Subject: Re: The issues for agreeing on a virtualization/namespaces implementation.
Posted by Hubertus Franke on Tue, 07 Feb 2006 23:35:52 GMT
View Forum Message <> Reply to Message

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.

>   - 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.

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

>
>
> 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 ..

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

Possible to split the lock up now.. one for each pidspace ?

>
> 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 ...

>
> 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.
>
>
> Eric
>

-- Hubertus