Subject: Re: [PATCH 1/2] rcfs core patch
Posted by Srivatsa Vaddagiri on Fri, 09 Mar 2007 18:14:22 GMT
View Forum Message <> Reply to Message

On Fri, Mar 09, 2007 at 01:48:16AM +0100, Herbert Poetzl wrote:
> > There have been various projects attempting to provide resource
> > management support in Linux, including CKRM/Resource Groups and UBC.
>
> let me note here, once again, that you forgot Linux-VServer
> which does quite non-intrusive resource management ...

Sorry, not intentionally. Maybe it slipped because I haven't seen much res mgmt related patches from Linux Vserver on lkml recently. Note that I -did- talk about VServer at one point in past (http://lkml.org/lkml/2006/06/15/112)!

> the basic 'context' (pid space) is the grouping mechanism
> we use for resource management too

so tasks sharing the same nsproxy->pid_ns is the fundamental unit of resource management (as far as vserver/container goes)?

> > As you know, the introduction of 'struct container' was objected
> > to and was felt redundant as a means to group tasks. Thats where I
> > took a shot at converting over Paul Menage's patch to avoid 'struct
> > container' abstraction and insead work with 'struct nsproxy'.
>
> which IMHO isn't a step in the right direction, as
> you will need to handle different nsproxies within
> the same 'resource container' (see previous email)

Isn't that made simple because of the fact that we have pointers to namespace objects (and not actual objects themselves) in nsproxy?

I mean, all that is required to manage multiple nsproxy's
is to have the pointer to the same resource object in all of them.

In system call terms, if someone does a unshare of uts namespace, he will get into a new nsproxy object sure (which has a pointer to the new uts namespace) but the new nsproxy object will still be pointing to the old resource controlling objects.

> > When we support task movement across resource classes, we need to find a
> > nsproxy which has the right combination of resource classes that the
> > task's nsproxy can be hooked to.
>
> no, not necessarily, we can simply create a new one
> and give it the proper resource or whatever-spaces

That would be the simplest, agreeably. But not optimal in terms of storage?

Pls note that task-movement can be not-so-infrequent (in other words, frequent) in context of non-container workload management.

> why is the filesystem approach so favored for this
> kind of manipulations?
>
> IMHO it is one of the worst interfaces I can imagine
> (to move tasks between spaces and/or assign resources)
> but yes, I'm aware that filesystems are 'in' nowadays

Ease of use maybe. Scripts can be more readily used with a fs-based interface.

--
Regards,
vatsa

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