Subject: Re: [PATCH] task containersv11 add tasks file interface fix for cpusets
Posted by David Rientjes on Sat, 06 Oct 2007 17:54:35 GMT
View Forum Message <> Reply to Message

On Sat, 6 Oct 2007, Paul Jackson wrote:

> This isn't working for me.
>
> The key kernel routine for updating a tasks cpus_allowed
> cannot be called while holding a spinlock.
>
> But the above loop holds a spinlock, css_set_lock, between
> the cgroup_iter_start and the cgroup_iter_end.
>
> I end up generating complaints of:
> BUG: scheduling while atomic
> when I invoke the set_cpus_allowed() above.
>
> Should css_set_lock be a mutex? Locking changes like that
> can be risky.
>

It would probably be better to just save references to the tasks.

```
 struct cgroup_iter it;
 struct task_struct *p, **tasks;
 int i = 0;

 cgroup_iter_start(cs->css.cgroup, &it);
 while ((p = cgroup_iter_next(cs->css.cgroup, &it))) {
  get_task_struct(p);
  tasks[i++] = p;
 }
 cgroup_iter_end(cs->css.cgroup, &it);

 while (--i >= 0) {
  set_cpus_allowed(tasks[i], cs->cpus_allowed);
  put_task_struct(tasks[i]);
 }
```

The getting and putting of the tasks will prevent them from exiting or being deallocated prematurely. But this is also a critical section that will need to be protected by some mutex so it doesn't race with other set_cpus_allowed().

  David

_____
Containers mailing list