Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Paul Menage on Wed, 07 Mar 2007 02:32:07 GMT
View Forum Message <> Reply to Message

Hi Vatsa,

Sorry for the delayed reply - the last week has been very busy ...

On 3/1/07, Srivatsa Vaddagiri <vatsa@in.ibm.com> wrote:
> Paul,
> Based on some of the feedback to container patches, I have
> respun them to avoid the "container" structure abstraction and instead use
> nsproxy structure in the kernel. User interface (which I felt was neat
> in your patches) has been retained to be same.

I'm not really sure that I see the value of having this be part of nsproxy rather than the previous independent container (and container_group) structure. As far as I can see, you're putting the container subsystem state pointers and the various task namespace pointers into the same structure (nsproxy) but then they're remaining pretty much independent in terms of code.

The impression that I'm getting (correct me if I'm wrong) is:

- when you do a mkdir within an rcfs directory, the nsproxy associated with the parent is duplicated, and then each rcfs subsystem gets to set a subsystem-state pointer in that nsproxy

- when you move a task into an rcfs container, you create a new nsproxy consisting of the task's old namespaces and its new subsystem pointers. Then you look through the current list of nsproxy objects to see if you find one that matches. If you do, you reuse it, else you create a new nsproxy and link it into the list

- when you do sys_unshare() or a clone that creates new namespaces, then the task (or its child) will get a new nsproxy that has the rcfs subsystem state associated with the old nsproxy, and one or more namespace pointers cloned to point to new namespaces. So this means that the nsproxy for the task is no longer the nsproxy associated with any directory in rcfs. (So the task will disappear from any "tasks" file in rcfs?)

You seem to have lost some features, including fork/exit subsystem callbacks

>
> What follows is the core (big) patch and the cpu_acct subsystem to serve

> as an example of how to use it. I suspect we can make cpusets also work
> on top of this very easily.

I'd like to see that. I suspect it will be a bit more fiddly than the
simple cpu_acct subsystem.

Paul

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers