Subject: Re: [ckrm-tech] [PATCH 1/7] containers (V7): Generic container system abstracted from cpusets code
Posted by Srivatsa Vaddagiri on Sun, 25 Mar 2007 02:21:09 GMT
View Forum Message <> Reply to Message

On Sat, Mar 24, 2007 at 06:41:28PM -0700, Paul Jackson wrote:
> > the following code becomes racy with cpuset_exit() ...
> >
> > atomic_inc(&cs->count);
> > rcu_assign_pointer(tsk->cpuset, cs);
> > task_unlock(tsk);
>
> eh ... so ... ?
>
> I don't know of any sequence where that causes any problem.
>
> Do you see one?

Let's say we had two cpusets CS1 amd CS2 (both different from top_cpuset).
CS1 has just one task T1 in it (CS1->count = 0) while CS2 has no tasks
in it (CS2->count = 0).

Now consider:

CPU0 (attach_task T1 to CS2)	CPU1 (T1 is exiting)
task_lock(T1);	
oldcs = tsk->cpuset; [oldcs = CS1]	
T1->flags & PF_EXITING? (No)	
	T1->flags = PF_EXITING;
atomic_inc(&CS2->count);	
	cpuset_exit() cs = tsk->cpuset; (cs = CS1)
T1->cpuset = CS2;	
	T1->cpuset = &top_cpuset;
task_unlock(T1);	

CS2 has one bogus count now (with no tasks in it), which may prevent it from being removed/freed forever.

Not just this, continuing further we have more trouble:

```
 ------------------------------------------------------------ --------
CPU0 (attach_task T1 to CS2)   CPU1 (T1 is exiting)
 ------------------------------------------------------------ --------

synchronize_rcu()
        atomic_dec(&CS1->count);
        [CS1->count = 0]

if atomic_dec_and_test(&oldcs->count))
 [CS1->count = -1]
```

We now have CS1->count negative. Is that good? I am uncomfortable ..

We need a task_lock() in cpuset_exit to avoid this race.

--
Regards,
vatsa