Subject: Re: The issues for agreeing on a virtualization/namespaces implementation.
Posted by Herbert Poetzl on Wed, 08 Feb 2006 04:56:14 GMT
View Forum Message <> Reply to Message

On Tue, Feb 07, 2006 at 03:06:51PM -0700, Eric W. Biederman wrote:
>
> I think I can boil the discussion down into some of the fundamental
> questions that we are facing.
>
> Currently everyone seems to agree that we need something like
> my namespace concept that isolates multiple resources.
>
> We need these for
> PIDS
> UIDS
> SYSVIPC
> NETWORK
> UTSNAME
> FILESYSTEM
> etc.
>
> 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?

no, just let the tasks be in groups of disjunct spaces
so that they can have shared or private structures for
each of the identified spaces

> - Do we put the references directly in struct task_struct?

yes, IMHO that's the way to do it .. Linux-VServer is
moving in this direction for some time now, but we need
to add a special space for context capabilities and
context flags, basically a context struct, similar to
a namespace ...

> 2) What is the syscall interface to create these namespaces?
> - Do we add clone flags?
> (Plan 9 style)

I'd definitely prefer that, maybe if necessary with a
'new' clone syscall which allows to do a little more
than clone does now, e.g.

- clone into container/context/guest

- set space initializations and limits, etc ...

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

doesn't make sense for the creation, but a syscall
for moving between and management of spaces are very
important ...

> - 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.
>
> If we have one syscall per each namespace why would we need a
> container structure?

for the beforementioned permissions and flags, but
you can as well see it as separate 'context' space

> 3) How do we refer to namespaces and containers when we are not members?
> - Do we refer to them indirectly by processes or other objects that
> we can see and are members?

the process will be an unique identifier to the
namespace, but it might not be easy to use it, so
IMHO it might at least make sense to ...

> - Do we assign some kind of unique id to the containers?

have an unique identifier for the context space so
that somebody can cherry pick namespaces from there

(in this case, the context space would hold references
to the other namespaces which are the ones used by
new tasks created into a context, basically a template
for them)

> 4) How do we implement each of these namespaces?
> Besides being maintainable are there other constraints?

extensible and with keeping hierarchical structures
in mind .. would be bad if we could not do sub-contexts
without a complete rewrite

> 5) How do we control the resource inside a namespace starting
>   from a process that is outside of that namespace?

depends on the resource, but limits have to go to the
context structure, so that they apply for the entire
context space, not just for a task

>   - The filesystem mount namespace gave an interesting answer.
>   So it is quite possible other namespaces will give
>   equally interesting and surprising answers.

> 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.

well, we have to be careful with the complexity ...
keep things like refcounting and 'magic' on clone
simple, for the default case ...

> 7) How do we allow a process inside a container to create containers
>   for it's children?

>   - In general this is trivial but there are a few ugly issues
>   here.

a flat implementation will work for a hierarchical
design if certain things are handled properly, just
think resource management for sub-contexts and
changes in the parent's limits ...

> I think these are the key questions of the conversation.
>
>
> Personally so long as we get true namespaces, implemented in a
> performant and maintainable way that a process from the inside can't
> distinguish from what we have now I have no hard requirements.

keep up the good work!

best,
Herbert

> Eric