Subject: Re: [PATCH] task containersv11 add tasks file interface fix for cpusets
Posted by David Rientjes on Fri, 12 Oct 2007 15:13:35 GMT
View Forum Message <> Reply to Message

On Thu, 11 Oct 2007, Paul Jackson wrote:

> Hmmm ... I hadn't noticed that sched_hotcpu_mutex before.
>
> I wonder what it is guarding? As best as I can guess, it seems, at
> least in part, to be keeping the following two items consistent:
> 1) cpu_online_map

Yes, it protects against cpu hot-plug or hot-unplug; cpu_online_map is guaranteed to be unchanged while the mutex is being held.

> 2) the per-task cpus_allowed masks
>

It doesn't need to protect the per-task cpus_allowed per se, that's already protected. If a task's cpu affinity changes during a call to set_cpus_allowed(), the migration thread will notice the change when it tries to deactive the task and activate it on the destination cpu. It then becomes a no-op.

That's a consequence of the fact that we can't migrate current and need a kthread, particularly the source cpu's runqueue migration thread, to do it when it's scheduled. A migration request such as that includes a completion variable so that the set_cpus_allowed() waits until it has either been migrated or changed cpu affinity again.

> That is, it seems to ensure that a task is allowed to run on some
> online CPU.
>

Right, the destination cpu will not be hot-unplugged out from underneath the task during migration.

> If that's approximately true, then shouldn't I take sched_hotcpu_mutex
> around the entire chunk of code that handles updating a cpusets 'cpus',
> from the time it verifies that the requested CPUs are online, until the
> time that every affected task has its cpus_allowed updated?
>

Not necessarily, you can iterate through a list of tasks and change their cpu affinity (represented by task->cpus_allowed) by migrating them away while task->cpuset->cpus_allowed remains unchanged. The hotcpu notifier cpuset_handle_cpuhp() will update that when necessary for cpu hot-plug or hot-unplug events.

So it's entirely possible that a cpu will be downed during your iteration
of tasks, but that's fine. Just as long as it isn't downed during the
migration. The cpuset's cpus_allowed will be updated by the hotcpu
notifier and sched_hotcpu_mutex will protect from unplugged cpus around
the set_cpus_allowed() call, which checks for intersection between your
new cpumask and cpu_online_map.

> Furthermore, I should probably guard changes to and verifications
> against the top_cpuset's cpus_allowed with this mutex as well, as it is
> supposed to be a copy of cpu_online_map.
>

The hotcpu notifier protects you there as well.
common_cpu_mem_hotplug_unplug() explicitly sets them.

> And since all descendent cpusets have to have 'cpus' masks that are
> subsets of their parents, this means guarding other chunks of cpuset
> code that depend on the consistency of various per-cpuset cpus_allowed
> masks and cpu_online_map.
>

Same as above, except now you're using
guarantee_online_cpus_mems_in_subtree().

> My current intuition is that this expanded use of sched_hotcpu_mutex in
> the cpuset code involving various cpus_allowed masks would be a good
> thing.
>
> In sum, perhaps sched_hotcpu_mutex is guarding the dispersed kernel
> state that depends on what CPUs are online. This includes the per-task
> and per-cpuset cpus_allowed masks, all of which are supposed to be some
> non-empty subset of the online CPUs.
>

It guards cpu_online_map from being changed while it's held.

> Taking and dropping the sched_hotcpu_mutex for each task, just around
> the call to set_cpus_allowed(), as you suggested above, doesn't seem to
> accomplish much that I can see, and certainly doesn't seem to guard the
> consistency of cpu_online_map with the tasks cpus_allowed masks.
>

It's needed to serialize with other migrations such as sched_setaffinity()
and you can use it since all migrations will inherently need this type of
protection. It makes the new cpumask consistent with cpu_online_map only
so far as that it's a subset; otherwise, set_cpus_allowed() will fail.
The particular destination cpu is chosen as any online cpu, which we know

won't be downed because we're holding sched_hotcpu_mutex.

David

_____

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