Subject: Re: [RFC] Transactional CGroup task attachment
Posted by Paul Menage on Fri, 11 Jul 2008 20:41:34 GMT
View Forum Message <> Reply to Message

On Fri, Jul 11, 2008 at 9:12 AM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> This would ideally be recursive mutexes, Linus does not like recursive mutexes.

But we already have them in function, if not in name, with things like cpu_hotplug.lock, which is an open-coded reader-writer mutex with recursion.

>>
>> int prepare_attach_sleep(struct cgroup_attach_state *state);
>>
>
> Is _sleep really required to be specfied? The function name sounds as if the
> callback processor will sleep.

I wasn't quite sure about the name of that method either. We could call it prepare_attach_maysleep(); just prepare_attach() seems a little under-descriptive.

>> Called with group_mutex (which prevents any other task movement
>> between cgroups) held plus any mutexes/semaphores taken by earlier
>> subsystems's callbacks.
>>
>
> This sounds almost like the BKL for cgroups :)

Yes, cgroup_mutex is indeed currently the BKL for cgroups. I'm working separately to reduce that with things like the per-subsystem hierarchy_mutex, which will also protect against task movements.

>
> 1. Prepare_attach() the subsystem does it's or fails

Did you miss a word here?

> 2. If someone failed, send out failed notifications to successfull callbacks
> 3. On receiving a failed notification (due to a different cgroup failure),
> clients undo their operation (done in prepare_attach())
> 4. If all was successful, move the task and call attached() after the task is
> attached.

That's pretty much what we have - except that I've got two prepare_attach() methods to handle the case that some subsystems might need mutexes and others might need spinlocks in order to handle

synchronization between the move and their resource charge/uncharge
mechanisms - because any mutexes (and memory allocations) need to be
handled before spinlocks.

>
> I read through the rest of it. The sleep/nosleep might make sense (to help the
> task acquire the type of lock it wants to acquire), but isn't sleep a generic
> case for nosleep as well?

Can you clarify what you mean by that?

Paul

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