Subject: Re: Network virtualization/isolation
Posted by ebiederm on Sat, 25 Nov 2006 08:21:39 GMT
View Forum Message <> Reply to Message

jamal <hadi@cyberus.ca> writes:

> On Fri, 2006-27-10 at 11:10 +0200, Daniel Lezcano wrote:
>
>> No, it uses virtualization at layer 2 and I had already mention it
>> before (see the first email of the thread), but thank you for the email
>> thread pointer.
>
>
> What would be really useful is someone takes the time and creates a
> matrix of the differences between the implementations.
> It seems there are quiet a few differences but without such comparison
> (to which all agree to) it is hard to form an opinion without a document
> of some form.
>
> For one, I am puzzled by the arguements about L2 vs L3 - Is this the
> host side or inside the VE?
>
> If it is a discussion of the host side:
> To me it seems it involves the classification of some packet header
> arriving on a physical netdevice on the host side (irrelevant whether
> they are L2 or L7) and reaching a decision to select some redirected to
> virtual netdevice.

There are two techniques in real use.
- Bind/Accept filtering

  Which layer 3 addresses a socket can bind/accept are filtered, but otherwise
  the network stack remains unchanged. When your container/VE only has
  a single IP address this works great. When you get multiple IPs
  this technique starts to fall down because it is not obvious how to
  make this correctly handle wild card ip addresses. This technique
  also falls down because it is very hard to support raw IP packets.

  The major advantage of this approach is that it is insanely simple
  and cheap.

  When the discussion started this is what I called Layer 3, bind
  filtering is probably a more precise name.

- Network object tagging

  Every network device, each socket, each routing table, each net
  filter table, everything but the packets themselves is associated

with a single VE/container. In principle the network stack doesn't change except everything that currently access global variables gets an additional pointer indirection.

To find where a packet is you must look at it's network device on ingress, and you must look at it's socket on egress.

This allows capabilities like CAP_NET_ADMIN to be fairly safely given to people inside a container without problems.

There are two basic concerns here.
1) This is a lot of code that needs to be touched.
2) There are not enough physical network devices to go around so we need something that maps packets coming in a physical network device into multiple virtual network devices.

   The obvious way to do this mapping is with either ethernet bridging or with the linux routing code if the external network is not ethernet, and some tunnel devices between the VE and the host environment. This allows firewalling and in general the full power of the linux network stack.

   The concern is that the extra trip through the network stack adds overhead.

This is the technique we have been calling layer two because it works below the IP layer and as such should work for everything.

There have been some intermediate solutions considered but generally the have the down sides of additional expense without the upsides of more power.

> The admin (on the host) decides what packets any VE can see.
> Once within the VE, standard Linux net stack applies. The same applies
> on the egress. The admin decides what packets emanating from the VE
> go where.
> I dont think this is a simple L2 vs L3. You need to be able to process
> IP as well as Decnet[1]

Except for some implementation details I don't think we have disagreement about what we are talking about although there is certainly a little confusion. The true issue is can we implement something that places the full power of the network stack (including things like creating virtual tunnel devices) into the container/VE without sacrificing performance.

I think we could all agree on the most general technique if we could convince ourselves the overhead was unmeasurable.

Given that performance is the primary concern this is something a network stack expert might be able to help with. My gut feel is the extra pointer indirection for the more general technique is negligible and will not affect the network performance. The network stack can be very sensitive to additional cache misses so I could be wrong. Opinions?

Then the question is how do we reduce the overhead when we don't have enough physical network interfaces to go around. My feeling is that we could push the work to the network adapters and allow single physical network adapters to support multiple network interfaces, each with a different link-layer address. At which point the overhead is nearly nothing and newer network adapters may start implementing enough filtering in hardware to do all of the work for us.

> [1] Since Linux has the only SMP-capable, firewall-capable Decnet
> implementation - wouldnt it be fun to have it be virtualized as
> well? ;->

Yes. The only problem I have seen with Decnet is the pain of teaching it that it's variables aren't global anymore. Not a show stopper but something that keeps Decnet out of existing proof of concept implications.

Eric

p.s. Sorry for the long delayed reply. I have had my head down stabilizing 2.6.19 and netdev is not a list I regularly watch.

p.p.s. I have CC'd the containers list so we catch all of the relevant people on the discussion. I believe this is an open list so shouldn't cause any problems.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Sat, 25 Nov 2006 09:09:20 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <dlezcano@fr.ibm.com> writes:

>> Then a matrix of how each requires what modifications in the network
>> code. Of course all players need to agree that the description is

>> accurate.
>> Is there such a document?
>> cheers,
>> jamal
>
> Hi,
>
> the attached document describes the network isolation at the layer 2 and at the
> layer 3, it presents the pros and cons of the different approaches, their common
> points and the impacted network code.
> I hope it will be helpful :)

Roughly it is correctly but I the tradeoffs you describe are incorrect.

> Isolating and virtualizing the network
> ---
>
> Some definitions:
> ---
>
> isolation : This is a restrictive technique which divides a set of the
> available system objects to smaller subsets assigned to a group of
> processes. This technique ensures an application will use only a
> subset of the system resources and will never access other
> resources.
>
> virtualization : This technique gives the illusion to an application
> that its owns all the system resources instead of a subset of them
> provided by the isolation.
>
> container: it is the name of the base element which brings the
> isolation and the virtualization where applications are running into.
>
> system container: operating system running inside a container.
>
> application container : application running inside a container.
>
> checkpoint/restart: take a snapshot of a container at a given time
> and recreate the container from this snapshot.
>
> mobility: checkpoint/restart used to move a container to one host to
> another host.
>
> ---
>
> Actually, containers are being developed in the kernel with the
> following functions :

>
> * separate the system resources between containers in order
> to avoid an application, running into a container, to
> access the resources outside the container. That
> facilitates the resources management, ensures the
> application is jailed and increases the security.
>
> * virtualize the resources, that avoids resources conflict
> between containers, that allows to run several instance of
> the same servers without modifying its network
> configuration.
>
> * the combination of the isolation and the virtualization is
> the base for the checkpoint/restart. The checkpoint is
> easier because the resources are identified by container
> and the restart is possible because the applications can
> be recreated with the same resources identifier without
> conflicts. For example, the application has the pid 1000,
> it is checkpointed and when it is restarted the same pid
> is assigned to it and it will not conflict because pids are
> isolated and virtualized.
>
> In all the system resources, the network is one of the biggest part
> to isolate and virtualize. Some solutions were proposed, with
> different approaches and different implementations.
>
> Layer 2 isolation and virtualization
> ----------

Guys this is probably where we need to focus and not look at the
other options until we find insurmountable challenges with this.
We can make it do everything everyone needs.

> The virtualization acts at the network device level. The routes and
> the sockets are isolated. Each container has its own network device
> and its own routes. The network must be configured in each container.
>
> This approach brings a very strong isolation and a perfect
> virtualization for the system containers.
>
>
> - Ingress traffic
>
> The packets arrive to the real network device, outside of the
> container. Depending on the destination, the packets are forwarded to
> the network device assigned to the container. From this point, the
> path is the same and the packets go through the routes and the sockets
> layer because they are isolated into the container.

You don't need the extra hop. The extra hop is only there because there
are not enough physical interfaces on a machine. Plus I think with a little
work this one particular case can be optimized to the point where it
is not significant.

> - Outgoing traffic
>
> The packets go through the sockets, the routes, the network device
> assigned to the container and finally to the real device.
>
>
> Implementation:
> ---------------
>
> Andrey Savochkin, from OpenVZ team, patchset of this approach uses the
> namespace concept. All the network devices are no longer stored into
> the "dev_base_list" but into a list stored into the network namespace
> structure. Each container has its own network namespace. The network
> device access has been changed to access the network device list
> relative to the current namespace's context instead of the global
> network device list. The same has been made for the routing tables,
> they are all relatives to the namespace and are no longer global
> static. The creation of a new network namespace implies the creation
> of a new set of routing table.
>
> After the creation of a container, no network device exists. It is
> created from outside by the container's parent. The communication
> between the new container and the outside is done via a special pair
> device which have each extremities into each namespace. The MAC
> addresses must be specified and these addresses should be handled by
> the containers developers in order to ensure MAC unicity.
>
> After this network device creation step into each namespace, the
> network configuration is done as usual, in other words, with a new
> operating system initialization or with the 'ifconfig' or 'ip'
> command.
>
> -----   ------   -------   ------   ----
> | LAN |<->| eth0 |<->| veth0 |<-|ns(1)|->| eth0 |<->| IP |
> -----   ------   -------   ------   ----

Note. veth is only necessary because there are enough physical
network interfaces to go around.

> (1) : ns = namespace (aka. Virtual Environment).
>

> The advantages of this implementation is the algorithms used by the
> network stack are not touched, only the network data access is
> modified. That's facilitate the maintenance and the evolution of the
> network code. The drawback is in the case of application container,
> the number of containers can be much more important, (hundred of
> them), that implies a number of network devices more important, a
> longer path to go through the virtualization layer and a more
> resources consumption.

>
> Layer 3 isolation and virtualization
> ----------
>
> The virtualization acts at the IP level. The routes can be isolated
> and the sockets are isolated.
>
> This approach does not bring isolation at the network device
> layer. The isolation and the virtualization is less stronger than the
> layer 2 but it presents a negligible overhead and resource
> consumption near from the non virtualized environment. Furthermore,
> the isolation at the IP level makes the administration very easy.

All administration issues I have seen can be fixed with good tools
and doing things the way the rest of linux does them. Currently
you do things differently.

> - Ingress traffic
>
> The packets arrive to the real device and go through the routes
> engine. From this point, the used route is enough to know to which
> container the traffic can go and the sockets subset assigned to the
> container.

Note this has potentially the highest overhead of them all because
this is the only approach in which it is mandatory to inspect the
network packets to see which container they are in.

My real problem with this approach besides seriously complicating
the administration by not delegating it is that you loose enormous
amounts of power.

> - Outgoing traffic:
>
> The packets go through the sockets, the assigned routes and finally to
> the real device.
>
> The socket are isolated for each container, the current container

> context is used to retrieve the IP address owned by the
> container. When the source address is not specified, the owned IP is
> used to fill the source address of the packet. This is done when doing
> raw, icmp, multicast, broadcast, tcp connection and udp send
> message. If the bind is done on the interface instead of a ip address,
> the source address should be checked to be owned by the container too.
>
>
> Implementation:
> ---------------
>
> Concerning the implementation, several solutions exist. All of them
> rely to the namespace concept but instead of having all the network
> resources relative to the namespace, the namespace pointer is used as
> an identifier.
>
> One of these solutions is the bind filtering. This implementation is
> the simplest to realize but it brings little isolation. If a mobility
> solution must be implemented on the top of that isolation, the bind
> filtering should be coupled with the socket isolation. The bind
> filtering consists in placing several hooks at some strategic points
> into function calls (bind, connect, send datagram, etc ...) in order
> to fill source address and avoid the bind to an IP address outside of
> the container. The container destination should be determined from the
> ingress traffic.
>
> The second solution consists in relying on the route engine to ensure
> the isolation. The routes are all accessible from all the namespaces
> but they contain the information of what namespace they belong. By
> this way, when the traffic is outgoing, only the routes belonging to
> the namespace are used. When the traffic is incoming, it goes through
> a route, because this one has the namespace owner information, the
> traffic can go to the right namespace. The advantage of this approach
> is to have an isolation near of what can provide the layer 2 isolation
> for the IP layer without loss of performances. The drawback is the
> complexity of the code which is strongly linked with the routing
> algorithms and that's do not facilitate the maintenance.
>
>
> -----   ------   -----------
> | LAN |<->| eth0 |<->| ns(1)| IP |
> -----   ------   -----------
>
> (1) : ns = namespace (aka. Virtual Environmenent).
>
> Common points between layer 2 and layer 3 implementations
> --
>

> Because the need of the sockets isolation is the same for the layer 2
> and the layer 3, the socket isolation is the same for the two
> approaches.
>
> The t-uple key, source address, source port, destination address,
> destination port is extended with the network namespace. At the bind
> time, the port usage verification is extended with the network
> namespace pointer too. A port is already in use only if the port and
> network namespace match. If the port match but namespace does not
> match, that means the port is in use but in another namespace.
>
> There can be several listening point on the same port with source
> address set to inaddr_any. When an incoming connection arrives, the
> namespace destination is already resolved and the right connection is
> found without ambiguity.
>
>
> Network resources
> ------------------
>
> L2 : Layer 2
> L3 : Layer 3
> BF : Bind Filtering
>
>
> | L2 | L3 | BF |
> | Sockets | Isolated | Isolated | Isolated(1) |
> | Routes | Isolated | Isolated(2) | X |
> | Inetdev | Virtualized | Virtualized | X |
> | Network devices | Virtualized | X | X |
>
> (1) : The socket should be isolated, in the case of mobility
> (2) : The routes can be isolated or not
>
>
> Network code modifications
> --------------------------
>
> | L2 | L3 | BF |
> | Sockets | struct sock hash tables | idem | idem(1) |

> | | async sock event | | |
> ---
> | Routes | routes table | route cache | X |
> | | | route resolver | |
> ---
> | | | struct ifaddr | |
> | Inetdev | X | add addr | X |
> | | | del addr | |
> | | | gifconf | |
> ---
> | | specific net dev | | |
> | Netdevice | loopback | X | X |
> | | dev list | | |
> ---
>
> (1) if mobility is needed
>
> Solution pros/cons
> ---
>
> ---
> | L2 | L3 | BF |
> ---
> | Isolation | Excellent | Good | Weak |
> ---
> | Virtualization | Total | Partial | None |
> ---
> | Network setup | Complicated | Trivial | Simple |
> ---
> | Overhead | High | Negligible | Negligible |
> ---

So you have two columns that you rate these things that I disagree
with, and you left out what the implications are for code maintenance.

1) Network setup.
   Past a certainly point both bind filtering and Daniel's L3 use a new
   paradigm for managing the network code and become nearly impossible for
   system administrators to understand. The classic one is routing packets
   between machines over the loopback interface by accident. Huh?

The L2. Network setup iss simply the cost of setting up a multiple
machine network. This is more complicated but it is well understood
and well documented today. Plus for the common cases it is easy to
get a tool to automate this for you. When you get a complicated
network this wins hands down because the existing tools work and
you don't have to retrain your sysadmins to understand what is
happening.

2) Runtime Overhead.

Your analysis is confused. Bind/Accept filter is much cheaper than
doing a per packet evaluation in the route cache of which container
it belongs to. Among other things Bind/Accept filtering allows all
of the global variables in the network stack to remain global and
only touches a slow path. So it is both very simple and very cheap.

Next in line comes L2 using real network devices, and Daniel's
L3 thing. Because there are multiple instances of the networking data
structures we have an extra pointer indirection.

Finally we get L2 with an extra network stack traversal, because
we either need the full power of netfilter and traffic shaping
gating access to what a node is doing or we simply don't have
enough real network interfaces. I assert that we can optimize
the lack of network interfaces away by optimizing the drivers
once this becomes an interesting case.

3) Long Term Code Maintenance Overhead.

- A pure L2 implementation. There is a big one time cost of
  changing all of the variable accesses. Once that transition
  is complete things just work. All code is shared so there
  is no real overhead.

- Bind/Connect/Accept filtering. There are so few places in
  the code this is easy to maintain without sharing code with
  everyone else.

- Daniel's L3. A big mass of special purpose code with peculiar
  semantics that no one else in the network stack cares about
  but is right in the middle of the code.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: RE: Network virtualization/isolation
Posted by Leonid Grossman on Sat, 25 Nov 2006 16:35:58 GMT
View Forum Message <> Reply to Message

> -----Original Message-----
> From: netdev-owner@vger.kernel.org
> [mailto:netdev-owner@vger.kernel.org] On Behalf Of Eric W. Biederman

> Then the question is how do we reduce the overhead when we
> don't have enough physical network interfaces to go around.
> My feeling is that we could push the work to the network
> adapters and allow single physical network adapters to
> support multiple network interfaces, each with a different
> link-layer address. At which point the overhead is nearly
> nothing and newer network adapters may start implementing
> enough filtering in hardware to do all of the work for us.

Correct, to a degree.
There will be always a limit on the number of physical "channels" that a NIC
can support, while keeping these channels fully independent and protected at the hw level.
So, you will probably still need to implement the sw path,
with the assumption that some containers (that care about performance) will get a separate
NIC interface and avoid the overhead, and other containers will have to use the sw path.
There are some multi-channel NICs shipping today so it would be possible to see the overhead between the two options (I suspect it will be quite noticeable), but for a general idea about what work could be pushed down to network adapters in the near future you can look at the pcisig.com I/O Virtualization Workgroup.
Once the single root I/O Virtualization spec is completed, it is likely to be supported by several NIC vendors to provide multiple network interfaces on a single NIC that you are looking for.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Sat, 25 Nov 2006 19:26:45 GMT
View Forum Message <> Reply to Message

"Leonid Grossman" <Leonid.Grossman@neterion.com> writes:

>
>
>> -----Original Message-----
>> From: netdev-owner@vger.kernel.org
>> [mailto:netdev-owner@vger.kernel.org] On Behalf Of Eric W. Biederman
>

>> Then the question is how do we reduce the overhead when we
>> don't have enough physical network interfaces to go around.
>> My feeling is that we could push the work to the network
>> adapters and allow single physical network adapters to
>> support multiple network interfaces, each with a different
>> link-layer address. At which point the overhead is nearly
>> nothing and newer network adapters may start implementing
>> enough filtering in hardware to do all of the work for us.
>
> Correct, to a degree.
> There will be always a limit on the number of physical "channels" that a
> NIC
> can support, while keeping these channels fully independent and
> protected at the hw level.
> So, you will probably still need to implement the sw path,
> with the assumption that some containers (that care about performance)
> will get a separate
> NIC interface and avoid the overhead, and other containers will have to
> use the sw path.
> There are some multi-channel NICs shipping today so it would be possible
> to see the overhead between the two options (I suspect it will be quite
> noticeable), but for a general idea about what work could be pushed down
> to network adapters in the near future you can look at the pcisig.com
> I/O Virtualization Workgroup.
> Once the single root I/O Virtualization spec is completed, it is likely
> to be supported by several NIC vendors to provide multiple network
> interfaces on a single NIC that you are looking for.

Pushing it all of the way into the hardware is an optimization, that while
great is likely not necessary. Simply doing a table lookup by
link-level address and selecting between several network interfaces
is enough to ensure we only traverse the network stack once.

To keep overhead down in the container case I don't need the hardware
support to be so good you can do kernel bypass and still trust that
everything is safe. I simply a fast link-level address to container
mapping. We already look at the link-level address on every packet
received so that should not generate any extra cache misses.

In the worst case I might need someone to go as far as the Grand
Unified Lookup to remove all of the overheads. Except for
distributing the work load more evenly across the machine with
separate interrupts and the like I see no need for separate hardware
channels to make things go fast for my needs.

Despite the title of this thread there is no virtualization or
emulation of the hardware involved. Just enhancements to the existing
hardware abstractions.

Eric

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: RE: Network virtualization/isolation
Posted by Leonid Grossman on Sat, 25 Nov 2006 22:17:03 GMT
View Forum Message <> Reply to Message

> -----Original Message-----
> From: Eric W. Biederman [mailto:ebiederm@xmission.com]
> Sent: Saturday, November 25, 2006 11:27 AM
> To: Leonid Grossman
> Cc: hadi@cyberus.ca; Daniel Lezcano; Dmitry Mishin; Stephen
> Hemminger; netdev@vger.kernel.org; Linux Containers
> Subject: Re: Network virtualization/isolation
>
> "Leonid Grossman" <Leonid.Grossman@neterion.com> writes:
>
> > >
> > >
> > >> -----Original Message-----
> > >> From: netdev-owner@vger.kernel.org
> > >> [mailto:netdev-owner@vger.kernel.org] On Behalf Of Eric W.
> Biederman
> > >
> > >> Then the question is how do we reduce the overhead when we
> don't have
> > >> enough physical network interfaces to go around.
> > >> My feeling is that we could push the work to the network
> adapters and
> > >> allow single physical network adapters to support multiple network
> > >> interfaces, each with a different link-layer address. At
> which point
> > >> the overhead is nearly nothing and newer network adapters
> may start
> > >> implementing enough filtering in hardware to do all of the
> work for
> > >> us.
> > >
> > > Correct, to a degree.
> > > There will be always a limit on the number of physical
> "channels" that
> > > a NIC can support, while keeping these channels fully
> independent and

> > protected at the hw level.
> > So, you will probably still need to implement the sw path, with the
> > assumption that some containers (that care about
> performance) will get
> > a separate NIC interface and avoid the overhead, and other
> containers
> > will have to use the sw path.
> > There are some multi-channel NICs shipping today so it would be
> > possible to see the overhead between the two options (I suspect it
> > will be quite noticeable), but for a general idea about what work
> > could be pushed down to network adapters in the near future you can
> > look at the pcisig.com I/O Virtualization Workgroup.
> > Once the single root I/O Virtualization spec is completed, it is
> > likely to be supported by several NIC vendors to provide multiple
> > network interfaces on a single NIC that you are looking for.
>
> Pushing it all of the way into the hardware is an
> optimization, that while great is likely not necessary.
> Simply doing a table lookup by link-level address and
> selecting between several network interfaces is enough to
> ensure we only traverse the network stack once.
>
> To keep overhead down in the container case I don't need the
> hardware support to be so good you can do kernel bypass and
> still trust that everything is safe. I simply a fast
> link-level address to container mapping. We already look at
> the link-level address on every packet received so that
> should not generate any extra cache misses.

I did not mean kernel bypass, just L2 hw channels that for
all practical purposes act as separate NICs -
different MAC addresses, no blocking, independent reset, etc.

>
> In the worst case I might need someone to go as far as the
> Grand Unified Lookup to remove all of the overheads. Except
> for distributing the work load more evenly across the machine
> with separate interrupts and the like I see no need for
> separate hardware channels to make things go fast for my needs.
>
> Despite the title of this thread there is no virtualization
> or emulation of the hardware involved. Just enhancements to
> the existing hardware abstractions.

Right, I was just trying to say that IOV support (likely, from multiple
vendors since
virtualization is expected to be widely used) would provide an option to
export multiple

independent L2 interfaces from a single NIC - even if only a subset of
IOV functionality would be used in this case.

>
> Eric
>

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Sat, 25 Nov 2006 23:16:06 GMT
View Forum Message <> Reply to Message

"Leonid Grossman" <Leonid.Grossman@neterion.com> writes:

> I did not mean kernel bypass, just L2 hw channels that for
> all practical purposes act as separate NICs -
> different MAC addresses, no blocking, independent reset, etc.

Yes. Nearly all of what you need for safe kernel bypass.

>> In the worst case I might need someone to go as far as the
>> Grand Unified Lookup to remove all of the overheads. Except
>> for distributing the work load more evenly across the machine
>> with separate interrupts and the like I see no need for
>> separate hardware channels to make things go fast for my needs.
>>
>> Despite the title of this thread there is no virtualization
>> or emulation of the hardware involved. Just enhancements to
>> the existing hardware abstractions.
>
> Right, I was just trying to say that IOV support (likely, from multiple
> vendors since
> virtualization is expected to be widely used) would provide an option to
> export multiple
> independent L2 interfaces from a single NIC - even if only a subset of
> IOV functionality would be used in this case.

Agreed, and I think I understood that. My basic point was that it
doesn't look to me like I need the hardware support, just that I can
use it when it is there.

The core advantage I see of the multiple queues, is in being able to

split the processing of network traffic and interrupts among multiple
cores.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Sun, 26 Nov 2006 18:34:02 GMT
View Forum Message <> Reply to Message

On Sat, Nov 25, 2006 at 01:21:39AM -0700, Eric W. Biederman wrote:
>
> jamal <hadi@cyberus.ca> writes:
>
> > On Fri, 2006-27-10 at 11:10 +0200, Daniel Lezcano wrote:
> >
> >> No, it uses virtualization at layer 2 and I had already mention it
> >> before (see the first email of the thread), but thank you for the email
> >> thread pointer.
> >
> >
> > What would be really useful is someone takes the time and creates a
> > matrix of the differences between the implementations.
> > It seems there are quiet a few differences but without such comparison
> > (to which all agree to) it is hard to form an opinion without a document
> > of some form.
> >
> > For one, I am puzzled by the arguements about L2 vs L3 - Is this the
> > host side or inside the VE?
> >
> > If it is a discussion of the host side:
> > To me it seems it involves the classification of some packet header
> > arriving on a physical netdevice on the host side (irrelevant whether
> > they are L2 or L7) and reaching a decision to select some redirected to
> > virtual netdevice.
>
> There are two techniques in real use.
> - Bind/Accept filtering
>
> Which layer 3 addresses a socket can bind/accept are filtered,
> but otherwise the network stack remains unchanged. When your
> container/VE only has a single IP address this works great.

> When you get multiple IPs this technique starts to fall down because

> it is not obvious how to make this correctly handle wild card ip
> addresses.

not really, you have to check for a (sub)set of IPs
that's quite simple and not hard to get right, I agree
that it increases the overhead on those checks, but
this only occurs at bind/connect/accept time ...

> This technique also falls down because it is very hard to
> support raw IP packets.

raw ip packets could be header checked in the same way
but in general, this type of isolation does not make
too much sense for raw ip purposes ...

> The major advantage of this approach is that it is insanely simple
> and cheap.

and don't forget flexible, because it also alows a few
things not quite easily done with layer2 techniques
(see below)

> When the discussion started this is what I called Layer 3, bind
> filtering is probably a more precise name.
>
> - Network object tagging
>
> Every network device, each socket, each routing table, each net
> filter table, everything but the packets themselves is associated
> with a single VE/container. In principle the network stack doesn't
> change except everything that currently access global variables gets
> an additional pointer indirection.
>
> To find where a packet is you must look at it's network device on
> ingress, and you must look at it's socket on egress.
>
> This allows capabilities like CAP_NET_ADMIN to be fairly safely
> given to people inside a container without problems.
>
> There are two basic concerns here.
> 1) This is a lot of code that needs to be touched.
> 2) There are not enough physical network devices to go around so
> we need something that maps packets coming in a physical network
> device into multiple virtual network devices.
>
> The obvious way to do this mapping is with either ethernet
> bridging or with the linux routing code if the external network
> is not ethernet, and some tunnel devices between the VE and the

> host environment. This allows firewalling and in general the
> full power of the linux network stack.
>
> The concern is that the extra trip through the network stack adds
> overhead.
>
> This is the technique we have been calling layer two because it
> works below the IP layer and as such should work for everything.
>
> There have been some intermediate solutions considered but generally
> the have the down sides of additional expense without the upsides of
> more power.
>
> > The admin (on the host) decides what packets any VE can see.
> > Once within the VE, standard Linux net stack applies. The same applies
> > on the egress. The admin decides what packets emanating from the VE
> > go where.
> > I dont think this is a simple L2 vs L3. You need to be able to process
> > IP as well as Decnet[1]
>
> Except for some implementation details I don't think we have
> disagreement about what we are talking about although there is
> certainly a little confusion. The true issue is can we implement
> something that places the full power of the network stack (including
> things like creating virtual tunnel devices) into the container/VE
> without sacrificing performance.
>
> I think we could all agree on the most general technique if we could
> convince ourselves the overhead was unmeasurable.

and we do not give away features like:

- sharing an IP between different guests, so that
  services can bind to the same IP as long as they
  do not collide

- allow simple wrapping (ala chroot()) for processes
  without requiring a complete routing/network setup
  plus a virtual switch/router/whatever

> Given that performance is the primary concern this is something a
> network stack expert might be able to help with. My gut feel is
> the extra pointer indirection for the more general technique is
> negligible and will not affect the network performance. The network
> stack can be very sensitive to additional cache misses so I could be
> wrong. Opinions?

well, here we are talking about layer2 _isolation_

if I got that right, i.e. you split the physical
interfaces up into separate network namespaces, which
then can make full use of the assigned interfaces

this is something which I'm perfectly fine with, as
I do not think it adds significant overhead (nevertheless
it needs some testing) but at the same time, this is
something which isn't very useful in the generic case,
where folks will have, let's say two network interfaces
and want to share one of them between 100 guests ...

> Then the question is how do we reduce the overhead when we don't have
> enough physical network interfaces to go around. My feeling is that
> we could push the work to the network adapters and allow single
> physical network adapters to support multiple network interfaces, each
> with a different link-layer address.

that would be something interesting, but again, the
number of nics allowing for an arbitrary number of
filters, which also can be identified/correlated to
the network context without adding even more overhead
is probably insignificant ... so IMHO that would:

 - keep all interfaces in promisc mode
 - check each packet for the set of MACs

as the checks would require to identify the interface,
that would immediately result in O(N) overhead for
each packet received, plus the overhead added by
disabling the hardware filters ... but maybe that
changed over the years, I'm definitely no network
stack/device expert ...

> At which point the overhead is nearly nothing and newer network
> adapters may start implementing enough filtering in hardware to do all
> of the work for us.

well, might be a solution in 4-5 years, when basically
any computer system uses such nics ...

> > [1] Since Linux has the only SMP-capable, firewall-capable Decnet
> > implementation - wouldnt it be fun to have it be virtualized as
> > well? ;->
>
> Yes. The only problem I have seen with Decnet is the pain of teaching
> it that it's variables aren't global anymore. Not a show stopper
> but something that keeps Decnet out of existing proof of concept
> implications.

>
> Eric
>
> p.s. Sorry for the long delayed reply. I have had my head down
> stabilizing 2.6.19 and netdev is not a list I regularly watch.
>
> p.p.s. I have CC'd the containers list so we catch all of the
> relevant people on the discussion. I believe this is an open
> list so shouldn't cause any problems.

thanks a lot!

best,
Herbert

> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Ben Greear on Sun, 26 Nov 2006 19:41:50 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Sat, Nov 25, 2006 at 01:21:39AM -0700, Eric W. Biederman wrote:
>
> Then the question is how do we reduce the overhead when we don't have
>> enough physical network interfaces to go around. My feeling is that
>> we could push the work to the network adapters and allow single
>> physical network adapters to support multiple network interfaces, each
>> with a different link-layer address.
>>
>
> that would be something interesting, but again, the
> number of nics allowing for an arbitrary number of
> filters, which also can be identified/correlated to
> the network context without adding even more overhead
> is probably insignificant ... so IMHO that would:
>
> - keep all interfaces in promisc mode
> - check each packet for the set of MACs
>

> as the checks would require to identify the interface,
> that would immediately result in O(N) overhead for
> each packet received, plus the overhead added by
> disabling the hardware filters ... but maybe that
> changed over the years, I'm definitely no network
> stack/device expert ...
>
This can be implemented similar to how MAC-VLANs are currently done (in my out-of-tree patch).
There is a performance hit with lots of virtual interfaces (maybe 10% in some cases), but this is still
greater than 500Mbps full-duplex on 2 ports on a modern dual-core machine.

I don't even have hashing implemented, but it could be easily added and that should
significantly decrease the search time from O(n) to something approaching O(1)
in the normal case.

This should also be an easy feature for NICs to add, and just as with 802.1Q VLANs, when
hardware support is available, the features can migrate into the NIC, with the software
mac-vlan logic handling generic hardware.

In a switched environment, going into PROMISC mode should not add any significant overhead..

Ben

--
Ben Greear <greearb@candelatech.com>
Candela Technologies Inc http://www.candelatech.com

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Sun, 26 Nov 2006 20:52:14 GMT
View Forum Message <> Reply to Message

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

> On Sat, Nov 25, 2006 at 01:21:39AM -0700, Eric W. Biederman wrote:

>> There are two techniques in real use.
>> - Bind/Accept filtering
>>
>> Which layer 3 addresses a socket can bind/accept are filtered,
>> but otherwise the network stack remains unchanged. When your
>> container/VE only has a single IP address this works great.
>
>> When you get multiple IPs this technique starts to fall down because
>> it is not obvious how to make this correctly handle wild card ip
>> addresses.
>
> not really, you have to check for a (sub)set of IPs
> that's quite simple and not hard to get right, I agree
> that it increases the overhead on those checks, but
> this only occurs at bind/connect/accept time ...

The general problem is you get into mental model problems. You think
you are isolated but you don't realize you can route packets over the
loopback interface for example. But with care yes you can solve it.

However while I think there is value in this technique it doesn't
solve any of my problems, nor do I think it can be easily stretched
to solve my problems. My gut feel for implementation still says
this should be a new netfilter table that filters binds and accepts
if we implement this.

For most of us we need more power than we can get with the simple
bind/accept filtering so we I think the network namespace work should
concentrate on the general technique that gives us the entire power of
the current network stack. At least until we have proved the
overheads are unacceptable.

>> Given that performance is the primary concern this is something a
>> network stack expert might be able to help with. My gut feel is
>> the extra pointer indirection for the more general technique is
>> negligible and will not affect the network performance. The network
>> stack can be very sensitive to additional cache misses so I could be
>> wrong. Opinions?
>
> well, here we are talking about layer2 _isolation_
> if I got that right, i.e. you split the physical
> interfaces up into separate network namespaces, which
> then can make full use of the assigned interfaces

Yes. Layer 2 isolation is a good description.

> this is something which I'm perfectly fine with, as

> I do not think it adds significant overhead (nevertheless
> it needs some testing)
Yes lots of testing and careful implementation.

> but at the same time, this is
> something which isn't very useful in the generic case,
> where folks will have, let's say two network interfaces
> and want to share one of them between 100 guests ...

It is useful in the generic case. It just requires being smart to
keep the overheads down.

> as the checks would require to identify the interface,
> that would immediately result in O(N) overhead for
> each packet received, plus the overhead added by
> disabling the hardware filters ... but maybe that
> changed over the years, I'm definitely no network
> stack/device expert ...

Getting this to O(log(N)) is easy, and you can probably
get the average case to O(1) without trying too hard. This
is no worse than routing tables or multiple IP addresses on
a single interface. Ben Greear has addressed this. His experience
suggest that even O(N) is not likely to be a significant problem.

Now I'm going to go bury my head in the sand for a bit. The hard
problems are not how do we reshape the network stack but how do we
get the appropriate context into all of our user space interfaces.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Daniel Lezcano on Tue, 28 Nov 2006 14:15:26 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:

[snip]

>>
>> The packets arrive to the real device and go through the routes
>> engine. From this point, the used route is enough to know to which

>> container the traffic can go and the sockets subset assigned to the
>> container.
>
> Note this has potentially the highest overhead of them all because
> this is the only approach in which it is mandatory to inspect the
> network packets to see which container they are in.

If the container is in the route information, when you use the route,
you have the container destination with it. I don't see the overhead here.

>
> My real problem with this approach besides seriously complicating
> the administration by not delegating it is that you loose enormous
> amounts of power.

I don't understand why you say administration is more complicated.
unshare -> ifconfig

1 container = 1 IP

[snip]

> So you have two columns that you rate these things that I disagree
> with, and you left out what the implications are for code maintenance.
>
> 1) Network setup.
> Past a certainly point both bind filtering and Daniel's L3 use a new
> paradigm for managing the network code and become nearly impossible for
> system administrators to understand. The classic one is routing packets
> between machines over the loopback interface by accident. Huh?

What is this new paradigm you are talking about ?

>
> The L2. Network setup iss simply the cost of setting up a multiple
> machine network. This is more complicated but it is well understood
> and well documented today. Plus for the common cases it is easy to
> get a tool to automate this for you. When you get a complicated
> network this wins hands down because the existing tools work and
> you don't have to retrain your sysadmins to understand what is
> happening.

unshare -> (guest) add mac address
        (host) add mac address
   (guest) set ip address
        (host) set ip address
        (host) setup bridge

1 container = 2 net devices (root + guest), 2 IPs, 2 mac addresses, 1
bridge.
100 containers = 200 net devices, 200 IPs, 200 mac addresses, 1 bridge.

>
> 2) Runtime Overhead.
>
> Your analysis is confused. Bind/Accept filter is much cheaper than
> doing a per packet evaluation in the route cache of which container
> it belongs to. Among other things Bind/Accept filtering allows all
> of the global variables in the network stack to remain global and
> only touches a slow path. So it is both very simple and very cheap.
>
> Next in line comes L2 using real network devices, and Daniel's
> L3 thing. Because there are multiple instances of the networking data
> structures we have an extra pointer indirection.

There is not extra networking data structure instantiation in the
Daniel's L3.
>
> Finally we get L2 with an extra network stack traversal, because
> we either need the full power of netfilter and traffic shaping
> gating access to what a node is doing or we simply don't have
> enough real network interfaces. I assert that we can optimize
> the lack of network interfaces away by optimizing the drivers
> once this becomes an interesting case.
>
> 3) Long Term Code Maintenance Overhead.
>
> - A pure L2 implementation. There is a big one time cost of
>   changing all of the variable accesses. Once that transition
>   is complete things just work. All code is shared so there
>   is no real overhead.
>
> - Bind/Connect/Accept filtering. There are so few places in
>   the code this is easy to maintain without sharing code with
>   everyone else.

For isolation too ? Can we build network migration on top of that ?

>
> - Daniel's L3. A big mass of special purpose code with peculiar
>   semantics that no one else in the network stack cares about
>   but is right in the middle of the code.

Thanks Eric for all your comments.

  -- Daniel

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Tue, 28 Nov 2006 16:51:57 GMT
View Forum Message <> Reply to Message

I do not want to get into a big debate on the merits of various
techniques at this time. We seem to be in basic agreement
about what we are talking about.

There is one thing I think we can all agree upon.
- Everything except isolation at the network device/L2 layer, does not
  allow guests to have the full power of the linux networking stack.

- There has been a demonstrated use for the full power of the linux
  networking stack in containers..

- There are a set of techniques which look as though they will give
  us full speed when we do isolation of the network stack at the
  network device/L2 layer.

Is there any reason why we don't want to implement network namespaces
without the full power of the linux network stack?

If there is a case where we clearly don't want the full power of the
linux network stack in a guest but we still need a namespace we can
start looking at the merits of the alternatives.

> What is this new paradigm you are talking about ?

The basic point is this. The less like stock linux the inside of a
container looks, and the more of a special case it is the more
confusing it is. The classic example is that for a system container
routing packets between containers over the loopback interface is
completely unexpected.

> There is not extra networking data structure instantiation in the
> Daniel's L3.
Nope just an extra field which serves the same purpose.

>> - Bind/Connect/Accept filtering. There are so few places in
>>   the code this is easy to maintain without sharing code with
>>   everyone else.

>
> For isolation too ? Can we build network migration on top of that ?

As long as you can take your globally visible network address with you
when you migrate you can build network migration on top of it. So yes
bind/accept filtering is sufficient to implement migration, if you are
only using IP based protocols.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Tue, 28 Nov 2006 17:37:19 GMT
View Forum Message <> Reply to Message

On Tue, Nov 28, 2006 at 09:51:57AM -0700, Eric W. Biederman wrote:
>
> I do not want to get into a big debate on the merits of various
> techniques at this time. We seem to be in basic agreement
> about what we are talking about.
>
> There is one thing I think we can all agree upon.
> - Everything except isolation at the network device/L2 layer, does not
>   allow guests to have the full power of the linux networking stack.
>
> - There has been a demonstrated use for the full power of the linux
>   networking stack in containers..

 - There has been a demonstrated use for the full performance
   IP layer isolation too, both in BSD and Linux for several
   years now ...

> - There are a set of techniques which look as though they will give
>   us full speed when we do isolation of the network stack at the
>   network device/L2 layer.
>
> Is there any reason why we don't want to implement network namespaces
> without the full power of the linux network stack?

duplicate negation ->

 "Is there any reason why we _want_ to implement network namespaces
  with the full power of the linux network stack?"

yes, I think you have some reasons for doing so, especially
the migration part seems to depend on it

OTOH, we _also_ want IP isolation, as it allows to separate
services (and even handle overlapping sets) in a very natural
(linux) way, without adding interfaces and virtual switches
and bridges at a potentially high overhead just to do simple
layer 3 isolation

> If there is a case where we clearly don't want the full power of the
> linux network stack in a guest but we still need a namespace we can
> start looking at the merits of the alternatives.

see above, of course, all cases can be 'simulated' by a
fully blown layer 2 virtualization, so that's not an argument
but OTOH, all this can also be achieved with Xen, so we
could as well bring the argument, why have network namespaces
at all, if you can get the same functionality (including the
migration) with a Xen domU ...

> > What is this new paradigm you are talking about ?
>
> The basic point is this. The less like stock linux the inside of a
> container looks, and the more of a special case it is the more
> confusing it is. The classic example is that for a system container
> routing packets between containers over the loopback interface is
> completely unexpected.

I disagree here, from the point of isolation that would be
the same as saying:

  "having a chroot(), it is completely unexpected that
   the files reside on the same filesystem and even will
   be cached in the same inode cache"

the thing is, once you depart from the 'container' = 'box'
idea, and accept that certain resources are shared (btw,
one of the major benefits of 'containers' over things like
Xen or UML) you can easily accept that:

 - host local traffic uses loopback
 - non local traffic uses the appropriate interfaces
 - guests _are_ local on the host, so
 - guest - guest and guest - host traffic _is_ local
   an therefore will be more performant than remote
   traffic (unless you add various virtual switches and
   bridges and stacks to the pathes)

> > There is not extra networking data structure instantiation in the
> > Daniel's L3.
> Nope just an extra field which serves the same purpose.
>
> >> - Bind/Connect/Accept filtering. There are so few places in
> >>   the code this is easy to maintain without sharing code with
> >>   everyone else.
> >
> > For isolation too ? Can we build network migration on top of that ?
>
> As long as you can take your globally visible network address
> with you when you migrate you can build network migration on
> top of it. So yes bind/accept filtering is sufficient to
> implement migration, if you are only using IP based protocols.

correct, don't get me wrong, I'm absolutely not against
layer 2 virtualization, but not at the expense of light-
weight layer 3 isolation, which _is_ the traditional way
'containers' are built (see BSD, solaris ...)

HTC,
Herbert

> Eric
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Daniel Lezcano on Tue, 28 Nov 2006 20:26:52 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
 > I do not want to get into a big debate on the merits of various
 > techniques at this time. We seem to be in basic agreement
 > about what we are talking about.
 >
 > There is one thing I think we can all agree upon.
 > - Everything except isolation at the network device/L2 layer, does not
 >   allow guests to have the full power of the linux networking stack.
Agree.
 >

> - There has been a demonstrated use for the full power of the linux
>   networking stack in containers..
Agree.
>
> - There are a set of techniques which look as though they will give
>   us full speed when we do isolation of the network stack at the
>   network device/L2 layer.
Agree.

> Is there any reason why we don't want to implement network namespaces
> without the full power of the linux network stack?
Don't make me wrong, I never said layer 2 should not be used. I am only arguing a layer 3 should use the mechanism provided by the layer 2 and use a subset of it like the sockets virtualization/isolation.

Just IP isolation for lightweight containers, applications containers in order to have mobility.

> If there is a case where we clearly don't want the full power of the
> linux network stack in a guest but we still need a namespace we can
> start looking at the merits of the alternatives.
Dmitry and I, we are looking for a l3 based on a subset of the l2 and according with Herbert needs.
If we can provide a l3 isolation based on the l2 which:
 - does not collide with l2
 - fit the needs of Herbert
 - allows the migration
 - use common code between l2 and l3
Should it not be sufficient to justify to have a l3 with the l2 isolation ?

>> What is this new paradigm you are talking about ?
>
> The basic point is this. The less like stock linux the inside of a
> container looks, and the more of a special case it is the more
> confusing it is. The classic example is that for a system container
> routing packets between containers over the loopback interface is
> completely unexpected.

Right for system container, but not necessary for application containers.

>
>> There is not extra networking data structure instantiation in the
>> Daniel's L3.
> Nope just an extra field which serves the same purpose.
>
>>> - Bind/Connect/Accept filtering. There are so few places in
>>>   the code this is easy to maintain without sharing code with
>>>   everyone else.

>> For isolation too ? Can we build network migration on top of that ?

> As long as you can take your globally visible network address with you
> when you migrate you can build network migration on top of it. So yes
> bind/accept filtering is sufficient to implement migration, if you are
> only using IP based protocols.

When you migrate an application, you must cleanup related sockets on the source machine. The cleanup can not rely on the IP addresses because you will be not able to discriminate all the sockets related to the container. Another stuff is the network objects life-cycle, the container will die when the application will finish, the timewait sockets will stay until all data are flushed to peer. You can not restart a new container with the same IP address, so you need to monitor the socket before relaunching a new container or unmounting the aliased interface associated with the container. You need a ref counting for the container and this refcount is exactly what has the network namespace. Another example, you can not have several application binding to INADDR_ANY:port without conflict. The multiport instantiation is exactly what brings the sockets isolation/virtualization with the l2/l3.

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Tue, 28 Nov 2006 21:50:03 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <dlezcano@fr.ibm.com> writes:

> Eric W. Biederman wrote:
>> I do not want to get into a big debate on the merits of various
>> techniques at this time. We seem to be in basic agreement
>> about what we are talking about.
>>
>> There is one thing I think we can all agree upon.
>> - Everything except isolation at the network device/L2 layer, does not
>>   allow guests to have the full power of the linux networking stack.
> Agree.
>>
>> - There has been a demonstrated use for the full power of the linux
>>   networking stack in containers..
> Agree.
>>

>> - There are a set of techniques which look as though they will give
>>   us full speed when we do isolation of the network stack at the
>>   network device/L2 layer.
> Agree.

Herbert Poetzl <herbert@13thfloor.at> writes:
> correct, don't get me wrong, I'm absolutely not against
> layer 2 virtualization, but not at the expense of light-
> weight layer 3 isolation, which _is_ the traditional way
> 'containers' are built (see BSD, solaris ...)

Ok. So on this point we agree. Full isolation at the network device/L2 level
is desirable and no one is opposed to that.

There is however a strong feeling especially for the case of application
containers that something more focused on what a non-privileged process can
use and deal with would be nice. The ``L3'' case.

I agree that has potential but I worry about 2 things.
- Premature optimization.
- A poor choice of semantics.
- Feature creep leading to insane semantics.

I feel there is something in the L3 arguments as well and it sounds
like it would be a good idea to flush out the semantics.

For full network isolation we have the case that every process,
every socket, and every network device belongs to a network namespace.
This is enough to derive the network namespace for all other user
visible data structures, and to a large extent to define their semantics.

We still need a definition of the non-privileged case, that is compatible
with the former definition.

.....

What unprivileged user space gets to manipulate are sockets. So perhaps
we can break our model into a network socket namespace and network device
namespace.

I would define it so that for each socket there is exactly one network
socket namespace. And for each network socket namespace there is exactly
one network device namespace.

The network socket namespace would be concerned with the rules for deciding
which local addresses a socket can connect/accept/bind to.

The network device namespace would be concerned with everything else.

The problem I see are the wild card binds. In general unmodified
server applications want to bind to *:port by default. Running
two such applications on different ip addresses is a problem. Even
if you can configure them not to do that it becomes easy to do that
be default.

There are some interesting flexible cases where we want one
application container to have one port on IP, and a different
application container to have a different port on the same IP.

So we need something flexible and not just based on IP addresses.
I think the right answer here is a netfilter table that defines
what we can accept/bind/connect the socket to.

The tricky part is when do we return -EADDRINUSE.

I think we can specify the rules such that if we conflict with
another socket in the same socket namespace the rules remain
as they are today, and the kernel returns it unconditionally.

I think for cases across network socket namespaces it should
be a matter for the rules, to decide if the connection should
happen and what error code to return if the connection does not
happen.

There is a potential in this to have an ambiguous case where two
applications can be listening for connections on the same socket
on the same port and both will allow the connection. If that
is the case I believe the proper definition is the first socket
that we find that will accept the connection gets the connection.

I believe this is a sufficiently general definition that we can
make it work with network types in the kernel including DECNET, IP,
and IPv6.

The only gain I see for having the socket namespace is socket
collision detection, and a convenient tag to distinguish containers.

I think this set of netfilter rules may be an interesting alternative
to ip connection tracking in the current firewall code.

...

Assuming the above scheme works does that sound about what people
actually want to use?

I think with the appropriate set of rules it provides what is needed
for application migration. I.e. 127.0.0.1 can be filtered so that
you can only connect to sockets in your current container.

It does get a little odd because it does allow for the possibility
that you can have multiple connected sockets with same source ip,
source port, destination ip, destination port. If the rules are
setup appropriately. I don't see that peculiarity being visible on
the outside network so it shouldn't be a problem.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Wed, 29 Nov 2006 05:54:32 GMT
View Forum Message <> Reply to Message

On Tue, Nov 28, 2006 at 02:50:03PM -0700, Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
> > Eric W. Biederman wrote:
> >> I do not want to get into a big debate on the merits of various
> >> techniques at this time. We seem to be in basic agreement
> >> about what we are talking about.
> >>
> >> There is one thing I think we can all agree upon.
> >> - Everything except isolation at the network device/L2 layer, does not
> >>   allow guests to have the full power of the linux networking stack.
> > Agree.
> >>
> >> - There has been a demonstrated use for the full power of the linux
> >>   networking stack in containers..
> > Agree.
> >>
> >> - There are a set of techniques which look as though they will give
> >>   us full speed when we do isolation of the network stack at the
> >>   network device/L2 layer.
> > Agree.
>
> Herbert Poetzl <herbert@13thfloor.at> writes:
> > correct, don't get me wrong, I'm absolutely not against
> > layer 2 virtualization, but not at the expense of light-
> > weight layer 3 isolation, which _is_ the traditional way
> > 'containers' are built (see BSD, solaris ...)

> Ok. So on this point we agree. Full isolation at the network device/L2
> level is desirable and no one is opposed to that.

> There is however a strong feeling especially for the case of
> application containers that something more focused on what a
> non-privileged process can use and deal with would be nice.
> The ``L3'' case.

> I agree that has potential but I worry about 2 things.
> - Premature optimization.
> - A poor choice of semantics.
> - Feature creep leading to insane semantics.

> I feel there is something in the L3 arguments as well and it sounds
> like it would be a good idea to flush out the semantics.

> For full network isolation we have the case that every process,
> every socket, and every network device belongs to a network namespace.

> This is enough to derive the network namespace for all other user
> visible data structures, and to a large extent to define their
> semantics.

> We still need a definition of the non-privileged case, that is
> compatible with the former definition.

yep, sounds interesting ...

> .....
>
> What unprivileged user space gets to manipulate are sockets.
> So perhaps we can break our model into a network socket namespace
> and network device namespace.

> I would define it so that for each socket there is exactly one
> network socket namespace. And for each network socket namespace
> there is exactly one network device namespace.

> The network socket namespace would be concerned with the rules for
> deciding which local addresses a socket can connect/accept/bind to.

> The network device namespace would be concerned with everything else.

hmm, guess I've read the word 'semantics' so many times
now, and always in conjunction with insane and unexpected,
so I think it can't hurt to explain the semantics behind
what we currently use once again, maybe I'm missing something

first, what we currently do:

- a network context consists of a bunch of flags, and
  a set of ip addresses

- a process is either part of exactly one such context
  or unrestricted

- at bind() time, collisions are checked (in the * case)
  and addresses are verified against the assigned set

- at lookup() time, addresses are checked against the
  assigned set (again in the * case)

- for queries, addresses are checked against the set, and
  if the address is found, the corresponding device will
  be visible (basically per address)

- for guest originating traffic, the src address will
  be picked from the set, where the first assigned IP
  is handled special as 'last resort' if no better one
  can be found

here now the semantics:

- bind() can be done for all IP/port pairs which do not
  conflict with existing sockets
  [identical to the current behaviour]

- bind() to * is handled like a bind() for each address
  in the assigned set of IPs (if one fails, then the
  entire bind will fail)
  [identical behaviour, subset]

- lookup() will only match sockets which match the address
  where * now means any IP from the IP set
  [identical behaviour, subset]

- the source address has to reside within the IP set for
  outgoing traffic

- netinfo/proc are filtered according to the rule
  address in set -> show address/interface

except for the last one, the behaviour is identical to the
current linux networking behaviour. the hiding of unavailable
interfaces/addresses is a virtualization mechanism we use to

make it look like a separate box, which is sometimes necessary
for certain applications and humans :)

> The problem I see are the wild card binds. In general unmodified
> server applications want to bind to *:port by default. Running
> two such applications on different ip addresses is a problem. Even
> if you can configure them not to do that it becomes easy to do that
> be default.

those are working and running perfectly fine, the only
time you have to take care of such applications is when you
start them outside any isolation container

> There are some interesting flexible cases where we want one
> application container to have one port on IP, and a different
> application container to have a different port on the same IP.

> So we need something flexible and not just based on IP addresses.
> I think the right answer here is a netfilter table that defines
> what we can accept/bind/connect the socket to.

I'm fine with such an approach, given that this can be used
to get reasonably similar semantics as above wihtout jumping
thorugh hoops

> The tricky part is when do we return -EADDRINUSE.

IMHO not at all, see the 'simple' semantics above

> I think we can specify the rules such that if we conflict with
> another socket in the same socket namespace the rules remain
> as they are today, and the kernel returns it unconditionally.
>
> I think for cases across network socket namespaces it should
> be a matter for the rules, to decide if the connection should
> happen and what error code to return if the connection does not
> happen.
>
> There is a potential in this to have an ambiguous case where two
> applications can be listening for connections on the same socket
> on the same port and both will allow the connection. If that
> is the case I believe the proper definition is the first socket
> that we find that will accept the connection gets the connection.

that is what I call 'unexpected behaviour' ....

> I believe this is a sufficiently general definition that we can
> make it work with network types in the kernel including DECNET,

> IP, and IPv6.

no idea about decnet, but for IP and IPv6 the beforementioned
semantics work quite fine ...

> The only gain I see for having the socket namespace is socket
> collision detection, and a convenient tag to distinguish containers.

well, you would need the tag anyway IMHO, otherwise I don't
see a way to map the netfilter chains/rules to the 'guests'
or am I missing something here?

> I think this set of netfilter rules may be an interesting alternative
> to ip connection tracking in the current firewall code.
>
> ...
>
> Assuming the above scheme works does that sound about what people
> actually want to use?

from my PoV, folks want to use chbind() to 'jail' a group
of processes (or a single process) to a subset of IP
addresses ...

> I think with the appropriate set of rules it provides what is needed
> for application migration. I.e. 127.0.0.1 can be filtered so that
> you can only connect to sockets in your current container.
>
> It does get a little odd because it does allow for the possibility
> that you can have multiple connected sockets with same source ip,
> source port, destination ip, destination port. If the rules are
> setup appropriately. I don't see that peculiarity being visible on
> the outside network so it shouldn't be a problem.

to the outside it looks perfectly normal, as it does on
the inside ... just the host has the 'full picture'

best,
Herbert

> Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Wed, 29 Nov 2006 05:58:57 GMT
View Forum Message <> Reply to Message

On Tue, Nov 28, 2006 at 09:26:52PM +0100, Daniel Lezcano wrote:
> Eric W. Biederman wrote:
> > I do not want to get into a big debate on the merits of various
> > techniques at this time. We seem to be in basic agreement
> > about what we are talking about.
> >
> > There is one thing I think we can all agree upon.
> > - Everything except isolation at the network device/L2 layer, does not
> > allow guests to have the full power of the linux networking stack.
> Agree.
> >
> > - There has been a demonstrated use for the full power of the linux
> > networking stack in containers..
> Agree.
> >
> > - There are a set of techniques which look as though they will give
> > us full speed when we do isolation of the network stack at the
> > network device/L2 layer.
> Agree.
>
> > Is there any reason why we don't want to implement network namespaces
> > without the full power of the linux network stack?
> Don't make me wrong, I never said layer 2 should not be used. I am only
> arguing a layer 3 should use the mechanism provided by the layer 2 and
> use a subset of it like the sockets virtualization/isolation.
>
> Just IP isolation for lightweight containers, applications containers in
> order to have mobility.
>
> > If there is a case where we clearly don't want the full power of the
> > linux network stack in a guest but we still need a namespace we can
> > start looking at the merits of the alternatives.
> Dmitry and I, we are looking for a l3 based on a subset of the l2 and
> according with Herbert needs.
> If we can provide a l3 isolation based on the l2 which:
> - does not collide with l2
> - fit the needs of Herbert
> - allows the migration
> - use common code between l2 and l3
> Should it not be sufficient to justify to have a l3 with the l2
> isolation?

sounds good to me ...

> >> What is this new paradigm you are talking about ?

> >
> > The basic point is this. The less like stock linux the inside of a
> > container looks, and the more of a special case it is the more
> > confusing it is. The classic example is that for a system container
> > routing packets between containers over the loopback interface is
> > completely unexpected.
>
> Right for system container, but not necessary for application containers.

yep

best,
Herbert

> >> There is not extra networking data structure instantiation in the
> >> Daniel's L3.
> > Nope just an extra field which serves the same purpose.
> >
> >>> - Bind/Connect/Accept filtering. There are so few places in
> >>> the code this is easy to maintain without sharing code with
> >>> everyone else.
> >> For isolation too ? Can we build network migration on top of that ?
>
> > As long as you can take your globally visible network address with you
> > when you migrate you can build network migration on top of it. So yes
> > bind/accept filtering is sufficient to implement migration, if you are
> > only using IP based protocols.
>
> When you migrate an application, you must cleanup related sockets on the
> source machine. The cleanup can not rely on the IP addresses because you
> will be not able to discriminate all the sockets related to the
> container. Another stuff is the network objects life-cycle, the
> container will die when the application will finish, the timewait
> sockets will stay until all data are flushed to peer. You can not
> restart a new container with the same IP address, so you need to monitor
> the socket before relaunching a new container or unmounting the aliased
> interface associated with the container. You need a ref counting for the
> container and this refcount is exactly what has the network namespace.
> Another example, you can not have several application binding to
> INADDR_ANY:port without conflict. The multiport instantiation is exactly
> what brings the sockets isolation/virtualization with the l2/l3.
>
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list

Subject: Re: Network virtualization/isolation
Posted by Brian Haley on Wed, 29 Nov 2006 20:21:00 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> I think for cases across network socket namespaces it should
> be a matter for the rules, to decide if the connection should
> happen and what error code to return if the connection does not
> happen.
>
> There is a potential in this to have an ambiguous case where two
> applications can be listening for connections on the same socket
> on the same port and both will allow the connection. If that
> is the case I believe the proper definition is the first socket
> that we find that will accept the connection gets the connection.

Wouldn't you want to catch this at bind() and/or configuration time and fail? Having overlapping namespaces/rules seems undesirable, since as Herbert said, can get you "unexpected behaviour".

> I think with the appropriate set of rules it provides what is needed
> for application migration. I.e. 127.0.0.1 can be filtered so that
> you can only connect to sockets in your current container.
>
> It does get a little odd because it does allow for the possibility
> that you can have multiple connected sockets with same source ip,
> source port, destination ip, destination port. If the rules are
> setup appropriately. I don't see that peculiarity being visible on
> the outside network so it shouldn't be a problem.

So if they're using the same protocol (eg TCP), how is it decided which one gets an incoming packet? Maybe I'm missing something as I don't understand your inside/outside network reference - is that to the loopback address comment in the previous paragraph?

Thanks,

-Brian

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Daniel Lezcano on Wed, 29 Nov 2006 22:10:39 GMT
View Forum Message <> Reply to Message

Brian Haley wrote:
> Eric W. Biederman wrote:
>> I think for cases across network socket namespaces it should
>> be a matter for the rules, to decide if the connection should
>> happen and what error code to return if the connection does not
>> happen.
>>
>> There is a potential in this to have an ambiguous case where two
>> applications can be listening for connections on the same socket
>> on the same port and both will allow the connection. If that
>> is the case I believe the proper definition is the first socket
>> that we find that will accept the connection gets the connection.
No. If you try to connect, the destination IP address is assigned to a
network namespace. This network namespace is used to leave the listening
socket ambiguity.
>
> Wouldn't you want to catch this at bind() and/or configuration time and
> fail? Having overlapping namespaces/rules seems undesirable, since as
> Herbert said, can get you "unexpected behaviour".

Overlapping is not a problem, you can have several sockets binded on the
same INADDR_ANY/port without ambiguity because the network namespace
pointer is added as a new key for sockets lookup, (src addr, src port,
dst addr, dst port, net ns pointer). The bind should not be forced to a
specific address because you will not be able to connect via 127.0.0.1.

>
>> I think with the appropriate set of rules it provides what is needed
>> for application migration. I.e. 127.0.0.1 can be filtered so that
>> you can only connect to sockets in your current container.
>>
>> It does get a little odd because it does allow for the possibility
>> that you can have multiple connected sockets with same source ip,
>> source port, destination ip, destination port. If the rules are
>> setup appropriately. I don't see that peculiarity being visible on
>> the outside network so it shouldn't be a problem.
>
> So if they're using the same protocol (eg TCP), how is it decided which
> one gets an incoming packet? Maybe I'm missing something as I don't
> understand your inside/outside network reference - is that to the
> loopback address comment in the previous paragraph?

The sockets for l3 isolation are isolated like the l2 (this is common
code). The difference is where the network namespace is found and used.
At the layer 2, it is at the network device level where the namespace is

found. At the layer 3, from the IP destination. So when you arrive to
sockets level, you have the network namespace packet destination
information and you search for sockets related to the specific namespace.

-- Daniel

Subject: Re: Re: Network virtualization/isolation
Posted by Vlad Yasevich on Thu, 30 Nov 2006 16:15:11 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Brian Haley wrote:
>> Eric W. Biederman wrote:
>>> I think for cases across network socket namespaces it should
>>> be a matter for the rules, to decide if the connection should
>>> happen and what error code to return if the connection does not
>>> happen.
>>>
>>> There is a potential in this to have an ambiguous case where two
>>> applications can be listening for connections on the same socket
>>> on the same port and both will allow the connection. If that
>>> is the case I believe the proper definition is the first socket
>>> that we find that will accept the connection gets the connection.
> No. If you try to connect, the destination IP address is assigned to a
> network namespace. This network namespace is used to leave the listening
> socket ambiguity.
>>
>> Wouldn't you want to catch this at bind() and/or configuration time and
>> fail? Having overlapping namespaces/rules seems undesirable, since as
>> Herbert said, can get you "unexpected behaviour".
>
> Overlapping is not a problem, you can have several sockets binded on the
> same INADDR_ANY/port without ambiguity because the network namespace
> pointer is added as a new key for sockets lookup, (src addr, src port,
> dst addr, dst port, net ns pointer). The bind should not be forced to a
> specific address because you will not be able to connect via 127.0.0.1.

So, all this leads to me ask, how to handle 127.0.0.1?

For L2 it seems easy. Each namespace gets a tagged lo device.
How do you propose to do it for L3, because disabling access to loopback is
not a valid option, IMO.

I agree that adding a namespace to the (using generic terms) TCB lookup
solves the conflict issue.

-vlad

Subject: Re: Network virtualization/isolation
Posted by jamal on Sun, 03 Dec 2006 12:26:02 GMT
View Forum Message <> Reply to Message

On Wed, 2006-14-11 at 16:17 +0100, Daniel Lezcano wrote:
> The attached document describes the network isolation at the layer 2
> and at the layer 3 ..

Daniel,

I apologize for taking this long to get back to you. The document (I hope) made it clear to me at least the difference between the two approaches. So thanks for taking the time to put it together.

So here are my thoughts ...
I havent read the rest of the thread so i may be repeating some of the discussion; i have time today, I will try to catchup with the discussion.

* i think the L2 approach is the more complete of the two approaches:

It caters to more applications: eg i can have network elements such as virtual bridges and routers. It doesnt seem like i can do that with the L3 approach. I think this in itself is a powerful enough reason to disqualify the L3 approach.

Leading from the above, I dont have to make _a single line of code change_ to any of the network element management tools inside the container. i.e i can just run quagga and OSPF and BGP will work as is or the bridge daemon and STP will work as is or tc to control "real" devices or ip to control "real" ip addresses. Virtual routers and bridges are real world applications (if you want more info ask me or ask google, she knows).

**** This wasnt clear to me from the doc on the L3 side of things, so please correct me:
because of the pid virtualization in the L2 approach(openvz?) I can run all applications as is. They just dont know they are running on a virtual environment. To use an extreme example: if i picked apache as a binary compiled 10 years ago, it will run on the L2 approach but not on the L3 approach. Is this understanding correct? I find it hard to believe that the L3 approach wouldnt work this way - it may be just my reading into the doc.

So lets say the approach taken is that of L2 (I am biased towards this

because i want to be able to do virtual bridges and routers). The disadvantage of the L2 approach (or is it just the openvz?) approach is:

- it is clear theres a lot more code needed to allow for the two level multiplexing every where. i.e first you mux to select the namespace then you do other things like find a pid, netdevice, ip address etc. I am also not sure how complete that code is; you clearly get everything attached to netdevices for free (eg networkc scheduler block) - which is nice in itself; but you may have to do the muxing code for other blocks. If my understanding is correct everything in the net subsystem has this mux levels already in place (at least with openvz)? I think each subsystem may have its own merit discussed (eg the L3 tables with the recent changes from Patrick allow up to 2^32 -1 tables, so a muxing layer at L3 maybe unnecessary)
---> To me this 2 level muxing looks like a clean design in that there is consistency (i.e no hack thats specific to just one sub-subsystem but not others). With this approach one could imagine hardware support that does the first level of muxing (selecting a namespace for you). This is clearly already happening with NICs supporting several unicast MAC addresses.
I think the litmus test for this approach is the answer to the question:
If i compiled in the containers in and do not use the namespaces, how much more overhead is there for the host path? I would hope that it is as close to 0 as possible. It should certainly be 0 if i dont compile in containers.

- The desire for many MAC addresses. I dont think this is a killer issue. NICs are begining to show up which capabilities for many unicast MACs; many current have multicast hardware tables that can be used for stashing unicast MAC addresses; it has also been shown you can use multicast MAC addresses and get away with it if there is no conflict (protocols such as VRRP, CARP etc do this).

- Manageability from the host side. It seems to be more complex with the L2 than with L3. But so what? These tools are written from scratch and there is no "backward compatibility" baggage.

Ok, I am out of coffee for the last 10 minutes;-> But above sit my views worth about $0.02 Canadian (which is about $0.02 US these days).

I will try later to catch up with the discussion that started from Daniels original posting.

cheers,
jamal

Subject: Network virtualization/isolation
Posted by jamal on Sun, 03 Dec 2006 14:13:18 GMT
View Forum Message <> Reply to Message

I have removed the Re: just to add some freshness to the discussion

So i read quickly the rest of the discussions. I was almost suprised to find that i agree with Eric on a lot of opinions (we also agree that vindaloo is good for you i guess);->
The two issues that stood out for me (in addition to what i already said below):

1) the solution must ease the migration of containers; i didnt see anything about migrating them to another host across a network, but i assume that this is a given.
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?
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)?

In any case, folks, i hope i am not treading on anyones toes; i know each one of you has implemented and has users and i am trying to be as neutral as i can (but clearly biased;->).

cheers,
jamal

On Sun, 2006-03-12 at 07:26 -0500, jamal wrote:
> On Wed, 2006-14-11 at 16:17 +0100, Daniel Lezcano wrote:
> > The attached document describes the network isolation at the layer 2
> > and at the layer 3 ..
>
> Daniel,
>

> I apologize for taking this long to get back to you. The document (I
> hope) made it clear to me at least the difference between the two
> approaches. So thanks for taking the time to put it together.
>
> So here are my thoughts ...
> I havent read the rest of the thread so i may be repeating some of the
> discussion; i have time today, I will try to catchup with the
> discussion.
>
> * i think the L2 approach is the more complete of the two approaches:
>
> It caters to more applications: eg i can have network elements such as
> virtual bridges and routers. It doesnt seem like i can do that with the
> L3 approach. I think this in itself is a powerful enough reason to
> disqualify the L3 approach.
>
> Leading from the above, I dont have to make _a single line of code
> change_ to any of the network element management tools inside the
> container. i.e i can just run quagga and OSPF and BGP will work as is or
> the bridge daemon and STP will work as is or tc to control "real"
> devices or ip to control "real" ip addresses. Virtual routers and
> bridges are real world applications (if you want more info ask me or ask
> google, she knows).
>
> **** This wasnt clear to me from the doc on the L3 side of things, so
> please correct me:
> because of the pid virtualization in the L2 approach(openvz?) I can run
> all applications as is. They just dont know they are running on a
> virtual environment. To use an extreme example: if i picked apache as a
> binary compiled 10 years ago, it will run on the L2 approach but not on
> the L3 approach. Is this understanding correct? I find it hard to
> believe that the L3 approach wouldnt work this way - it may be just my
> reading into the doc.
>
> So lets say the approach taken is that of L2 (I am biased towards this
> because i want to be able to do virtual bridges and routers). The
> disadvantage of the L2 approach (or is it just the openvz?) approach is:
>
> - it is clear theres a lot more code needed to allow for the two level
> multiplexing every where. i.e first you mux to select the namespace then
> you do other things like find a pid, netdevice, ip address etc. I am
> also not sure how complete that code is; you clearly get everything
> attached to netdevices for free (eg networkc scheduler block) - which is
> nice in itself; but you may have to do the muxing code for other blocks.
> If my understanding is correct everything in the net subsystem has this
> mux levels already in place (at least with openvz)? I think each
> subsystem may have its own merit discussed (eg the L3 tables with the
> recent changes from Patrick allow up to 2^32 -1 tables, so a muxing

> layer at L3 maybe unnecessary)
> ---> To me this 2 level muxing looks like a clean design in that there
> is consistency (i.e no hack thats specific to just one sub-subsystem but
> not others). With this approach one could imagine hardware support that
> does the first level of muxing (selecting a namespace for you). This is
> clearly already happening with NICs supporting several unicast MAC
> addresses.
> I think the litmus test for this approach is the answer to the question:
> If i compiled in the containers in and do not use the namespaces, how
> much more overhead is there for the host path? I would hope that it is
> as close to 0 as possible. It should certainly be 0 if i dont compile in
> containers.
>
> - The desire for many MAC addresses. I dont think this is a killer
> issue. NICs are begining to show up which capabilities for many unicast
> MACs; many current have multicast hardware tables that can be used for
> stashing unicast MAC addresses; it has also been shown you can use
> multicast MAC addresses and get away with it if there is no conflict
> (protocols such as VRRP, CARP etc do this).
>
> - Manageability from the host side. It seems to be more complex with the
> L2 than with L3. But so what? These tools are written from scratch and
> there is no "backward compatibility" baggage.
>
> Ok, I am out of coffee for the last 10 minutes;-> But above sit my views
> worth about $0.02 Canadian (which is about $0.02 US these days).
>
> I will try later to catch up with the discussion that started from
> Daniels original posting.
>
> cheers,
> jamal

Subject: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Sun, 03 Dec 2006 16:37:19 GMT
View Forum Message <> Reply to Message

On Sun, Dec 03, 2006 at 07:26:02AM -0500, jamal wrote:
> On Wed, 2006-14-11 at 16:17 +0100, Daniel Lezcano wrote:
> > The attached document describes the network isolation at the layer 2
> > and at the layer 3 ..
>
> Daniel,
>
> I apologize for taking this long to get back to you. The document (I
> hope) made it clear to me at least the difference between the two
> approaches. So thanks for taking the time to put it together.

>
> So here are my thoughts ...
> I havent read the rest of the thread so i may be repeating some of the
> discussion; i have time today, I will try to catchup with the
> discussion.
>
> * i think the L2 approach is the more complete of the two approaches:
>
> It caters to more applications: eg i can have network elements such as
> virtual bridges and routers. It doesnt seem like i can do that with the
> L3 approach. I think this in itself is a powerful enough reason to
> disqualify the L3 approach.
>
> Leading from the above, I dont have to make _a single line of code
> change_ to any of the network element management tools inside the
> container. i.e i can just run quagga and OSPF and BGP will work as is or
> the bridge daemon and STP will work as is or tc to control "real"
> devices or ip to control "real" ip addresses. Virtual routers and
> bridges are real world applications (if you want more info ask me or ask
> google, she knows).
>
> **** This wasnt clear to me from the doc on the L3 side of things, so
> please correct me:
> because of the pid virtualization in the L2 approach(openvz?) I can run
> all applications as is. They just dont know they are running on a
> virtual environment. To use an extreme example: if i picked apache as a
> binary compiled 10 years ago, it will run on the L2 approach but not on
> the L3 approach. Is this understanding correct? I find it hard to
> believe that the L3 approach wouldnt work this way - it may be just my
> reading into the doc.

the 10 year old apache will run with layer 3 isolation
as well as with layer 2 virtualization (probably a little
faster though, we do not know yet :), because what it
does is IP (layer 3) traffic ...

