Subject: L2 network namespace benchmarking
Posted by Daniel Lezcano on Tue, 27 Mar 2007 22:16:34 GMT
View Forum Message <> Reply to Message

Hi,

I did some benchmarking on the existing L2 network namespaces.

These patches are included in the lxc patchset at:
http://lxc.sourceforge.net/patches/2.6.20
The lxc7 patchset series contains Dmitry's patchset
The lxc8 patchset series contains Eric's patchset

Here are the following scenarii I made in order to do some simple benchmarking on the network namespace. I tested three kernels:

* Vanilla kernel 2.6.20

* lxc7 with Dmitry's patchset based on 2.6.20
  * L3 network namespace has been removed to do testing

* lxc8 with Eric's patchset based on 2.6.20

I didn't do any tests on Linux-Vserver because it is L3 namespace and it is not comparable with the L2 namespace implementation. If anyone is interessted by Linux-Vserver performances, that can be found at http://lxc.sf.net. Roughly, we know there is no performance degradation.

For each kernel, several configurations were tested:

* vanilla, obviously, only one configuration was tested for reference values.

* lxc7, network namespace
 - compiled out
 - compiled in
   - without container
   - inside a container with ip_forward, route and veth
   - inside a container with a bridge and veth

* lxc8, network namespace
 - compiled out
 - compiled in
   - without container
   - inside a container with a real network device (eth1 was moved in the container instead of using an etun device)
   - inside a container with ip_forward, route and etun

- inside a container with a bridge and etun

Each benchmarking has been done with 2 machines running netperf and tbench. A dedicated machine with a RH4 kernel run the bench servers.

For each bench, netperf and tbench, the tests are ran on:

* Intel Xeon EM64T, Bi-processor 2,8GHz with hyperthreading activated, 4GB of RAM and Gigabyte NIC (tg3)

* AMD Athlon MP 1800+, Bi-processor 1,5GHz, 1GB of RAM and Gigabyte NIC (dl2000)

Each tests are run on these machines in order to have a CPU relative overhead.

bench on vanilla
==========

Netperf	CPU usage (%)	Throughput (Mbits/s)
on xeon	5.99	941.38
on athlon	28.17	844.82

Tbench	Throughput (MBytes/s)
on xeon	66.35
on athlon	65.31

bench from Dmitry's patchset
==========

1 - with net_ns compiled out

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed

on xeon	5.93 / -1 %	941.32 / 0 %
on athlon	28.89 / +2.5 %	842.78 / -0.2 %

Tbench	Throughput (MBytes/s) / changed
on xeon	67.00 / +0.9 %
on athlon	65.45 / 0 %

Observation : no noticeable overhead

2 - with net_ns compiled in

2.1 - without container

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed
on xeon	6.23 / +4 %	941.35 / 0 %
on athlon	28.83 / +2.3 %	850.76 / +0.7 %

Tbench	Throughput (MBytes/s) / changed
on xeon	67.00 / 0 %
on athlon	65.45 / 0 %

Observation : no noticeable overhead

2.2 - inside the container with veth and routes

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed

on xeon	17.14 / +186.1 %	941.34 / 0 %
on athlon	49.99 / +77.45 %	838.85 / +0.7 %

Tbench	Throughput (MBytes/s) / changed
on xeon	66.00 / -0.5 %
on athlon	61.00 / -6.65 %

Observation : CPU overhead is very big, throughput is impacted on the less powerful machine

2.3 - inside the container with veth and bridge

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed
on xeon	19.14 / +299 %	941.18 / 0 %
on athlon	49.98 / +77.42 %	831.65 / -1.5 %

Tbench	Throughput (MBytes/s) / changed
on xeon	64.00 / -3.5 %
on athlon	60.07 / -8.3 %

Observation : CPU overhead is very big, throughput is impacted on the less powerful machine

bench from Eric's patchset

1 - with net_ns compiled out

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed
on xeon	6.04 / +0.8 %	941.33 / 0 %
on athlon	28.45 / +1 %	840.76 / -0.5 %

Tbench	Throughput (MBytes/s) / changed
on xeon	65.69 / -1 %
on athlon	65.35 / -0.2 %

Observation : no noticeable overhead

2 - with net_ns compiled in

2.1 - without container

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed
on xeon	6.02 / +0.5 %	941.34 / 0 %
on athlon	27.93 / -0.8 %	833.53 / -1.3 %

Tbench	Throughput (MBytes/s) / changed
on xeon	66.00 / -0.5 %
on athlon	64.94 / -0.9 %

Observation : no noticeable overhead

2.2 - inside the container with real device

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed
on xeon	5.60 / -6.5 %	941.42 / 0 %
on athlon	27.73 / -1.5 %	835.11 / +1.5 %

Tbench	Throughput (MBytes/s) / changed
on xeon	74.36 / +12 %
on athlon	70.87 / +8.2 %

Observation : no noticeable overhead. The network interface is only used by the container, so I guess it does not interact with another network traffic and that explains the performances are better.

