Subject: Re: [patch -mm 08/17] nsproxy: add hashtable
Posted by ebiederm on Tue, 12 Dec 2006 08:57:48 GMT
View Forum Message <> Reply to Message

Kirill Korotaev <dev@sw.ru> writes:

>>
>> 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.
> what is container then from your POV?

A nested instance of user space. User space may unshare things
such as the mount namespace so it can give users the ability to
control their own mounts and the like.

>> 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.
> If container is something different, then please define it.

nsproxy has exactly one instance of all namespaces. A container
in the general case can hold other containers, and near containers
(like processes with separate mount namespaces). As well as
processes.

So nsproxy currently captures the common case for containers but not
the general case.

>> 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 questionable. I can say that we had a nice experience, when adding
> a single dereference in TCP code resulted in ~0.5% performance degradation.

I totally agree with that, nsproxy is something we need to watch from
a performance point of view. nsproxy is primarily a space

optimization to keep from bloating task struct, and possibly a fork
time optimization. At least at the point we added it no one could
measure overhead from using it.

That is one of the reasons I don't want nsproxy to become explicit and
be exported to user space. So if it is a performance problem we can
change the implementation without affecting users.

Eric