> So lets say the approach taken is that of L2 (I am biased towards this
> because i want to be able to do virtual bridges and routers).
> The disadvantage of the L2 approach (or is it just the openvz?)
> approach is:
>
> - it is clear theres a lot more code needed to allow for the two level
> multiplexing every where. i.e first you mux to select the namespace then
> you do other things like find a pid, netdevice, ip address etc. I am
> also not sure how complete that code is; you clearly get everything
> attached to netdevices for free (eg networkc scheduler block) - which is
> nice in itself; but you may have to do the muxing code for other blocks.
> If my understanding is correct everything in the net subsystem has this

> mux levels already in place (at least with openvz)? I think each
> subsystem may have its own merit discussed (eg the L3 tables with the
> recent changes from Patrick allow up to 2^32 -1 tables, so a muxing
> layer at L3 maybe unnecessary)

> ---> To me this 2 level muxing looks like a clean design in that there
> is consistency (i.e no hack thats specific to just one sub-subsystem but
> not others). With this approach one could imagine hardware support that
> does the first level of muxing (selecting a namespace for you). This is
> clearly already happening with NICs supporting several unicast MAC
> addresses.

> I think the litmus test for this approach is the answer to the question:
> If i compiled in the containers in and do not use the namespaces, how
> much more overhead is there for the host path? I would hope that it is
> as close to 0 as possible. It should certainly be 0 if i dont compile in
> containers.

IMHO there are three cases to consider, to get valid
'performance' numbers:

 - host system with and without containers enabled
 - single guest (container) compared to host system _without_
 - bunch of guests (e.g. 10) compared to 10 apps/threads on host

one proven feature of the L3 isolation is that those
all end up with the same or even better performance

> - The desire for many MAC addresses. I dont think this is a killer
> issue. NICs are begining to show up which capabilities for many unicast
> MACs; many current have multicast hardware tables that can be used for
> stashing unicast MAC addresses; it has also been shown you can use
> multicast MAC addresses and get away with it if there is no conflict
> (protocols such as VRRP, CARP etc do this).
>
> - Manageability from the host side. It seems to be more complex with the
> L2 than with L3. But so what? These tools are written from scratch and
> there is no "backward compatibility" baggage.

well, no, actually the 'tools' to manage layer 3 isolation
are already there, and except for the 'setup' there is
nothing special to configure, as networking still lives
on the host

> Ok, I am out of coffee for the last 10 minutes;-> But above sit my views
> worth about $0.02 Canadian (which is about $0.02 US these days).
>
> I will try later to catch up with the discussion that started from

> Daniels original posting.

I would be interested in a config layout for a typical
L3 isolation setup when you 'only' have L2 virtualization

- typical host system with apache, mysql, postfix, ssh
  and ftp is broken down into security contexts to
  allow for increased security
- as part of that process, the services are isolated,
  while apache and ftp share the same ip [ip0], mysql
  will be using a local one [ip1], and postfix/ssh a
  second public one [ip2]

the L3 isolation approach is straight forward:

- assign the two public ips to eth0, the local one
  to lo or dummy0
- create five isolation areas where 0 and 1 share ip0,
  2 uses ip1 and 3,4 uses ip2

that's it, all will work as expected ... let's see with
what L2 isolation example you come up with, which is
able to 'mimic' this setup ...

note: no question it is possible to do that with L2

best,
Herbert

> cheers,
> jamal
>
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by jamal on Sun, 03 Dec 2006 16:58:30 GMT
View Forum Message <> Reply to Message

On Sun, 2006-03-12 at 17:37 +0100, Herbert Poetzl wrote:
> On Sun, Dec 03, 2006 at 07:26:02AM -0500, jamal wrote:

> To use an extreme example: if i picked apache as a
> > binary compiled 10 years ago, it will run on the L2 approach but not on
> > the L3 approach. Is this understanding correct? I find it hard to
> > believe that the L3 approach wouldnt work this way - it may be just my
> > reading into the doc.
>
> the 10 year old apache will run with layer 3 isolation
> as well as with layer 2 virtualization (probably a little
> faster though, we do not know yet :), because what it
> does is IP (layer 3) traffic ...
>

Ok, thanks for clarifying this.

> > I think the litmus test for this approach is the answer to the question:
> > If i compiled in the containers in and do not use the namespaces, how
> > much more overhead is there for the host path? I would hope that it is
> > as close to 0 as possible. It should certainly be 0 if i dont compile in
> > containers.
>
> IMHO there are three cases to consider, to get valid
> 'performance' numbers:
>
> - host system with and without containers enabled
> - single guest (container) compared to host system _without_

Sound reasonable.

> - bunch of guests (e.g. 10) compared to 10 apps/threads on host
>

Your mileage may vary. For me trying to run virtual routers; this is
not an important test. I want to be able to have containers each running
quagga and OSPF. I cant achieve my goals with with 10 quaggas without
making some major changes to quagga.

> one proven feature of the L3 isolation is that those
> all end up with the same or even better performance

I think it is valuable to reduce the overhead. I think that it may be
reasonable to some threshold to trade a little performance for
genericity. What the threshold is, i dont know.

> > - Manageability from the host side. It seems to be more complex with the
> > L2 than with L3. But so what? These tools are written from scratch and
> > there is no "backward compatibility" baggage.
>

> well, no, actually the 'tools' to manage layer 3 isolation
> are already there,
> and except for the 'setup' there is
> nothing special to configure, as networking still lives
> on the host
>

I dont see the two as being separate issues. You must create container;
you must configure networking on them; it is forgivable to have the
second part of that process to involve some non-standard tools for the
containers (from the host).
It is not forgivable to have speacilized tools within the container.

> I would be interested in a config layout for a typical
> L3 isolation setup when you 'only' have L2 virtualization
>
> - typical host system with apache, mysql, postfix, ssh
>   and ftp is broken down into security contexts to
>   allow for increased security
> - as part of that process, the services are isolated,
>   while apache and ftp share the same ip [ip0], mysql
>   will be using a local one [ip1], and postfix/ssh a
>   second public one [ip2]
>
> the L3 isolation approach is straight forward:
>
> - assign the two public ips to eth0, the local one
>   to lo or dummy0
> - create five isolation areas where 0 and 1 share ip0,
>   2 uses ip1 and 3,4 uses ip2
>
> that's it, all will work as expected ... let's see with
> what L2 isolation example you come up with, which is
> able to 'mimic' this setup ...
>
> note: no question it is possible to do that with L2

Unless i am misreading, isnt this merely a matter of configuring
on the container side eth0 (I think you are talking about VE side eth0
in your example above) two public ip addresses (or even two ethx
devices) and then attach IP addresses to them? mysql gets an lo address.
Would this not work?

Out of curiosity: assume we have a local LAN (perhaps something upstream
does NAT), is it possible to have the same IP address going to multiple
containers?

cheers,

jamal

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Daniel Lezcano on Mon, 04 Dec 2006 10:18:09 GMT
View Forum Message <> Reply to Message

Hi Jamal,

thanks for taking the time read the document.

The objective of the document was not to convince one approach is better than other. I wanted to show the pros and the cons of each approach and to point that the 2 approaches are complementary.

Currently, there are some resources moved to a namespace relative access, the IPC and the utsname and this is into the 2.6.19 kernel. The work on the pid namespace is still in progress.

The idea is to use a "clone" approach relying on the "unshare_ns" syscall. The syscall is called with a set of flags for pids, ipcs, utsname, network ... You can then "unshare" only the network and have an application into its own network environment.

For a l3 approach, like a l2, you can run an apache server into a unshared network environment. Better, you can run several apaches server into several network namespaces without modifying the server's network configuration.

Some of us, consider l2 as perfectly adapted for some kind of containers like system containers and some kind of application containers running big servers, but find the l2 too much (seems to be a hammer to crush a beetle) for simple network requirements like for network migration, jails or containers which does not take care of such virtualization. For example, you want to create thousands of containers for a cluster of HPC jobs and just to have migration for these jobs. Does it make sense to have l2 approach ?

Dmitry Mishin and I, we thought about a l2/l3 solution and we thing we found a solution to have the 2 at runtime. Roughly, it is a l3 based on bind filtering and socket isolation, very similar to what vserver

provides. I did a prototype, and it works well for IPV4/unicast.

So, considering, we have a l2 isolation/virtualization, and having a l3
relying on the l2 network isolation resources subset. Is it an
acceptable solution ?

-- Daniel

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Mon, 04 Dec 2006 12:15:00 GMT
View Forum Message <> Reply to Message

jamal <hadi@cyberus.ca> writes:

> I have removed the Re: just to add some freshness to the discussion
>
> So i read quickly the rest of the discussions. I was almost suprised to
> find that i agree with Eric on a lot of opinions (we also agree that
> vindaloo is good for you i guess);->
> The two issues that stood out for me (in addition to what i already said
> below):
>
> 1) the solution must ease the migration of containers; i didnt see
> anything about migrating them to another host across a network, but i
> assume that this is a given.

It is mostly a given. It is a goal for some of us and not for others.
Containers are a necessary first step to getting migration and checkpoint/restart
assistance from the kernel.

> 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.

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

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.

> In any case, folks, i hope i am not treading on anyones toes; i know
> each one of you has implemented and has users and i am trying to be as
> neutral as i can (but clearly biased;->).

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.

Eric

Subject: Re: Network virtualization/isolation
Posted by jamal on Mon, 04 Dec 2006 13:22:37 GMT
View Forum Message <> Reply to Message

Daniel,

On Mon, 2006-04-12 at 11:18 +0100, Daniel Lezcano wrote:
> Hi Jamal,

> Currently, there are some resources moved to a namespace relative
> access, the IPC and the utsname and this is into the 2.6.19 kernel.
> The work on the pid namespace is still in progress.
>
> The idea is to use a "clone" approach relying on the "unshare_ns"
> syscall. The syscall is called with a set of flags for pids, ipcs,
> utsname, network ... You can then "unshare" only the network and have an
> application into its own network environment.

>

Ok, so i take it this call is used by the setup manager on the host
side?

> For a l3 approach, like a l2, you can run an apache server into a
> unshared network environment. Better, you can run several apaches server
> into several network namespaces without modifying the server's network
> configuration.
>

ok - as i understand it now, this will be the case for all the
approaches taken?

> Some of us, consider l2 as perfectly adapted for some kind of containers
> like system containers and some kind of application containers running
> big servers, but find the l2 too much (seems to be a hammer to crush a
> beetle) for simple network requirements like for network migration,
> jails or containers which does not take care of such virtualization. For
> example, you want to create thousands of containers for a cluster of HPC
> jobs and just to have migration for these jobs. Does it make sense to
> have l2 approach ?
>

Perhaps not for the specific app you mentioned above.
But it makes sense for what i described as virtual routers/bridges.
I would say that the solution has to cater for a variety of
applications, no?

> Dmitry Mishin and I, we thought about a l2/l3 solution and we thing we
> found a solution to have the 2 at runtime. Roughly, it is a l3 based on
> bind filtering and socket isolation, very similar to what vserver
> provides. I did a prototype, and it works well for IPV4/unicast.
>

ok - so you guys seem to be reaching at least some consensus then.

> So, considering, we have a l2 isolation/virtualization, and having a l3
> relying on the l2 network isolation resources subset. Is it an
> acceptable solution ?

As long as you can be generic enough so that a wide array of apps can be
met, it should be fine. For a test app, consider the virtual
bridges/routers i mentioned.
The other requirement i would see is that apps that would run on a host
would run unchanged. The migration of containers you folks seem to be
having under control - my only input into that thought since it is early
enough, you may want to build your structuring in such a way that this

is easy to do.

cheers,
jamal

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

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Mon, 04 Dec 2006 15:35:42 GMT
View Forum Message <> Reply to Message

jamal <hadi@cyberus.ca> writes:

> 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.

Having the proper semantics is a MUST, which generally makes those
a requirement to get consensus and to build the general mergeable
solution.

The logic for serializing the state is totally uninteresting
for the first pass at containers. The applications inside the
containers simply don't care.

There are two basic techniques for containers.
1) Name filtering.
   Where you keep the same global identifiers as you do now, but
   applications inside the container are only allowed to deal with a
   subset of those names. The current vserver layer 3 networking
   approach is a handy example of this. But this can apply to process
   ids and just about everything else.

2) Independent namespaces. (Name duplication)
   Where you allow the same global name to refer to two different
   objects at the same time, with the context the reference comes
   being used to resolve which global object you are talking about.

Independent namespaces are the only core requirement for migration,
because the ensure when you get to the next machine you don't have
a conflict with your global names.

So at this point simply allowing duplicate names is the only
requirement for migration. But yes that part is a MUST.

>> > 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? ;->

Yes. As far as I can tell this is what we are doing and generally
it doesn't even require a hash to get the namespace. Just an appropriate
place to look for the pointer to the namespace structure.

The practical problem with socket lookup is that is a hash table today,
allocating the top level of that hash table dynamically at run-time looks
problematic, as it is more than a single page.

>> > 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.

Not for namespaces, the problem is for existing hash tables, like the
ipv4 routing cache, and for the sockets...

>> 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.

I don't think we have gone far enough to prove what has good performance.

>> 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?

For non networking stuff you can look at the uts and ipc namespaces
that have been merged into 2.6.19. There is also the struct pid work
that is a lead up to the pid namespace.

We have very carefully broken the problem by subsystem so we can do
incremental steps to get container support into the kernel.

That I don't think is the answer you want I think you are looking
for networking stack agreement. If we had that we would be submitting
patches at the moment.

The OpenVz and Vserver code is available.

I have my proof of concept git tree up on kernel.org which has a terribly
messing history but it's network stack is largely the L2 we are talking about.

>> 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;

I thought that was what I just said! Yes you are 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.

As a general technique I am in favor of that as well. The intrusive
part is that you have to touch every global variable access in the
networking stack. It is fairly clean and fairly non-intrusive, and
certainly makes it easy to that the patch does nothing nasty. But you
do have to touch a lot of code.

