Subject: Re: semantics for namespace naming
Posted by ebiederm on Thu, 14 Dec 2006 05:41:20 GMT
View Forum Message <> Reply to Message

Temporarily restricting myself to system containers because
they are well defined.

We have two things we need to name.
- Entire containers.
- Namespaces inside of a container.

So far Cedric's suggestion is a peculiar way of naming namespaces.
Which for the bind_ns is probably what we want, but it is not
what we want for container identification.

Entire container identification.

All process in unix are organized into a process tree.

Every system container has a unique init process that always
exists for the life of the container.

In the process tree the descendants of that init process are
the process in the specified container.

So if we remember the init process to container mapping we
can find the containers of any process merely by following
the parent process (ppid) up the process tree.

Which leads to the strong suggestion that for application containers
we somehow maintain the cohesiveness of the process tree.

To be clear, the unit of checkpoint/restart/migration is the
container. So far only our definitions of system containers have
managed the state properly for checkpoint/restart but ideally
application containers should be designed so we can do that as
well.

The basic operations on a whole container are pretty much:
suspend/restart, checkpoint/restart/migration, kill, accounting
and user display.

Namespace identification.

We also need a way to talk about individual namespaces.

We need this so we can clearly export to user space which process

share a namespace and which processes don't. Allowing us to talk
clearly about the group of process that share that namespace, as
well as give us the opportunity to debug reference counting problems.

For functionality like bind_ns per identifiers that clearly identify
a namespace are what we really want.

Debugging

Capturing a checkpoint of a set of processes and debugging a set of
processes is a very similar operation. Entering a namespace and
debugging processes in a namespace is a very similar operation.

I can currently manipulate processes in namespace, and by
manipulating those processes create new processes in a namespace
with sys_ptrace, the standard debugging facility.

I have yet to look at the possibilities in great detail but it looks
to me that what we want for containers is an enhancement of our
debugging mechanisms. So we can do the inspection and manipulation
we find desirable.

The classic enter implementation seems weak and error prone
when compared to what the current sys_ptrace can do and what we would
like to do in terms of checkpoint restart.

One of the issues is for good long term support is that we want
interfaces that are either absolutely trivial to implement or
interfaces that large numbers of people will use. The more people
using an interface the more free testers and fixers we get and
the higher the priority of keeping our code working.

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Let's say we have a vserver, from which we start some jobs
> which we want to checkpoint/restart/migrate. These are two
> of the usages we currently foresee for the namespaces, though
> I'd say it's safe to assume there will be more.
>
> I'll want to be able to address the c/r jobs by some ID in
> order to checkpoint and kill them. I'll also want to be
> able to address the entire vserver by some ID, in order to
> kill it. In that case the c/r jobs should also be killed.
> So those jobs are known by at least two id's. Furthermore, I
> may want two vservers on the same machine, both running a c/r

> job called 'calculate_pi'.
>
> So we can look at this as a filesystem. In the above scenario,
> we've got /sergesvserver, /sergesvserver/calculate_pi,
> /randomvserver, and /randomvserver/calculate_pi. And, if
> user hallyn logs into /sergesvserver using pam_namespace.so,
> unsharing his mounts namespace to get a private /tmp and /home,
> then he ends up in /sergesvserver/unnamed1. So each nsproxy
> has a node in the namespace id filesystem, with random names
> unless/until it is renamed to a more meaningful name. This
> allows us to switch to a vserver by specifying the vserver's
> name (ln /sys/namespaces/vserver1 /proc/nsproxy or whatever
> semantics we end up using), kill an entire vserver recursively
> (rm -rf /sys/namespaces/vserver1), perhaps even checkpoint
> (tar jcf /tarballs/vserver1 /sys/namespaces/vserver1) and
> certainly rename (mv /sys/namespaces/unnamed1
> /sys/namespaces/sergeprivhome).

I certainly see merit in using a file system interface for some
aspects of namespace manipulation. As much as possible we want
to keep to the old interfaces but that should not be a big deal.

> One key observeration which I haven't made explicit is that you
> never actually leave a nsid ("container"). If you start under
> /vserver1, you will always be under /vserver1. I don't know of
> any reason that would not be appropriate. If I start a nested
> vserver from there, then to me it may be known as
> 'vserver_testme', while to the admin of the machine, it would be
> known as /vserver1/vserver_testme.

Yes. Although on the crazy suggestion from I have heard
pivot_container suggested... Which may have some merit for the
software suspend story but otherwise doesn't seem useful...

> This makes one possible implementation of the container struct:
>
> struct container {
> struct container *parent;
> char *name;
> struct nsproxy *nsproxy;
> struct list_head children;
> };
> struct nsproxy {
> ...
> struct container *container;
> };

For your chosen struct container I guess if the hierarchy are

struct containers that will work. Going from your struct container
to anything interesting is currently a walk through the process list
which is painful. So I would suggest putting a pointer to the
init process of the container, that is probably better than the
nsproxy.

I'm not quite convinced we need the struct container. But I have
no fundamental objects to it either.

> Plus of course relevant sysfs stuff.

/proc is actually the appropriate filesystem for this sort of
information not sysfs. Handling the network information that
is in sysfs is going to be hard enough.

Eric

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers