Subject: Re: [RFD] L2 Network namespace infrastructure
Posted by ebiederm on Sat, 23 Jun 2007 17:19:34 GMT
View Forum Message <> Reply to Message

Patrick McHardy <kaber@trash.net> writes:

> Eric W. Biederman wrote:
>> -- The basic design
>>
>> There will be a network namespace structure that holds the global
>> variables for a network namespace, making those global variables
>> per network namespace.
>>
>> One of those per network namespace global variables will be the
>> loopback device. Which means the network namespace a packet resides
>> in can be found simply by examining the network device or the socket
>> the packet is traversing.
>>
>> Either a pointer to this global structure will be passed into
>> the functions that need to reference per network namespace variables
>> or a structure that is already passed in (such as the network device)
>> will be modified to contain a pointer to the network namespace
>> structure.
>
>
> I believe OpenVZ stores the current namespace somewhere global,
> which avoids passing the namespace around. Couldn't you do this
> as well?

It sucks. Especially in the corner cases. Think macvlan
with the real network device in one namespace and the ``vlan''
device in another device.

The implementation of a global is also pretty a little questionable.
Last I looked it didn't work on the transmit path at all and
interesting on the receive path.

Further and fundamentally all a global achieves is removing the need
for the noise patches where you pass the pointer into the various
functions. For long term maintenance it doesn't help anything.

All of the other changes such as messing with the
initialization/cleanup and changing access to access the per network
namespace data structure, and modifying the code partly along the way
to reject working in other non-default network namespaces that are
truly intrusive we both still have to make.

So except as an implementation detail how we pass the per network

namespace pointer is uninteresting.

Currently I am trying for the least clever most straight forward implementation I can find, that doesn't give us a regression in network stack performance.

So yes if we want to do passing through a magic per cpu global on the packet receive path now is the time to decide to do that. Currently I don't see the advantage in doing that so I'm not suggesting it.

In general if there are any specific objections people have written complicated code that allows us to avoid those objections, so it should just be a matter of dusting those patches off. I would much rather go with something stupid and simple if people are willing to merge that however.

>> Depending upon the data structure it will either be modified to hold
>> a per entry network namespace pointer or it there will be a separate
>> copy per network namespace. For large global data structures like
>> the ipv4 routing cache hash table adding an additional pointer to the
>> entries appears the more reasonable solution.
>
>
> So the routing cache is shared between all namespaces?

Yes. Each namespaces has it's own view so semantically it's not shared. But the initial fan out of the hash table 2M or something isn't something we want to replicate on a per namespace basis even assuming the huge page allocations could happen.

So we just tag the entries and add the network namespace as one more part of the key when doing hash table look ups.

>> --- Performance
>>
>> In initial measurements the only performance overhead we have been
>> able to measure is getting the packet to the network namespace.
>> Going through ethernet bridging or routing seems to trigger copies
>> of the packet that slow things down. When packets go directly to
>> the network namespace no performance penalty has yet been measured.
>
>
> It would be interesting to find out whats triggering these copies.
> Do you have NAT enabled?

I would have to go back and look. There was a skb_cow call someplace in the routing path. Something else with ipfilter, ethernet bridging.

So yes it is probably interesting to dig into.

So the thread where we dug into this last time to the point of
identifying the problem is here:
https://lists.linux-foundation.org/pipermail/containers/2007-March/004309.html
The problem in the bridging was here:
https://lists.linux-foundation.org/pipermail/containers/2007-March/004336.html
I can't find a good pointer to the bit of discussion that described
the routing. I just remember it was an skb_cow somewhere in the
routing output path, I believe at the point where we write in the new
destination IP. I haven't a clue why the copy was triggering.

Design wise the interesting bit was it nothing was measurable when the
network device was in the network namespace. So adding an extra
pointer parameter to functions and dereferencing the pointer has not
measurably affected performance at this point.

Eric

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