2.3 - inside the container with etun and routes

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed
on xeon	16.25 / +171 %	941.31 / 0 %
on athlon	49.99 / +77 %	828.94 / -1.9 %

Tbench	Throughput (MBytes/s) / changed
on xeon	65.61 / -1.1 %
on athlon	62.58 / -4.5 %

Observation : The CPU overhead is very big. Throughput is a little impacted on the less powerful machine.

2.4 - inside the container with etun and bridge

Netperf	CPU usage (%) / overhead	Throughput (Mbits/s) / changed

on xeon	18.39 / +207 %	941.30 / 0 %
on athlon	49.94 / +77 %	823.75 / -2.5 %

Tbench	Throughput (MBytes/s) / changed
on xeon	66.52 / +0.2 %
on athlon	61.07 / -6.8 %

Observation : The CPU overhead is very big. Throughput is a little impacted on the less powerful machine.

3. General observations

The objective to have no performances degrations, when the network namespace is off in the kernel, is reached in both solutions.

When the network is used outside the container and the network namespace are compiled in, there is no performance degradations.

Eric's patchset allows to move network devices between namespaces and this is clearly a good feature, missing in the Dmitry's patchset. This feature helps us to see that the network namespace code does not add overhead when using directly the physical network device into the container.

The loss of performances is very noticeable inside the container and seems to be directly related to the usage of the pair device and the specific network configuration needed for the container. When the packets are sent by the container, the mac address is for the pair device but the IP address is not owned by the host. That directly implies to have the host to act as a router and the packets to be forwarded. That adds a lot of overhead.

A hack has been made in the ip_forward function to avoid useless skb_cow when using the pair device/tunnel device and the overhead is reduced by the half.

Regards.

-- Daniel

_____

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

---

Subject: Re: L2 network namespace benchmarking
Posted by Herbert Poetzl on Tue, 27 Mar 2007 23:08:27 GMT
View Forum Message <> Reply to Message

On Wed, Mar 28, 2007 at 12:16:34AM +0200, Daniel Lezcano wrote:
>
> Hi,
>
> I did some benchmarking on the existing L2 network namespaces.
>
> These patches are included in the lxc patchset at:
> http://lxc.sourceforge.net/patches/2.6.20
> The lxc7 patchset series contains Dmitry's patchset
> The lxc8 patchset series contains Eric's patchset
>
> Here are the following scenarii I made in order to do some simple
> benchmarking on the network namespace. I tested three kernels:
>
> * Vanilla kernel 2.6.20
>
> * lxc7 with Dmitry's patchset based on 2.6.20
> * L3 network namespace has been removed to do testing
>
> * lxc8 with Eric's patchset based on 2.6.20
>
> I didn't do any tests on Linux-Vserver because it is L3 namespace and
> it is not comparable with the L2 namespace implementation. If anyone
> is interessted by Linux-Vserver performances, that can be found at
> http://lxc.sf.net. Roughly, we know there is no performance
> degradation.
>
> For each kernel, several configurations were tested:
>
> * vanilla, obviously, only one configuration was tested for reference
> values.
>
> * lxc7, network namespace
> - compiled out

> - compiled in
> - without container
> - inside a container with ip_forward, route and veth
> - inside a container with a bridge and veth
>
> * lxc8, network namespace
> - compiled out
> - compiled in
> - without container
> - inside a container with a real network device (eth1 was moved
> in the container instead of using an etun device)
> - inside a container with ip_forward, route and etun
> - inside a container with a bridge and etun
>
> Each benchmarking has been done with 2 machines running netperf and
> tbench. A dedicated machine with a RH4 kernel run the bench servers.
>
> For each bench, netperf and tbench, the tests are ran on:
>
> * Intel Xeon EM64T, Bi-processor 2,8GHz with hyperthreading
> activated, 4GB of RAM and Gigabyte NIC (tg3)
>
> * AMD Athlon MP 1800+, Bi-processor 1,5GHz, 1GB of RAM and Gigabyte
> NIC (dl2000)
>
> Each tests are run on these machines in order to have a CPU relative
> overhead.
>
>
> # bench on vanilla
> ==================
>
>
> ---------- --------------------------------------
> | Netperf | CPU usage (%) | Throughput (Mbits/s) |
> ---------- --------------------------------------
> | on xeon | 5.99 | 941.38 |
> --
> | on athlon | 28.17 | 844.82 |
> --
>
> ---------- ----------------------
> | Tbench | Throughput (MBytes/s) |
> ---------- ----------------------
> | on xeon | 66.35 |
> ---------------------------------
> | on athlon | 65.31 |
> ---------------------------------

>
>
> # bench from Dmitry's patchset
> ==============================
>
>
> 1 - with net_ns compiled out
> ----------------------------
>
> ---------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- --
> | on xeon | 5.93 / -1 % | 941.32 / 0 % |
> ---------- --
> | on athlon | 28.89 / +2.5 % | 842.78 / -0.2 % |
> ---------- --
>
> ---------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- --------------------------------
> | on xeon | 67.00 / +0.9 % |
> ---------- --------------------------------
> | on athlon | 65.45 / 0 % |
> ---------- --------------------------------
>
>   Observation : no noticeable overhead
>
>
> 2 - with net_ns compiled in
> ----------------------------
>
>
>   2.1 - without container
>   -----------------------
>
> ---------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- --
> | on xeon | 6.23 / +4 % | 941.35 / 0 % |
> ---------- --
> | on athlon | 28.83 / +2.3 % | 850.76 / +0.7 % |
> ---------- --
>
> ---------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- --------------------------------
> | on xeon | 67.00 / 0 % |
> ---------- --------------------------------

> | on athlon | 65.45 / 0 % |
> --
>
> Observation : no noticeable overhead
>
>
>   2.2 - inside the container with veth and routes
>   --
>
> ---------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- --
> | on xeon | 17.14 / +186.1 % | 941.34 / 0 % |
> ---------- --
> | on athlon | 49.99 / +77.45 % | 838.85 / +0.7 % |
> --
>
> ---------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- --------------------------------
> | on xeon | 66.00 / -0.5 % |
> --
> | on athlon | 61.00 / -6.65 % |
> --
>
> Observation : CPU overhead is very big, throughput is impacted on
> the less powerful machine
>
>
>   2.3 - inside the container with veth and bridge
>   --
>
> ---------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- --
> | on xeon | 19.14 / +299 % | 941.18 / 0 % |
> ---------- --
> | on athlon | 49.98 / +77.42 % | 831.65 / -1.5 % |
> --
>
> ---------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- --------------------------------
> | on xeon | 64.00 / -3.5 % |
> --
> | on athlon | 60.07 / -8.3 % |
> --
>

> Observation : CPU overhead is very big, throughput is impacted on
> the less powerful machine
>
>
> # bench from Eric's patchset
> ============================
>
>
> 1 - with net_ns compiled out
> ----------------------------
>
> ---------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- --
> | on xeon | 6.04 / +0.8 % | 941.33 / 0 % |
> --
> | on athlon | 28.45 / +1 % | 840.76 / -0.5 % |
> --
>
> ---------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- --------------------------------
> | on xeon | 65.69 / -1 % |
> --
> | on athlon | 65.35 / -0.2 % |
> --
>
> Observation : no noticeable overhead
>
>
> 2 - with net_ns compiled in
> ---------------------------
>
>
>   2.1 - without container
>   -----------------------
>
> ---------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- --
> | on xeon | 6.02 / +0.5 % | 941.34 / 0 % |
> --
> | on athlon | 27.93 / -0.8 % | 833.53 / -1.3 % |
> --
>
> ---------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- --------------------------------

> | on xeon | 66.00 / -0.5 % |
> ----------
> | on athlon | 64.94 / -0.9 % |
> ----------
>
> Observation : no noticeable overhead
>
>
> 2.2 - inside the container with real device
> ----------
>
> ---------- ----------
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- ----------
> | on xeon | 5.60 / -6.5 % | 941.42 / 0 % |
> ----------
> | on athlon | 27.73 / -1.5 % | 835.11 / +1.5 % |
> ----------
>
> ---------- ----------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- ----------
> | on xeon | 74.36 / +12 % |
> ----------
> | on athlon | 70.87 / +8.2 % |
> ----------
>
> Observation : no noticeable overhead. The network interface is only
> used by the container, so I guess it does not interact with another
> network traffic and that explains the performances are better.
>
>
> 2.3 - inside the container with etun and routes
> ----------
>
> ---------- ----------
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ---------- ----------
> | on xeon | 16.25 / +171 % | 941.31 / 0 % |
> ----------
> | on athlon | 49.99 / +77 % | 828.94 / -1.9 % |
> ----------
>
> ---------- ----------
> | Tbench | Throughput (MBytes/s) / changed |
> ---------- ----------
> | on xeon | 65.61 / -1.1 % |
> ----------

