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