Subject: Re: [ckrm-tech] [PATCH 1/7] containers (V7): Generic container system abstracted from cpusets code
Posted by Srivatsa Vaddagiri on Sat, 24 Mar 2007 14:58:00 GMT
View Forum Message <> Reply to Message

On Mon, Feb 12, 2007 at 12:15:22AM -0800, menage@google.com wrote:
> +static int attach_task(struct container *cont, char *pidbuf, char **ppathbuf)
> +{
> + pid_t pid;
> + struct task_struct *tsk;
> + struct container *oldcont;
> + int retval;
> +
> + if (sscanf(pidbuf, "%d", &pid) != 1)
> + return -EIO;
> +
> + if (pid) {
> + read_lock(&tasklist_lock);
> +
> + tsk = find_task_by_pid(pid);
> + if (!tsk || tsk->flags & PF_EXITING) {

This is probably carrying over code from cpuset.c, but :

/me thinks that there is a ugly race here with 'tsk' exiting.
What happens if the tsk is marked PF_EXITING just after this check?
If that happens, then:

> + read_unlock(&tasklist_lock);
> + return -ESRCH;
> + }
> +
> + get_task_struct(tsk);
> + read_unlock(&tasklist_lock);
> +
> + if ((current->euid) && (current->euid != tsk->uid)
> + && (current->euid != tsk->suid)) {
> + put_task_struct(tsk);
> + return -EACCES;
> + }
> + } else {
> + tsk = current;
> + get_task_struct(tsk);
> + }
> +
> + retval = security_task_setscheduler(tsk, 0, NULL);
> + if (retval) {
> + put_task_struct(tsk);

```
> +  return retval;
> + }
> +
> + mutex_lock(&callback_mutex);
> +
> + task_lock(tsk);
> + oldcont = tsk->container;
> + if (!oldcont) {
> +  task_unlock(tsk);
> +  mutex_unlock(&callback_mutex);
> +  put_task_struct(tsk);
> +  return -ESRCH;
> + }
> + atomic_inc(&cont->count);
> + rcu_assign_pointer(tsk->container, cont);
```

Above assignment A1 can race with below assignment A2 in container_exit() :

```
 tsk->container = &top_container; /* the_top_container_hack - see above */
```

What happens if A1 follows after A2? I feel very uncomfortable abt it.

IMO, we need to use task_lock() in container_exit() to avoid this race.

(I think this race already exists in mainline cpuset.c?)

P.S : cpuset.c checks for PF_EXITING twice in attach_task(), while this
patch seems to be checking only once. Is that fine?

--
Regards,
vatsa