Subject: Re: Roadmap for features planed for containers where and Some future features ideas.

Posted by Oren Laadan on Tue, 22 Jul 2008 14:05:27 GMT

View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> "Peter Dolding" <oiaohm@gmail.com> writes:
>
>> On Mon, Jul 21, 2008 at 10:13 PM, Eric W. Biederman
>> <ebiederm@xmission.com> wrote:
>>> "Peter Dolding" <oiaohm@gmail.com> writes:
>>>
>>>> http://opensolaris.org/os/community/brandz/ I would like to see if
>>>> something equal to this is on the roadmap in particular.   Being able
>>>> to run solaris and aix closed source binaries contained would be
>>>> useful.
>>> There have been projects to do this at various times on linux. Having
>>> a namespace dedicated to a certain kind of application is no big deal.
>>> Someone would need to care enough to test and implement it though.
>>>
>>>> Other useful feature is some way to share a single process between PID
>>>> containers as like a container bridge. For containers used for
>>>> desktop applications not having a single X11 server interfacing with
>>>> video card is a issue.
>>> X allows network connections, and I think unix domain sockets will work.
>>> The latter I need to check on.
>> Does to a point until you see that local X11 is using shared memory
>> for speed.   Hardest issue is getting GLX working.
>
> That is easier in general. Don't unshare the sysvipc namespace.
> Or share the mount of /dev/shmem at least for the file X cares about.
>
>>> The pid namespace is well defined and no a task will not be able
>>> to change it's pid namespace while running. That is nasty.
>> Ok if that is imposable to extremely risky.
>>
>> What about a form of a proxy pid in the pid namespace proxying
>> application chatter between 1 name space to another. Applications
>> being the bridge if its not possible to do it invisible to application
>> could be made aware of it.   So they can provide shared memory and the
>> like across pid namespaces. But only where they have a activated proxy
>> to do there bidding. This also allows applications to maintain there
>> own internal secuirty between namespaces.
>>
>> Ie application is 1 pid number in its source container and virtual pid
>> numbers in the following containers. Symbolic linking at task level
>> yes a little warped. Yes this will annoying mean a special set of
>> syscalls and a special set of capabilities and restrictions. Like PID

>> containers starting up forbidding proxy pid's or allowing them.
>>
>> If I am thinking right that avoids not be able to change it's pid.
>> Instead sending and receiving the messages you need in the other name
>> space threw a small proxy.   Yes I know that will cost some
>> performance.
>
> Proxy pids don't actually do anything for you, unless you want to send
> signals. Because all of the namespaces are distinct. So even at the
> best of it you can see the X server but it still can't use your
> network sockets or ipc shm.
>
> Better is working out the details on how to manipulate multiple
> sysvipc and network namespaces from a single application. Mostly
> that is supported now by the objects there is just no easy way
> of dealing with it.
>
>> Basically want to setup a neat universal container way of handling
>> stuff like http://www.cs.toronto.edu/~andreslc/xen-gl/ without having
>> to go network and hopefully in a way that limitations don't have to
>> exist since messages are really only be sent threw 1 X11 server to 1
>> driver system. Only thing is really sending the correct messages to
>> the correct place.   There will most likely be other services were a
>> single entity at times is preferred.   Worst out come is if proxying
>> .so is required.
>
> Yes. I agree that is essentially desirable. Given that I think
> high end video card actually have multiple hardware contexts that
> can be mapped into different user space processes there may be other
> ways of handling this.
>
> Ideally we can find a high performance solution to X that also gives
> us good isolation and migration properties. Certainly something to talk
> about tomorrow in the conference.

In particular, if you wish to share private resources of a container
between more than a single container, then you won't be able to use
checkpoint/restart on neither container (unless you make special
provisions in the code).

I agree with Eric that the way to handle this is via virtualization
as opposed to direct sharing. The same goes for other hardware, e.g.
in the context of a user desktop - /dev/rtc, sound, and so on. My
experience is that a proxy/virtualized device is what we probably
want.

Oren.

>
> Eric
>
> ____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers