Subject: Re: [PATCH 00/10] Containers(V10): Generic Process Containers
Posted by serue on Thu, 07 Jun 2007 00:05:59 GMT
View Forum Message <> Reply to Message

Quoting Paul Jackson (pj@sgi.com):
> > > I wasn't paying close enough attention to understand why you couldn't
> > > do it in two steps - make the container, and then populate it with
> > > resources.
> >
> > Sorry, please clarify - are you saying that now you do understand, or
> > that I should explain?
>
> Could you explain -- I still don't understand why you need this option.
> I still don't understand why you can't do it in two steps - make the
> container, then add cpu/mem separately.

Sure - the key is that the ns subsystem uses container_clone() to automatically create a new container (on sys_unshare() or clone(2) with certain flags) and move the current task into it. Let's say we have done

```
 mount -t container -o ns,cpuset nsproxy /containers
```

and we, as task 875, happen to be in the topmost container:

```
 /containers/
```

Now we fork task 999 which does an unshare(CLONE_NEWNS), or we just clone(CLONE_NEWNS). This will create

```
 /containers/node_999
```

and move task 999 into that container. Except that when it tries attach_task() it is refused by cpuset. So the container_clone() fails, and in turn the sys_unshare() or clone() fails. A login making use of the pam_namespace.so library would fail this way with the ns and cpuset subsystems composed.

We could special case this by having kernel/container.c:container_clone() check whether one of the subsystems is cpusets and, if so, setting the defaults for mems and cpus, but that is kind of ugly. I suppose as a cleaner alternative we could add a container_subsys->inherit_defaults() handler, to be called at container_clone(), and for cpusets this would set cpus and mems to the parent values - sibling exclusive values. If that comes to nothing, then the attach_task() is still refused, and the unshare() or clone() fails, but this time with good reason.

thanks,
-serge