Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view Posted by Herbert Poetzl on Tue, 27 Jun 2006 16:02:42 GMT

View Forum Message <> Reply to Message

```
On Tue, Jun 27, 2006 at 05:52:52AM -0600, Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
>>>>My point is that if you make namespace tagging at routing time,
>>>>and your packets are being routed only once, you lose the ability
>>>>to have separate routing tables in each namespace.
> >>>
>>>Right. What is the advantage of having separate the routing tables?
>>> Routing is everything. For example, I want namespaces to have their
>>> private tunnel devices. It means that namespaces should be allowed
>>> have private routes of local type, private default routes, and so
>>> on...
> >>
> >
>> Ok, we are talking about the same things. We do it only in a different way:
> >
>> * separate routing table :
     namespace
> >
     \--- route_tables
> >
> >
> >
      \---routes
> >
>> * tagged routing table :
    route tables
> >
> >
     \---routes
> >
      \---namespace
> >
> There is a third possibility, that falls in between these two if local
> communication is really the bottle neck.
> We have the dst cache for caching routes and cache multiple
> transformations that happen on a packet.
> With a little extra knowledge it is possible to have the separate
> routing tables but have special logic that recognizes the local
> tunnel device that connects namespaces and have it look into the next
> namespaces routes, and build up a complete stack of dst entries of
> where the packet needs to go.
> I keep forgetting about that possibility. But as long as everything is
> done at the routing layer that should work.
```

> > I use the second method, because I think it is more effecient and
> > reduce the overhead. But the isolation is minimalist and only aims
> > to avoid the application using ressources outside of the container
> > (aka namespace) without taking care of the system. For example, I
> > didn't take care of network devices, because as far as see I can't
> > imagine an administrator wanting to change the network device name
> > while there are hundred of containers running. Concerning tunnel
> > devices for example, they should be created inside the container.
>
> Inside the containers I want all network devices named eth0!

huh? even if there are two of them? also tun?

I think you meant, you want to be able to have eth0 in _more_ than one guest where eth0 in a guest can also be/use/relate to eth1 on the host, right?

> I think, private network ressources method is more elegant
> and involves more network ressources, but there is probably a
> significant overhead and some difficulties to have __lightweight__
> container (aka application container), make nfs working well,
> etc... I did some tests with the loopback with the
> private namespace and there is roughly an overhead of 4 % without
> the isolation since with the tagging method there is 1 % with the
> isolation.
>
The overhead went down?

yes, this might actually happen, because the guest has only to look at a certain subset of entries but this needs a lot more testing, especially with a lot of guests

> The network namespace aims the isolation for now, but the container > based on the namespaces will probably need checkpoint/restart and > migration ability. The migration is needed not only for servers but > for HPC jobs too.
> Yes.
> So I don't know what level of isolation/virtualization is really > needed by users, what should be acceptable (strong isolation and > overhead / weak isolation and efficiency). I don't know if people > wanting strong isolation will not prefer Xen (cleary with much more

well, Xen claims something below 2% IIRC, and would

> > overhead than your patches;))

be clearly the better choice if you want strict separation with the complete functionality, especially with hardware support

> We need a clean abstraction that optimizes well.

>

- > However local communication between containers is not what we
- > should benchmark. That can always be improved later. So long as
- > the performance is reasonable. What needs to be benchmarked is the
- > overhead of namespaces when connected to physical networking devices
- > and on their own local loopback, and comparing that to a kernel
- > without namespace support.

well, for me (obviously advocating the lightweight case) it seems improtant that the following conditions are met:

- loopback traffic inside a guest is insignificantly slower than on a normal system
- loopback traffic on the host is insignificantly slower than on a normal system
- inter guest traffic is faster than on-wire traffic, and should be withing a small tolerance of the loopback case (as it really isn't different)
- network (on-wire) traffic should be as fast as without the namespace (i.e. within 1% or so, better not really measurable)
- all this should be true in a setup with a significant number of guests, when only one guest is active, but all other guests are ready/configured
- all this should scale well with a few hundred guests
- > If we don't hurt that core case we have an implementation we can
- > merge. There are a lot of optimization opportunities for local
- > communications and we can do that after we have a correct and accepted
- > implementation. Anything else is optimizing too soon, and will
- > just be muddying the waters.

what I fear is that once something is in, the kernel will just become slower (as it already did in some areas) and nobody will care/be-able to fix that later on ...

best, Herbert Page 4 of 4 ---- Generated from OpenVZ Forum