Subject: Re: [ckrm-tech] [PATCH 7/7] containers (V7): Container interface to nsproxy subsystem
Posted by Paul Menage on Fri, 06 Apr 2007 21:54:48 GMT
View Forum Message <> Reply to Message

On 4/5/07, Srivatsa Vaddagiri <vatsa@in.ibm.com> wrote:
>
> The approach I am on currently doesnt deal with dynamically loaded
> modules ..Partly because it allows subsystem ids to be compile-time
> decided

Yes, that part is definitely a good idea, since it removes one of the potential performance complaints that people have compared to hard-coded pointers in a structure.

I've reworked my patches to require subsystems to be declared at compile time.

> and also init_nsproxy.ctlr_data[] can be initialised to default
> values at compile time itself.

> Ok ..by posting rcfs patches, I didn't mean to introduce a "yours" and
> "mine" rift ..honestly. In fact you would notice that they have your
> (sole) copyright still on them! It took me just two days to convert over the
> patches to use nsproxy and come up with the rcfs patches and obviously I
> couldnt have done that without your excellent patches to start with.

OK, sorry if I came across as possessive :-) There are definitely some great ideas in your patches, some of which I've incorporated in my patches as you'll see when I send them out later this afternoon

>
> I am still trying to come to terms with this null groupings and how they
> would be used in real life.
>
> - Can you list a real world use of it?

As a simple job tracking mechanism, without any other implications.

>
> - If they are "inescapable" task groups, how does the first task enter
> such a group, using just the filesystem interface?

Root would be able to move tasks around between containers, as normal.

>
> - If there is no real kernel use for such groups, can this be
> implemented in userspace (user ids, session ids etc)

Not cleanly. (Multiple jobs with the same user, session ids can be changed by the user). Currently in the job-control system I'm working on here, I was tagging any processes introduced in a job with a job-unique extra gid, so we could identify which job a process was in by looking at its group list. But that's a bit ugly.

In a more modern kernel we can just use cpusets without bothering to make distinctions between the memory and cpus in different cpusets, but it seems ugly to have to use a more heavyweight solution than you really need.

In practice, this would be more of a toy/example, since anyone doing job control probably is interested in at least some rudimentary kind of resource tracking/control.

Paul