> | on athlon | 62.58 / -4.5 % |
> --
>
>   Observation : The CPU overhead is very big. Throughput is a little
>   impacted on the less powerful machine.
>
>
>   2.4 - inside the container with etun and bridge
>   --
>
> ----------- --
> | Netperf | CPU usage (%) / overhead | Throughput (Mbits/s) / changed |
> ----------- --
> | on xeon | 18.39 / +207 % | 941.30 / 0 % |
> --
> | on athlon | 49.94 / +77 % | 823.75 / -2.5 % |
> --
>
> ----------- --------------------------------
> | Tbench | Throughput (MBytes/s) / changed |
> ----------- --------------------------------
> | on xeon | 66.52 / +0.2 % |
> --
> | on athlon | 61.07 / -6.8 % |
> --
>
>   Observation : The CPU overhead is very big. Throughput is a little
>   impacted on the less powerful machine.
>
>
> 3. General observations
> -----------------------
>
> The objective to have no performances degrations, when the network
> namespace is off in the kernel, is reached in both solutions.
>
> When the network is used outside the container and the network
> namespace are compiled in, there is no performance degradations.
>
> Eric's patchset allows to move network devices between namespaces and
> this is clearly a good feature, missing in the Dmitry's patchset. This
> feature helps us to see that the network namespace code does not add
> overhead when using directly the physical network device into the
> container.
>
> The loss of performances is very noticeable inside the container and
> seems to be directly related to the usage of the pair device and the
> specific network configuration needed for the container. When the

> packets are sent by the container, the mac address is for the pair
> device but the IP address is not owned by the host. That directly
> implies to have the host to act as a router and the packets to be
> forwarded. That adds a lot of overhead.
>
> A hack has been made in the ip_forward function to avoid useless
> skb_cow when using the pair device/tunnel device and the overhead
> is reduced by the half.

would it be possible to do some tests regarding scalability?

i.e. I would be interested how the following would look like:

 10 connections on a single host (in parallel, overall performance)
 10 connections from the same net space
 10 connections from 10 different net spaces
    (i.e. one connection from each space)

we can assume that L3 isolation will give similar results to
the first case, but if needed, we can provide a patch to
test this too ...

TIA,
Herbert

PS: great work! tx!

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

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

---

Subject: Re: L2 network namespace benchmarking
Posted by Daniel Lezcano on Wed, 28 Mar 2007 07:07:56 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Wed, Mar 28, 2007 at 12:16:34AM +0200, Daniel Lezcano wrote:

>> Hi,

[cut]

>> 3. General observations
>> ----------------------
>>
>> The objective to have no performances degrations, when the network
>> namespace is off in the kernel, is reached in both solutions.
>>
>> When the network is used outside the container and the network
>> namespace are compiled in, there is no performance degradations.
>>
>> Eric's patchset allows to move network devices between namespaces and
>> this is clearly a good feature, missing in the Dmitry's patchset. This
>> feature helps us to see that the network namespace code does not add
>> overhead when using directly the physical network device into the
>> container.
>>
>> The loss of performances is very noticeable inside the container and
>> seems to be directly related to the usage of the pair device and the
>> specific network configuration needed for the container. When the
>> packets are sent by the container, the mac address is for the pair
>> device but the IP address is not owned by the host. That directly
>> implies to have the host to act as a router and the packets to be
>> forwarded. That adds a lot of overhead.
>>
>> A hack has been made in the ip_forward function to avoid useless
>> skb_cow when using the pair device/tunnel device and the overhead
>> is reduced by the half.
>
> would it be possible to do some tests regarding scalability?
>
> i.e. I would be interested how the following would look like:
>
> 10 connections on a single host (in parallel, overall performance)
> 10 connections from the same net space
> 10 connections from 10 different net spaces
>   (i.e. one connection from each space)
>
> we can assume that L3 isolation will give similar results to
> the first case, but if needed, we can provide a patch to
> test this too ...
>

Ok. Assuming, Eric's and Dmitry's patchset are very similar, I will
focus on the Eric's patchset because it is more mature and more easy to
setup. I will have a look on the bridge optimization before doing that.

>
> PS: great work! tx!
>

Thanks.

