Subject: Re: The issues for agreeing on a virtualization/namespaces implementation.
Posted by Hubertus Franke on Wed, 08 Feb 2006 14:40:17 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Hubertus Franke <frankeh@watson.ibm.com> writes:
>
>
>>Eric W. Biederman wrote:
>>

>>>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.
>>
>>
>>>   - Do we add a syscall (similar to setsid) per namespace?
>>>   (Traditional unix style)?
>>
>>Where does that approach end .. what's wrong with doing it at clone() time ?
>>Mainly the naming issue. Just providing a flag does not give me name.
>
>
> It really is a fairly even toss up. The usual argument for doing it
> this way is that you will get a endless stream of arguments added to
> fork+exec other wise. Look of posix_spawn or the windows version if
> you want an example. Bits to clone are skirting the edge of a slippery
> slope.
>

So it seems the clone(flags) is a reasonable approach to create new
namespaces. Question is what is the initial state of each namespace?
In pidspace we know we should be creating an empty pidmap !
In network, someone suggested creating a loopback device
In uts, create "localhost"
Are there examples where we rather inherit ? Filesystem ?
Can we iterate the assumption for each subsystem what people thing is right?

IMHO, there is only a need to refer to a namespace from the global context.
Since one will be moving into a new container, but getting out of one
could be prohibitive (e.g. after migration)
It does not make sense therefore to know the name of a namespace in
a different container.

The example you used below by using the pid comes natural, because that already limits visibility.

I am still struggling with why we need new sys_calls.
sys_calls already exist for changing certain system parameters (e.g. utsname)
so to me it boils down to identifying a proper initial state when the
namespace is created.

```
>
>>>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?
>>>   - Do we assign some kind of unique id to the containers?
>>
>>In containers I simply created an explicite name, which ofcourse colides with
>>the
>>clone() approach ..
>>One possibility is to allow associating a name with a namespace.
>>For instance
>>int set_namespace_name( long flags, const char *name ) /* the once we are using
>>in clone */
>>{
>> if (!flag)
>>  set name of container associated with current.
>> if (flag())
>>  set the name if only one container is associated with the
>>namespace(s)
>>  identified .. or some similar rule
>>}
>>
>
>
> What I have done which seems easier than creating new names is to refer
> to the process which has the namespace I want to manipulate.
```

Is then the idea to only allow the container->init to manipulate
or is there need to allow other priviliged processes to perform namespace
manipulation?
Also after thinking about it.. why is there a need to have an external name
for a namespace ?

```
>
>
>>>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.
>>
```

>>Possible to split the lock up now.. one for each pidspace ?
>
>
> At the moment it is worth thinking about. If the problem isn't
> so bad that people aren't actively working on it we don't have to
> solve the problem for a little while, just be aware of it.
>

Agree, just need to be sure we can split it up. But you already keep
a task list per pid-namespace, so there should be no problem IMHO.
If so let's do it now and take it of the table it its as simple as

task_list_lock ::= pspace->task_list_lock

>
>>>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.
>>
>>Speaking of pids only here ...
>>Does it matter, you just hang all those containers hang of init.
>>What ever hierarchy they form is external ...
>
>
> In general it is simple. For resource accounting, and for naming so
> you can migrate a container with a nested container it is a question
> you need to be slightly careful with.

Absolutely, that's why it is useful to have an "external" idea of how
containers are constructed of basic namespaces==subsystems.
The it "simply" becomes a policy. E.g. one can not migrate a container
that has shared subsystems.
Resource accounting I agree, that might required active aggregation
at request time.

-- Hubertus