Subject: Re: The issues for agreeing on a virtualization/namespaces implementation.
Posted by ebiederm on Wed, 08 Feb 2006 05:23:15 GMT
View Forum Message <> Reply to Message

Hubertus Franke <frankeh@watson.ibm.com> writes:

> Eric W. Biederman wrote:
>> I think I can boil the discussion down into some of the fundamental
>> questions that we are facing.
>>
> Man, bearly can keep up with this email load. Addressed some in
> previous thread, but will reiterate under this context.

:)

>> 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?
>>   - 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.
>
>> 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.

```
>> 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.

```
>> 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.

```
>> 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.

Eric