Subject: Re: [patch -mm 08/17] nsproxy: add hashtable
Posted by Cedric Le Goater on Wed, 13 Dec 2006 15:17:39 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Tue, Dec 12, 2006 at 11:43:38AM +0300, Kirill Korotaev wrote:
>>>>> Even letting the concept of nsproxy escape to user space sounds wrong.
>>>>> nsproxy is an internal space optimization. It's not struct container
>>>>> and I don't think we want it to become that.
>
>>>> i don't agree here. we need that, so does openvz, vserver, people
>>>> working on resource management.
>>>
>>> I think what those projects need is _some_ way to group tasks. I'm
>>> not sure they actually need nsproxies.
>>>
>>> Two tasks in the same container could very well have different
>>> nsproxies.
>
> and typically, they will ...

that means we are missing a container object then, a vps, a vcontext, a
vsomething. nop ?

>> what is container then from your POV?
>
> from my PoV, a container is something keeping
> processes _inside_ which basically requires
> the following elements:
>
> - isolation from other containers
> - virtualization of unique elements
> - limitation on resources
> - policy on all interfaces
>
> the current spaces mostly address the isolation
> and to some degree, the virtualization, which
> is a good thing, but the container also requires
> the resource limitation and the policy, to handle
> interfaces to the outside (should not be new to
> you, actually :)
>
> so the container (may it be represented by a
> structure or not), may reference an nsproxy
> (as we do in the 2.6.19 versions of Linux-VServer)
> but an nsproxy is not the proper element to
> define a container ..

agree. it's not complete.

should we address that by introducing a new object ?
could that be done on per-product basis ? I mean like
in a driver model.

> we also want to be able to have sub spaces inside
> a container, as long as they do not interfere or
> overcome the limitations and policy
>
>>> The nsproxy defines how the pid namespace, and pid<->task
>>> mappings happen for a given task. The init process for a container is
>>> special and might actually appear in more than one pid namespace, while
>>> its children might only appear in one. That means that this init
>>> process's nsproxy can and should actually be different from its
>>> children's. This is despite the fact that they are in the same
>>> container.
>
>> nsproxy has references to all namespaces, not just pid namespace.
>> Thus it is a container "view" effectively.
>
> it is a view into the world of one or more processes,
> but not necessarily the view of all processes inside
> a container :)
>
>> If container is something different, then please define it.
>
> see above ...
>
>>> If we really need this 'container' grouping, it can easily be something
>>> pointed to _by_ the nsproxy, but it shouldn't _be_ the nsproxy.
>
>> You can add another indirection if really want it so much...
>> But is it required?
>> We created nsproxy which adds another level of indirection, but from
>> performance POV it is questinable.
>
> I'm not very happy with the nsproxy abstraction,
> as I think it would be better handled per task,
> and I still have no real world test results what
> overhead the nsproxy indirection causes
>
>> I can say that we had a nice experience, when adding a single
>> dereference in TCP code resulted in ~0.5% performance degradation.
>
> yes, that is what I fear is happening right now
> with the nsproxy ... but I think we need to test
> that, and if it makes sense, switch to task direct

> spaces (as we had before), just more of them ...

getting some figures would be nice and we might also be able
to improve the current nsproxy model.

C.

_____

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