Subject: Re: [ckrm-tech] [PATCH 7/7] containers (V7): Container interface to nsproxy subsystem
Posted by Paul Menage on Wed, 04 Apr 2007 07:00:07 GMT
View Forum Message <> Reply to Message

On 4/3/07, Srivatsa Vaddagiri <vatsa@in.ibm.com> wrote:
> On Tue, Apr 03, 2007 at 09:04:59PM -0700, Paul Menage wrote:
> > Have you posted the cpuset implementation over your system yet?
>
> Yep, here:
>
> http://lists.linux-foundation.org/pipermail/containers/2007- March/001497.html
>
> For some reason, the above mail didnt make it into lkml (maybe it
> exceeded the max size allowed). I also have a updated version of that
> which I hope to post as soon as I am done with something else I am
> working on (sigh ..)

OK, looking at that, I see a few problems related to the use of nsproxy and lack of a container object:

- your find_nsproxy() function can return an nsproxy object that's correct in its set of resource controllers but not in its other nsproxy pointers.

- rcfs_rmdir() checks the count on the dentry's nsproxy pointer. But it doesn't check for any of the other nsproxy objects that tasks in the same grouping in this hierarchy might have.

- rcfs_rmdir() sets ns->count to 0. But it doesn't stop anyone else from picking up a reference to that nsproxy from the list. Which could happen if someone else has opened the container's tasks file and is trying to write into it (but is blocked on manage_mutex). You can possibly get around this by completely freeing the namespace and setting dentry->fsdata to NULL before you release manage_mutex (and treat a NULL fsdata as a dead container).

- how do you handle additional reference counts on subsystems? E.g. beancounters wants to be able to associate each file with the container that owns it. You need to be able to lock out subsystems from taking new reference counts on an unreferenced container that you're deleting, without making the refcount operation too heavyweight.

- I think there's value in being able to mount a containerfs with no attached subsystems, simply for secure task grouping (e.g. job tracking). My latest patch set didn't support that, but it's a trivial small change to allow it. How would you do that with no container-like

object?

You could have a null subsystem that does nothing other than let you
attach processes to it, but then you'd be limited to one such
grouping. (Since a given subsystem can only be instantiated in one
hierarchy).

>
> > The drawback to that is that every subsystem has to add a dentry to
> > its state, and handle the processing.
>
> Again this depends on whether every subsystem need to be able to support
> the user-space query you pointed out.

Well, if more than one wants to support it, it means duplicating code
that could equally easily be generically provided.

>
> > >Do you see similar queries coming in for every resource controller object
> > >(show me the path of cpu_acct, cpu_ctl, rss_ctl ... objects to which this
> > >task belongs)? IMO that will not be the case, in which case we can avoid
> > >adding N pointers (N = max hierarchies) in nsproxy just to support queries
> > >of
> > >those sort.
> >
> > OK, I see your argument that putting it in the aggregator probably
> > isn't the best thing to do from a space point of view in the case when
> > the number of aggregators
>
> Sorry that sentence seems to be garbled by some mail router :)

Nope, it got garbled because I didn't proof-read my email sufficiently ...

>
> Did you mean to say "when the number of aggregators sharing the same
> container object are more" ?

Yes. Although having thought about the possibility of null groupings
that I described above, I'm no longer convinced that argument is
valid. It would depend a lot on how people used containers in practice
- whether the number of aggregators was very small (when all
subsystems are in one hierarchy, or in different hierarchies with
isomorphic groupings) or very large (when all subsystems are in
different hierarchies, with orthogonal groupings).

>
> I agree ..Putting N pointers in container_group object just to support
> queries isn't justified at this point, because we don't know whether all

> subsystems need to support such queries.

It's not just to support such queries - that's just one example. There are definitely other uses to having the container pointer directly in the aggregator, including those that I mentioned above.

Paul
