Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Tue, 27 Jun 2006 16:47:29 GMT
View Forum Message <> Reply to Message

Herbert Poetzl <herbert@13thfloor.at> writes:

> On Tue, Jun 27, 2006 at 05:52:52AM -0600, Eric W. Biederman wrote:
>>
>> 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?

Right I want to have an eth0 in each guest where eth0 is
it's own network device and need have no relationship to
eth0 on the host.

>> 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

Ultimately I agree. However. Only host performance should be
a merge blocker. Allowing us to go back and reclaim the few
percentage points we lost later.

>> 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 ...

If nobody cares it doesn't matter.

If no one can fix it that is a problem. Which is why we need
high standards and clean code, not early optimizations.

But on that front each step of the way must be justified on
it's own merits. Not because it will give us some holy grail.

The way to keep the inter guest performance from degrading is
to measure it an complain. But the linux network stack is too
big to get in one pass.

Eric