It occurs to me that what we really want as a first step to this
is simply to implement noop versions of the accessors functions we
are going to need. That way we can merge the intrusive bits before
we do the actual implementation.

If we could get a reasonably clear design with how to do the variable
accesses and the incremental approach to changing all of them, I think
we would see a lot less resistance to the L2 work. Simply because
it would go from a mammoth patch to a relatively small patch.

Thinking out loud. As far as I have seen there are two paths for
looking up the network namespace.

- Packets transmitted out of the kernel.
- Packets being received by the kernel.

For out going packets we can look at the socket to find the
network namespace. For in coming packets we can look at the
network device. In neither case do we actually have to tag
the packets themselves.

In addition there are a couple of weird cases with things like
ARP. Where the kernel generates the reply packet without the
packet really coming all of the way into the kernel, that
I recall having to special case.

Is that roughly what you were thinking with respect to finding
the current network namespace?

Where and when you look to find the network namespace that applies to
a packet is the primary difference between the OpenVZ L2
implementation and my L2 implementation.

If there is a better and less intrusive while still being obvious
method I am all for it. I do not like the OpenVZ thing of doing the
lookup once and then stashing the value in current and the special
casing the exceptions.

For me I'm just trying to keep everyone from going in really insane
directions, while I work through the really non-obvious bits like
how do we successfully handle sysctl, and sysfs. Those things that
must be refactored to be able to cope with multiple instances of
some of the primary kernel data structures.

Eric

Subject: Re: Network virtualization/isolation
Posted by Mishin Dmitry on Mon, 04 Dec 2006 16:00:42 GMT
View Forum Message <> Reply to Message

On Monday 04 December 2006 18:35, Eric W. Biederman wrote:
[skip]
> Where and when you look to find the network namespace that applies to
> a packet is the primary difference between the OpenVZ L2
> implementation and my L2 implementation.
>
> If there is a better and less intrusive while still being obvious
> method I am all for it. I do not like the OpenVZ thing of doing the
> lookup once and then stashing the value in current and the special
> casing the exceptions.
Why?

--
Thanks,
Dmitry.

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Mon, 04 Dec 2006 16:52:10 GMT
View Forum Message <> Reply to Message

Dmitry Mishin <dim@openvz.org> writes:

> On Monday 04 December 2006 18:35, Eric W. Biederman wrote:

> [skip]
>> Where and when you look to find the network namespace that applies to
>> a packet is the primary difference between the OpenVZ L2
>> implementation and my L2 implementation.
>>
>> If there is a better and less intrusive while still being obvious
>> method I am all for it. I do not like the OpenVZ thing of doing the
>> lookup once and then stashing the value in current and the special
>> casing the exceptions.
> Why?

I like it when things are obvious and not implied.

The implementations seems to favor fewer lines of code touched over
maintainability of the code. Which if you are maintaining out of
tree code is fine. At leas that was my impression last time
I looked at the code.

I know there are a lot of silly things in the existing implementations
because they were initially written without the expectation of being
able to merge the code into the main kernel. This resulted in some
non-general interfaces, and a preference for patches that touch
as few lines of code as possible.

Anyway this has bit has been discussed before and we can discuss it
seriously in the context of patch review.

Eric

Subject: Re: Network virtualization/isolation
Posted by ebiederm on Mon, 04 Dec 2006 16:58:04 GMT
View Forum Message <> Reply to Message

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

> On Mon, Dec 04, 2006 at 06:19:00PM +0300, Dmitry Mishin wrote:
>> On Sunday 03 December 2006 19:00, Eric W. Biederman wrote:
>> > Ok. Just a quick summary of where I see the discussion.
>> >
>> > We all agree that L2 isolation is needed at some point.
>
>> As we all agreed on this, may be it is time to send patches
>> one-by-one? For the beggining, I propose to resend Cedric's
>> empty namespace patch as base for others - it is really empty,
>> but necessary in order to move further.
>>
>> After this patch and the following net namespace unshare

>> patch will be accepted,
>
> well, I have neither seen any performance tests showing
> that the following is true:
>
> - no change on network performance without the
>   space enabled
> - no change on network performance on the host
>   with the network namespaces enabled
> - no measureable overhead inside the network
>   namespace
> - good scaleability for a larger number of network
>   namespaces

Yes all important criteria for selecting the implementation.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Mon, 04 Dec 2006 17:19:26 GMT
View Forum Message <> Reply to Message

On Mon, Dec 04, 2006 at 08:02:48PM +0300, Dmitry Mishin wrote:
> On Monday 04 December 2006 19:43, Herbert Poetzl wrote:
> > On Mon, Dec 04, 2006 at 06:19:00PM +0300, Dmitry Mishin wrote:
> > > On Sunday 03 December 2006 19:00, Eric W. Biederman wrote:
> > > > Ok. Just a quick summary of where I see the discussion.
> > > >
> > > > We all agree that L2 isolation is needed at some point.
> > >
> > > As we all agreed on this, may be it is time to send patches
> > > one-by-one? For the beggining, I propose to resend Cedric's
> > > empty namespace patch as base for others - it is really empty,
> > > but necessary in order to move further.
> > >
> > > After this patch and the following net namespace unshare
> > > patch will be accepted,
> >
> > well, I have neither seen any performance tests showing
> > that the following is true:
> >
> > - no change on network performance without the
> >   space enabled

> > - no change on network performance on the host
> >   with the network namespaces enabled
> > - no measureable overhead inside the network
> >   namespace
> > - good scaleability for a larger number of network
> >   namespaces

> These questions are for complete L2 implementation,
> not for these 2 empty patches.

well, I fear that we will have lot of overhead
'sneaking' in via small patches (with almost
unnoticeable overhead) making the 2.6 branch slower
and slower (regarding networking) so IMHO a complete
solution should be drafted, and tested performance
wise, we can then adjust it and possibley improve
it, untill it shows no measureable overhead ...

but IMHO it should be developed 'outside' the kernel,
in small and reviewable pieces which are constantly
updated to match the recent kernels ... something
like stacked git or quilt ...

> If you need some data relating to Andrey's
> implementation, I'll get it. Which test do you accept?

hmm, I think a good mix of netperf, netpipe and
iperf would be a good start, probably network folks
know better tests to exercise the stack ... at least
I hope so ...

of course, a good explanation _why_ this or that
code path does not add overhead here or there is
nice to have too ...

best,
Herbert

> > > I could send network devices virtualization patches for
> > > review and discussion.
> >
> > that won't hurt ...
> >
> > best,
> > Herbert
> >
> > > What do you think?
> > >

> > > > The approaches discussed for L2 and L3 are sufficiently orthogonal
> > > > that we can implement then in either order. You would need to
> > > > unshare L3 to unshare L2, but if we think of them as two separate
> > > > namespaces we are likely to be in better shape.
> > > >
> > > > The L3 discussion still has the problem that there has not been
> > > > agreement on all of the semantics yet.
> > > >
> > > > More comments after I get some sleep.
> > > >
> > > > Eric
> > > > -
> > > > To unsubscribe from this list: send the line "unsubscribe netdev" in
> > > > the body of a message to majordomo@vger.kernel.org
> > > > More majordomo info at http://vger.kernel.org/majordomo-info.html
> > >
> > > --
> > > Thanks,
> > > Dmitry.
> > > _____
> > > Containers mailing list
> > > Containers@lists.osdl.org
> > > https://lists.osdl.org/mailman/listinfo/containers
>
> --
> Thanks,
> Dmitry.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Network virtualization/isolation
Posted by Daniel Lezcano on Mon, 04 Dec 2006 17:41:33 GMT
View Forum Message <> Reply to Message

Dmitry Mishin wrote:
> On Monday 04 December 2006 19:43, Herbert Poetzl wrote:
>> On Mon, Dec 04, 2006 at 06:19:00PM +0300, Dmitry Mishin wrote:
>>> On Sunday 03 December 2006 19:00, Eric W. Biederman wrote:
>>>> Ok. Just a quick summary of where I see the discussion.
>>>>
>>>> We all agree that L2 isolation is needed at some point.
>>> As we all agreed on this, may be it is time to send patches
>>> one-by-one? For the beggining, I propose to resend Cedric's
>>> empty namespace patch as base for others - it is really empty,
>>> but necessary in order to move further.

>>>
>>> After this patch and the following net namespace unshare
>>> patch will be accepted,
>> well, I have neither seen any performance tests showing
>> that the following is true:
>>
>> - no change on network performance without the
>>   space enabled
>> - no change on network performance on the host
>>   with the network namespaces enabled
>> - no measureable overhead inside the network
>>   namespace
>> - good scaleability for a larger number of network
>>   namespaces
> These questions are for complete L2 implementation, not for these 2 empty
> patches. If you need some data relating to Andrey's implementation, I'll get
> it. Which test do you accept?

tbench ?

With the following scenarii:

  * intra host communication (one time with IP on eth and one time with
127.0.0.1)
  * inter host communication

Each time:
 - a single network namespace
 - with 100 network namespace. 1 server communicating and 99 listening
but doing nothing.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by dev on Wed, 06 Dec 2006 11:45:04 GMT
View Forum Message <> Reply to Message

>>>If there is a better and less intrusive while still being obvious
>>>method I am all for it. I do not like the OpenVZ thing of doing the
>>>lookup once and then stashing the value in current and the special
>>>casing the exceptions.
>>
>>Why?

>
>
> I like it when things are obvious and not implied.
>
> The implementations seems to favor fewer lines of code touched over
> maintainability of the code. Which if you are maintaining out of
> tree code is fine. At leas that was my impression last time
> I looked at the code.
FYI, when we started doing networking virtualization many years ago
we tried both approaches.
Over time, context notion looked much more natural and easier for us.
Even Alexey Kuznetsov tells that he prefers exec_env as the logic
becomes very clear and little mess is introduced.

> I know there are a lot of silly things in the existing implementations
> because they were initially written without the expectation of being
> able to merge the code into the main kernel. This resulted in some
> non-general interfaces, and a preference for patches that touch
> as few lines of code as possible.
Sure, but OpenVZ code is being constantly cleaned from such code
and we are open for discussion. No one pretends that code is perferct
from the beginning.

> Anyway this has bit has been discussed before and we can discuss it
> seriously in the context of patch review.
Let me explain when explicit context like exec_env IMHO is cleaner:
- context is a natural notion of linux kernel. e.g. current.
  why not pass 'current' to all the functions as an argument
  starting from entry.S?
  in_atomic(), in_interrupt() etc. all these functions deal with current context.
  IMHO when one needs to pass an argument too many times like 'current'
  it is better to use a notion of the context.
- e.g. NFS should set networking context of the mount point or socket.
But, ok, it is not the real point to argue so much imho and waste our time instead of
doing things.

Thanks,
Kirill

Subject: Re: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Wed, 06 Dec 2006 18:30:36 GMT
View Forum Message <> Reply to Message

On Wed, Dec 06, 2006 at 02:54:16PM +0300, Kirill Korotaev wrote:
> >>>If there is a better and less intrusive while still being obvious
> >>>method I am all for it. I do not like the OpenVZ thing of doing the
> >>>lookup once and then stashing the value in current and the special

> >>>casing the exceptions.
> >>
> >>Why?
> >
> >
> > I like it when things are obvious and not implied.
> >
> > The implementations seems to favor fewer lines of code touched over
> > maintainability of the code. Which if you are maintaining out of
> > tree code is fine. At leas that was my impression last time
> > I looked at the code.

> FYI, when we started doing networking virtualization many years ago
> we tried both approaches.
> Over time, context notion looked much more natural and easier for us.
> Even Alexey Kuznetsov tells that he prefers exec_env as the logic
> becomes very clear and little mess is introduced.
>
> > I know there are a lot of silly things in the existing implementations
> > because they were initially written without the expectation of being
> > able to merge the code into the main kernel. This resulted in some
> > non-general interfaces, and a preference for patches that touch
> > as few lines of code as possible.
> Sure, but OpenVZ code is being constantly cleaned from such code
> and we are open for discussion. No one pretends that code is perferct
> from the beginning.
>
> > Anyway this has bit has been discussed before and we can discuss it
> > seriously in the context of patch review.
> Let me explain when explicit context like exec_env IMHO is cleaner:
> - context is a natural notion of linux kernel. e.g. current.
>   why not pass 'current' to all the functions as an argument
>   starting from entry.S?
>   in_atomic(), in_interrupt() etc. all these functions deal with
>   current context. IMHO when one needs to pass an argument too many
>   times like 'current'
>   it is better to use a notion of the context.
> - e.g. NFS should set networking context of the mount point or socket.

how would that work for a 'shared' NFS partition?
(shared between different context)

> But, ok, it is not the real point to argue so much imho and waste our
> time instead of doing things.

well, IMHO better talk (and think) first, then implement
something ... not the other way round, and then start
fixing up the mess ...

best,
Herbert

> Thanks,
> Kirill
>
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by ebiederm on Fri, 08 Dec 2006 19:57:49 GMT
View Forum Message <> Reply to Message

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

>> But, ok, it is not the real point to argue so much imho and waste our
>> time instead of doing things.
>
> well, IMHO better talk (and think) first, then implement
> something ... not the other way round, and then start
> fixing up the mess ...

Well we need a bit of both.

This is thankfully not exported to user space, so as long as our
implementation is correct it doesn't much matter.

I do agree with the point that context may make sense. I have
yet to be convinced though.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Sat, 09 Dec 2006 03:50:02 GMT
View Forum Message <> Reply to Message

On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:
> Herbert Poetzl <herbert@13thfloor.at> writes:

>
> >> But, ok, it is not the real point to argue so much imho
> >> and waste our time instead of doing things.

> > well, IMHO better talk (and think) first, then implement
> > something ... not the other way round, and then start
> > fixing up the mess ...
>
> Well we need a bit of both.

hmm, are 'we' in a hurry here?

until recently, 'Linux' (mainline) didn't even want
to hear about OS Level virtualization, now there
is a rush to quickly get 'something' in, not knowing
or caring if it is usable at all?

