Subject: Re: Network virtualization/isolation
Posted by jamal on Mon, 04 Dec 2006 13:44:54 GMT
View Forum Message <> Reply to Message

On Mon, 2006-04-12 at 05:15 -0700, Eric W. Biederman wrote:
> jamal <hadi@cyberus.ca> writes:
>

> Containers are a necessary first step to getting migration and checkpoint/restart
> assistance from the kernel.

Isnt it like a MUST have if you are doing things from scratch instead of
it being an after thought.

>
> > 2) the socket level bind/accept filtering with multiple IPs. From
> > reading what Herbert has, it seems they have figured a clever way to
> > optimize this path albeit some challenges (speacial casing for raw
> > filters) etc.
> >
> > I am wondering if one was to use the two level muxing of the socket
> > layer, how much more performance improvement the above scheme provides
> > for #2?
>
> I don't follow this question.

if you had the sockets tables being in two level mux, first level to
hash on namespace which leads to an indirection pointer to the table
to find the socket and its bindings (with zero code changes to the
socket code), then isnt this "fast enough"? Clearly you can optimize as
in the case of bind/accept filtering, but then you may have to do that
for every socket family/protocol (eg netlink doesnt have IP addresses,
but the binding to multiple groups is possible)

Am i making any more sense? ;->

> > Consider the case of L2 where by the time the packet hits the socket
> > layer on incoming, the VE is already known; in such a case, the lookup
> > would be very cheap. The advantage being you get rid of the speacial
> > casing altogether. I dont see any issues with binds per multiple IPs etc
> > using such a technique.
> >
> > For the case of #1 above, wouldnt it be also easier if the tables for
> > netdevices, PIDs etc were per VE (using the 2 level mux)?
>
> Generally yes. s/VE/namespace/. There is a case with hash tables where
> it seems saner to add an additional entry because hash it is hard to dynamically
> allocate a hash table, (because they need something large then a

> single page allocation).

A page to store the namespace indirection hash doesnt seem to be such a
big waste; i wonder though why you even need a page. If i had 256 hash
buckets with 1024 namespaces, it is still not too much of an overhead.

> But for everything else yes it makes things
> much easier if you have a per namespace data structure.

Ok, I am sure youve done the research; i am just being a devils
advocate.

> A practical question is can we replace hash tables with some variant of
> trie or radix-tree and not take a performance hit. Given the better scaling of
> tress to different workload sizes if we can use them so much the
> better. Especially because a per namespace split gives us a lot of
> good properties.

Is there a patch somewhere i can stare at that you guys agree on?

> Well we rather expect to bash heads until we can come up with something
> we all can agree on with the people who more regularly have to maintain
> the code. The discussions so far have largely been warm ups, to actually
> doing something.
>
> Getting feedback from people who regularly work with the networking stack
> is appreciated.

I hope i am being helpful;
It seems to me that folks doing the different implementations may have
had different apps in mind. IMO, as long as the solution caters for all
apps (can you do virtual bridges/routers?), then we should be fine.
Intrusiveness may not be so bad if it needs to be done once. I have to
say i like the approach where the core code and algorithms are
untouched. Thats why i am humping on the two level mux approach, where
one level is to mux and find the namespace indirection and the second
step is to use the current datastructures and algorithms as is. I dont
know how much more cleaner or less intrusive you can be compared to
that. If i compile out the first level mux, I have my old net stack as
is, untouched.

cheers,
jamal