looked at the comment for rebuild_sched_domains() and it says
" * Call with cgroup_mutex held. ..." that's why I thought it's safe and it
worked on the test stations.

Anyway, we need definitely need to make rebuild_sched_domains() work from the
hotplug handler.

> Well, someone should when you change the online map.
>
> Max, Paul, can we handle this in update_sched_domains() instead?

That'd be exactly the same as calling rebuild_sched_domains() outside of the
cgroup_lock(). So I do not think it'll help. Paul has more info in his reply
so I'll reply to his email.

Max

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

---

Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by Paul Menage on Thu, 26 Jun 2008 07:25:57 GMT
View Forum Message <> Reply to Message

On Mon, Jun 23, 2008 at 11:29 PM, Max Krasnyansky <maxk@qualcomm.com> wrote:
> Peter Zijlstra wrote:
>> On Mon, 2008-06-23 at 00:34 +0900, KOSAKI Motohiro wrote:
>>> CC'ed Paul Jackson
>>>
>>> it seems typical ABBA deadlock.
>>> I think cpuset use cgrou_lock() by mistake.
>>>
>>> IMHO, cpuset_handle_cpuhp() sholdn't use cgroup_lock() and
>>> shouldn't call rebuild_sched_domains().
>>
>> Looks like Max forgot to test with lockdep enabled...
> Hmm, I don't think I actually changed any lock nesting/dependencies. Did I ?
> Oh, I see rebuild_sched_domains() is now called from cpuset hotplug handler.
> I just looked at the comment for rebuild_sched_domains() and it says
> " * Call with cgroup_mutex held. ..." that's why I thought it's safe and it
> worked on the test stations.
>
> Anyway, we need definitely need to make rebuild_sched_domains() work from the
> hotplug handler.

In that case the obvious solution would be to nest inside
cgroup_lock() inside cpuhotplug.lock. i.e. require that
update_sched_domains() be called inside get_online_cpus(), and call
get_online_cpus() prior to calling cgroup_lock() in any code path that
might call update_sched_domains(). That's basically:

cpuset_write_u64()
cpuset_write_s64()
cpuset_destroy()
common_cpu_hotplug_unplug()
cpuset_write_resmask()

i.e. almost all the cpuset userspace APIs. A bit ugly, but probably
not a big deal given how infrequently CPU hotplug/hotunplug occurs?

Probably simplest with a wrapper function such as:

```
static bool cpuset_lock_live_cgroup(struct cgroup *cgrp)
{
  get_online_cpus();
  if (cgroup_lock_live_cgroup())
    return true;
  put_online_cpus();
  return false;
}

static void cpuset_unlock()
{
  cgroup_unlock();
  put_online_cpus();
}
```

and use those in the relevant entry points in place of
cgroup_lock_live_group()/cgroup_unlock()

Oh, except that cpuset_destroy() is called firmly inside cgroup_mutex,
and hence can't nest the call to cgroup_lock() inside the call to
get_online_cpus().

Second idea - can we just punt the call to rebuild_sched_domains() to
a workqueue thread if it's due to a flag or cpumask change? Does it
matter if the call doesn't happen synchronously? The work handler
could easily nest the cgroup_lock() call inside get_online_cpus() and
then call rebuild_sched_domains()

Paul

Subject: Re: v2.6.26-rc7/cgroups: circular locking dependency
Posted by Max Krasnyanskiy on Thu, 26 Jun 2008 17:45:02 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> Second idea - can we just punt the call to rebuild_sched_domains() to
> a workqueue thread if it's due to a flag or cpumask change? Does it
> matter if the call doesn't happen synchronously? The work handler

> could easily nest the cgroup_lock() call inside get_online_cpus() and
> then call rebuild_sched_domains()

I was thinking about exactly the same thing. I kind of don't like async nature of it. Maybe it's ok but there might be some interesting races with async domain updates.

Max