I think there are a lot of 'potential users' for
this kind of virtualization, and so 'we' can test
almost all aspects outside of mainline, and once
we know the stuff works as expected, then we can
integrate it ...

the UTS namespace was something 'we all' had already
implemented in this (or a very similar) way, and in
one or two interations, it should actually work as
expected. nevertheless, it was one of the simplest
spaces ...

we do not yet know the details for the IPC namespace,
as IPC is not that easy to check as UTS, and 'we'
haven't gotten real world feedback on that yet ...

so personally I think we should start some serious
testing on the upcoming namespaces, and we should
continue discussing the various approaches, until
'we' can agree on the (almost) 'perfect' solution

> This is thankfully not exported to user space, so as long
> as our implementation is correct it doesn't much matter.

that's something I do not really agree with, stuff
integrated into the kernel should be well designed
and it should be tested ...

best,
Herbert

> I do agree with the point that context may make sense.
> I have yet to be convinced though.
>
> Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Andrew Morton on Sat, 09 Dec 2006 06:13:48 GMT
View Forum Message <> Reply to Message

On Sat, 9 Dec 2006 04:50:02 +0100
Herbert Poetzl <herbert@13thfloor.at> wrote:

> On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:
> > Herbert Poetzl <herbert@13thfloor.at> writes:
> >
> > >> But, ok, it is not the real point to argue so much imho
> > >> and waste our time instead of doing things.
> >
> > > well, IMHO better talk (and think) first, then implement
> > > something ... not the other way round, and then start
> > > fixing up the mess ...
> >
> > Well we need a bit of both.
>
> hmm, are 'we' in a hurry here?
>
> until recently, 'Linux' (mainline) didn't even want
> to hear about OS Level virtualization, now there
> is a rush to quickly get 'something' in, not knowing
> or caring if it is usable at all?

It's actually happening quite gradually and carefully.

> I think there are a lot of 'potential users' for
> this kind of virtualization, and so 'we' can test
> almost all aspects outside of mainline, and once
> we know the stuff works as expected, then we can
> integrate it ...
>
> the UTS namespace was something 'we all' had already
> implemented in this (or a very similar) way, and in
> one or two interations, it should actually work as
> expected. nevertheless, it was one of the simplest

> spaces ...
>
> we do not yet know the details for the IPC namespace,
> as IPC is not that easy to check as UTS, and 'we'
> haven't gotten real world feedback on that yet ...

We are very dependent upon all stakeholders including yourself to review,
test and comment upon this infrastructure as it is proposed and merged.
If something is proposed which will not suit your requirements then it
is important that we hear about it, in detail, at the earliest possible time.

Thanks.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Sat, 09 Dec 2006 06:35:17 GMT
View Forum Message <> Reply to Message

On Fri, Dec 08, 2006 at 10:13:48PM -0800, Andrew Morton wrote:
> On Sat, 9 Dec 2006 04:50:02 +0100
> Herbert Poetzl <herbert@13thfloor.at> wrote:
>
> > On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:
> > > Herbert Poetzl <herbert@13thfloor.at> writes:
> > >
> > > >> But, ok, it is not the real point to argue so much imho
> > > >> and waste our time instead of doing things.
> >
> > > > well, IMHO better talk (and think) first, then implement
> > > > something ... not the other way round, and then start
> > > > fixing up the mess ...
> > >
> > > Well we need a bit of both.
> >
> > hmm, are 'we' in a hurry here?
> >
> > until recently, 'Linux' (mainline) didn't even want
> > to hear about OS Level virtualization, now there
> > is a rush to quickly get 'something' in, not knowing
> > or caring if it is usable at all?
>
> It's actually happening quite gradually and carefully.

hmm, I must have missed a testing phase for the

IPC namespace then, not that I think it is broken
(well, maybe it is, we do not know yet)

> > I think there are a lot of 'potential users' for
> > this kind of virtualization, and so 'we' can test
> > almost all aspects outside of mainline, and once
> > we know the stuff works as expected, then we can
> > integrate it ...
> >
> > the UTS namespace was something 'we all' had already
> > implemented in this (or a very similar) way, and in
> > one or two interations, it should actually work as
> > expected. nevertheless, it was one of the simplest
> > spaces ...
> >
> > we do not yet know the details for the IPC namespace,
> > as IPC is not that easy to check as UTS, and 'we'
> > haven't gotten real world feedback on that yet ...
>
> We are very dependent upon all stakeholders including yourself
> to review, test and comment upon this infrastructure as it is
> proposed and merged. If something is proposed which will not
> suit your requirements then it is important that we hear about
> it, in detail, at the earliest possible time.

okay, good to hear that I'm still considered a stakeholder

will try to focus the feedback and cc as many folks
as possible, as it seems that some feedback is lost
on the way upstream ...

best,
Herbert

> Thanks.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by ebiederm on Sat, 09 Dec 2006 08:07:22 GMT
View Forum Message <> Reply to Message

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

> On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:

>> Herbert Poetzl <herbert@13thfloor.at> writes:
>>
>> >> But, ok, it is not the real point to argue so much imho
>> >> and waste our time instead of doing things.
>
>> > well, IMHO better talk (and think) first, then implement
>> > something ... not the other way round, and then start
>> > fixing up the mess ...
>>
>> Well we need a bit of both.
>
> hmm, are 'we' in a hurry here?

We need to talk about code, and particular patches not just talk.

There are two sides to what we are building.
- The user interface, and semantics.
- The kernel implementation.

For the user interface getting it as close to perfect as we can
the first time is extremely important. Because we won't be able
to change it.

For the kernel implementation we don't have to be perfect we have
to have something that is good enough. We can change the
implementation every release if we find better ways of implementing
our user space semantics.

> until recently, 'Linux' (mainline) didn't even want
> to hear about OS Level virtualization, now there
> is a rush to quickly get 'something' in, not knowing
> or caring if it is usable at all?
>
> I think there are a lot of 'potential users' for
> this kind of virtualization, and so 'we' can test
> almost all aspects outside of mainline, and once
> we know the stuff works as expected, then we can
> integrate it ...

We should do this as part of the linux kernel community. There
is no outside of mainline development. We need to get feedback from
other developers whose code we may effect. This is particularly true
of the kernel networking stack.

> that's something I do not really agree with, stuff
> integrated into the kernel should be well designed
> and it should be tested ...

Yes but you can break a problem into reasonable chunks and
solve each of those pieces individually.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Tomasz Torcz on Sat, 09 Dec 2006 11:27:34 GMT
View Forum Message <> Reply to Message

On Sat, Dec 09, 2006 at 04:50:02AM +0100, Herbert Poetzl wrote:
> On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:
> > Herbert Poetzl <herbert@13thfloor.at> writes:
> >
> > >> But, ok, it is not the real point to argue so much imho
> > >> and waste our time instead of doing things.
> >
> > > well, IMHO better talk (and think) first, then implement
> > > something ... not the other way round, and then start
> > > fixing up the mess ...
> >
> > Well we need a bit of both.
>
> hmm, are 'we' in a hurry here?
>
> until recently, 'Linux' (mainline) didn't even want
> to hear about OS Level virtualization, now there
> is a rush to quickly get 'something' in, not knowing
> or caring if it is usable at all?

  Maybe beacuse other Operating Systems have it? For example Solaris'
Crossbow...

--
Tomasz Torcz   RIP is irrevelant. Spoofing is futile.
zdzichu@irc.-nie.spam-.pl   Your routes will be aggreggated. -- Alex Yuriev

_____

Subject: Re: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Sat, 09 Dec 2006 19:04:14 GMT
View Forum Message <> Reply to Message

On Sat, Dec 09, 2006 at 12:27:34PM +0100, Tomasz Torcz wrote:
> On Sat, Dec 09, 2006 at 04:50:02AM +0100, Herbert Poetzl wrote:
> > On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:
> > > Herbert Poetzl <herbert@13thfloor.at> writes:
> > >
> > >> But, ok, it is not the real point to argue so much imho
> > >> and waste our time instead of doing things.
> >
> > > well, IMHO better talk (and think) first, then implement
> > > something ... not the other way round, and then start
> > > fixing up the mess ...
> > >
> > > Well we need a bit of both.
> >
> > hmm, are 'we' in a hurry here?
> >
> > until recently, 'Linux' (mainline) didn't even want
> > to hear about OS Level virtualization, now there
> > is a rush to quickly get 'something' in, not knowing
> > or caring if it is usable at all?
>
> Maybe beacuse other Operating Systems have it?

well, that wasn't a good enough reason four years
ago, when Linux-VServer tried to push a 'jail'
implementation into mainline (was called security
contexts back then, and maintained by Jacques Gelinas)

> For example Solaris' Crossbow...

yes, but the technology isn't really new, not even
on Linux and not even in the Open Source community

but don't get me wrong here, I'm absolutely for
having virtualization (or virtualization elements)
in mainline, I just don't want to see a Q&D hack
'we' have to suffer from the next two years :)

HTC,

Herbert

> --
> Tomasz Torcz RIP is irrevelant. Spoofing is futile.
> zdzichu@irc.-nie.spam-.pl Your routes will be aggreggated. -- Alex Yuriev
>

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Mishin Dmitry on Sat, 09 Dec 2006 21:18:08 GMT
View Forum Message <> Reply to Message

On Saturday 09 December 2006 09:35, Herbert Poetzl wrote:
> On Fri, Dec 08, 2006 at 10:13:48PM -0800, Andrew Morton wrote:
> > On Sat, 9 Dec 2006 04:50:02 +0100
> > Herbert Poetzl <herbert@13thfloor.at> wrote:
> >
> > > On Fri, Dec 08, 2006 at 12:57:49PM -0700, Eric W. Biederman wrote:
> > > > Herbert Poetzl <herbert@13thfloor.at> writes:
> > > >
> > > > >> But, ok, it is not the real point to argue so much imho
> > > > >> and waste our time instead of doing things.
> > > >
> > > > > well, IMHO better talk (and think) first, then implement
> > > > > something ... not the other way round, and then start
> > > > > fixing up the mess ...
> > > >
> > > > Well we need a bit of both.
> > >
> > > hmm, are 'we' in a hurry here?
> > >
> > > until recently, 'Linux' (mainline) didn't even want
> > > to hear about OS Level virtualization, now there
> > > is a rush to quickly get 'something' in, not knowing
> > > or caring if it is usable at all?
> >
> > It's actually happening quite gradually and carefully.
>
> hmm, I must have missed a testing phase for the
> IPC namespace then, not that I think it is broken
> (well, maybe it is, we do not know yet)
Herbert,

you know that this code is used in our product. And in its turn, our
product is tested internally and by a community. We have no reports about
bugs in this code. If you have to say more than just "something to say",
please, say it.

>
> > > I think there are a lot of 'potential users' for
> > > this kind of virtualization, and so 'we' can test
> > > almost all aspects outside of mainline, and once
> > > we know the stuff works as expected, then we can
> > > integrate it ...
> > >
> > > the UTS namespace was something 'we all' had already
> > > implemented in this (or a very similar) way, and in
> > > one or two interations, it should actually work as
> > > expected. nevertheless, it was one of the simplest
> > > spaces ...
> > >
> > > we do not yet know the details for the IPC namespace,
> > > as IPC is not that easy to check as UTS, and 'we'
> > > haven't gotten real world feedback on that yet ...
> >
> > We are very dependent upon all stakeholders including yourself
> > to review, test and comment upon this infrastructure as it is
> > proposed and merged. If something is proposed which will not
> > suit your requirements then it is important that we hear about
> > it, in detail, at the earliest possible time.
>
> okay, good to hear that I'm still considered a stakeholder
>
> will try to focus the feedback and cc as many folks
> as possible, as it seems that some feedback is lost
> on the way upstream ...
>
> best,
> Herbert
>
> > Thanks.
>

--
Thanks,
Dmitry.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by kir on Sat, 09 Dec 2006 22:34:14 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Fri, Dec 08, 2006 at 10:13:48PM -0800, Andrew Morton wrote:
>
>>
>> It's actually happening quite gradually and carefully.
>>
>
> hmm, I must have missed a testing phase for the
> IPC namespace then, not that I think it is broken
> (well, maybe it is, we do not know yet)
>
>
You have announced at LKML that Linux-VServer now uses the stuff that
was merged in 2.6.19-rc1, haven't you? I suppose that means you are
using IPC namespaces from mainstream? Isn't that considered testing? Or
you don't test Linux-VServer? Please clarify, I'm a bit lost here.

Speaking of OpenVZ, as Kirill Korotaev said before we have backported
all that to 2.6.18 back in September and are using it since then. And
yes, we found a bug in IPC namespaces, and fix from Pavel Emelyanov has
made it to 2.6.19-rc5 (see commit
c7e12b838989b0e432c7a1cdf1e6c6fd936007f6 to linux-2.6-git).

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: Re: Network virtualization/isolation
Posted by Herbert Poetzl on Sun, 10 Dec 2006 02:21:00 GMT
View Forum Message <> Reply to Message

On Sun, Dec 10, 2006 at 01:34:14AM +0300, Kir Kolyshkin wrote:
> Herbert Poetzl wrote:
> >On Fri, Dec 08, 2006 at 10:13:48PM -0800, Andrew Morton wrote:
> >
> >>
> >>It's actually happening quite gradually and carefully.
> >>
> >
> >hmm, I must have missed a testing phase for the
> >IPC namespace then, not that I think it is broken
> >(well, maybe it is, we do not know yet)

> >

> You have announced at LKML that Linux-VServer now uses the
> stuff that was merged in 2.6.19-rc1, haven't you?

yes, correct, and we already fixed several issues
the changes caused, both in handling as well as
functionality

> I suppose that means you are using IPC namespaces from
> mainstream?

yes, we do

> Isn't that considered testing?

of course it is testing, but it is already in
mainstream, and for my part, I wasn't able to
provide feedback from testing yet ...

> Or you don't test Linux-VServer?

we do the same testing you folks do IIRC
(i.e. some secret test procedure which takes
roughly a week or so, after which we can tell
that everything works as expected :)

> Please clarify, I'm a bit lost here.

> Speaking of OpenVZ, as Kirill Korotaev said before we have
> backported all that to 2.6.18 back in September

nice, but what relevance has that for 2.6.19?

> and are using it since then.

cool, how much feedback regarding IPC did you get
since then?

> And yes, we found a bug in IPC namespaces, and fix from
> Pavel Emelyanov has made it to 2.6.19-rc5 (see commit
> c7e12b838989b0e432c7a1cdf1e6c6fd936007f6 to linux-2.6-git).

it's good that some bugs have been found, but
of what relevance is that for testing mainline
patches?

- typical linux users will only excercise a
  small fraction of the new code, if at all
- virtualization solutions like OpenVZ and
  Linux-VServer add their custom modifications
  and/or adjustments, and serve a much smaller
  userbase
- I haven't seen any test suites or similar
  for the spaces

so it all boils down to waiting for somebody to
stumble over an issue, which then will get fixed
just that the number of folks testing that is
quite small compared to 'other' mainline pathes

anyway, originally I was just answering to an
email pushing for 'fast' inclusion, which I do
not consider a good idea (as I already stated)

best,
Herbert

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers