Subject: Re: The issues for agreeing on a virtualization/namespaces implementation.
Posted by dev on Mon, 20 Feb 2006 12:08:58 GMT
View Forum Message <> Reply to Message

>> The questions seem to break down into:
>> 1) Where do we put the references to the different namespaces?
>> - Do we put the references in a struct container that we reference
>> from struct task_struct?
>> - Do we put the references directly in struct task_struct?
>
>
> You "cache" task_struct->container->hotsubsys under
> task_struct->hotsubsys.
> We don't change containers other then at clone time, so no coherency
> issue here !!!!
> Which subsystems pointers to "cache", should be agreed by the experts,
> but first approach should always not to cache and go through the container.
agreed. I see no much reason to cache it and make tons of the same
pointers in all the tasks. Only if needed.
Also, in OpenVZ container has many fields intergrated inside, so there
is no additional dereference, but task->container->subsys_field

>> 2) What is the syscall interface to create these namespaces?
>> - Do we add clone flags? (Plan 9 style)
> Like that approach .. flexible .. particular when one has well specified
> namespaces.
mmm, how do you plan to pass additional flags to clone()?
e.g. strong or weak isolation of pids?

another questions:
how do you plan to meet the dependancies between namespaces?
e.g. conntracks require netfilters to be initialized.
network requires sysctls and proc to be initialized and so on.
do you propose to track all this in clone()? huh...

>> - Do we add a syscall (similar to setsid) per namespace?
>> (Traditional unix style)?
can be so...

>> - Do we in addition add syscalls to manipulate containers generically?
>>
>> I don't think having a single system call to create a container and
>> a new
>> instance of each namespace is reasonable as that does not give us a
>> path into the future when we create yet another namespace.
>>
> Agreed.

why do you think so?
this syscalls will start handling this new namespace and that's all.
this is not different from many syscalls approach.

>> 4) How do we implement each of these namespaces?
>> Besides being maintainable are there other constraints?
>>
> Good question... at least with PID and FS two are there ..
>>
>> 6) How do we do all of this efficiently without a noticeable impact on
>> performance?
>> - I have already heard concerns that I might be introducing cache
>> line bounces and thus increasing tasklist_lock hold time.
>> Which on big way systems can be a problem.
this is nothing compared to hierarchy operations.
BTW, heirarchy also introduces complicated resource accounting,
sometimes making it even impossible.

Kirill
