

On 5/1/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

```
> > +   if (container_is_removed(cont)) {  
> > +       retval = -ENODEV;  
> > +       goto out2;  
> > +   }  
>  
> Can't we make this check prior to kmalloc() and copy_from_user()?
```

We could but I'm not sure what it would buy us - we'd be optimizing for the case that essentially never occurs.

```
>  
>  
>  
> > +int container_task_count(const struct container *cont) {  
> > +   int count = 0;  
> > +   struct task_struct *g, *p;  
> > +   struct container_subsys_state *css;  
> > +   int subsys_id;  
> > +   get_first_subsys(cont, &css, &subsys_id);  
> > +  
> > +   read_lock(&tasklist_lock);  
>  
> Can be replaced with rcu_read_lock() and rcu_read_unlock()
```

Are you sure about that? I see many users of `do_each_thread()/while_each_thread()` taking a lock on `tasklist_lock`, and only one (`fs/binfmt_elf.c`) that's clearly relying on an RCU critical sections. Documentation?

```
>  
> Any chance we could get a per-container task list? It will  
> help subsystem writers as well.
```

It would be possible, yes - but we probably wouldn't want the overhead (additional ref counts and list manipulations on every fork/exit) of it on by default. We could make it a config option that particular subsystems could select.

I guess the question is how useful is this really, compared to just doing a `do_each_thread()` and seeing which tasks are in the container? Certainly that's a non-trivial operation, but in what circumstances is it really necessary to do it?

Paul

---