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

---

Subject: Re: L2 network namespace benchmarking
Posted by ebiederm on Wed, 28 Mar 2007 11:52:31 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <daniel.lezcano@free.fr> writes:

> Eric W. Biederman wrote:
>> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>>
>>> 3. General observations
>>> ----------------------
>>>
>>> The objective to have no performances degrations, when the network
>>> namespace is off in the kernel, is reached in both solutions.
>>>
>>> When the network is used outside the container and the network
>>> namespace are compiled in, there is no performance degradations.
>>>
>>> Eric's patchset allows to move network devices between namespaces and
>>> this is clearly a good feature, missing in the Dmitry's patchset. This
>>> feature helps us to see that the network namespace code does not add
>>> overhead when using directly the physical network device into the
>>> container.
>>
>> Assuming these results are not contradicted this says that the extra
>> dereference where we need it does not add measurable to the overhead
>> in the Linus network stack. Performance wise this should be good
>> enough to allow merging the code into the linux kernel, as it does
>> not measurably affect networking when we do not have multiple
>> containers in use.
>
> I have a few questions about merging code into the linux kernel.
>
> * How do you plan to do that ?
One small comprehensible piece at a time.

Basically some variant of etun should not be a problem to merge
then I have to get some part of the network namespace code merged,
and the concept accepted.

Once the basic acceptance occurs it just becomes a long slog of
merging more and more patches.

> * When do you expect to have the network namespace into mainline ?
My current goal is to finish my rebase against 2.6.linus_lastest in
the next couple of days after having figured out how to deal with sysfs.

I have been doing reviewing in more code then I know what to do with,
and fighting some very strange bugs during the stabilization window.
Which has kept me from doing additional development. Plus I have
had a cold.

> * Are Dave Miller and Alexey Kuznetov aware of the network namespace ?
Aware yes, reviewed not yet. I believe Alexey is a little more
familiar with the OpenVZ work. The high level concepts still apply.

> * Did they saw your patchset or ever know it exists ?
Yes.

> * Do you have any feedbacks from netdev about the network namespace ?
Not really. Except that Dave Miller wanted to review what I posted
last time but the timing was bad and he failed to get around to it.

>> To be fully satisfactory how we get the packets to the namespace
>> still appears to need work.
>>
>> We have overhead in routing. That may simply be the cost of
>> performing routing or there may be some optimizations opportunities
>> there.
>> We have about the same overhead when performing bridging which I
>> actually find more surprising, as the bridging code should involve
>> less packet handling.
>
> Yep. I will try to figure out what is happening.

Thanks.

>> Ideally we can optimize the bridge code or something equivalent to
>> it so that we can take one look at the destination mac address and
>> know which network namespace we should be in. Potentially moving this
>> work to hardware when the hardware supports multiple queues.
>>
>> If we can get the overhead out of the routing code that would be

>> tremendous. However I think it may be more realistic to get the
>> overhead out of the ethernet bridging code where we know we don't need
>> to modify the packet.
>
> The routing was optimized for the loopback, no ? Why can't we do the same for
> the etun device ?

I have no problem with it if we can use valid optimizations. Avoiding a
packet copy when the packet is marked as having a second copy somewhere
else does not sound like a valid optimization to me.

Routing through both network namespaces so that we can set up a dst
cache entry that takes you to the final destination I am will to
working with. Perhaps something that hits this piece of the etun driver,
so we don't have to make a second set of routing decisions.

```
 if (skb->dst)
  skb->dst = dst_pop(skb->dst); /* Allow for smart routing */
```

tcpdump at any phase of the process should be able to do the right thing.

Mostly I care right now in that it is interesting to know where the
performance overhead is coming from. Unless it is something of a
merge stopper I don't much care about how we are going to fix it yet,
especially if it is only cross network namespace traffic.

If I read the results right it took a 32bit machine from AMD with
a gigabit interface before you could measure a throughput difference.
That isn't shabby for a non-optimized code path.

Eric

Subject: Re: L2 network namespace benchmarking
Posted by ebiederm on Wed, 28 Mar 2007 12:06:19 GMT
View Forum Message <> Reply to Message

Kirill Korotaev <dev@sw.ru> writes:

>> Ideally we can optimize the bridge code or something equivalent to
>> it so that we can take one look at the destination mac address and
>> know which network namespace we should be in. Potentially moving this
>> work to hardware when the hardware supports multiple queues.

> yes, we can hack the bridge, so that packets coming out of eth devices
> can go directly to the container and get out of veth devices from
> inside the container.
>
>> If we can get the overhead out of the routing code that would be
>> tremendous. However I think it may be more realistic to get the
>> overhead out of the ethernet bridging code where we know we don't need
>> to modify the packet.
> Why not optimize both? :)

If the optimizations are safe and correct I don't have a problem.

When we seem to have multiple copies of a packet in circulation and
we skip a what appears to be a required copy on write, I'm dubious.

Although the more I look at suggested optimization the less dubious I
am as it appears all we are skipping is a ttl decrement and the cow
flag exclusively applies to the data chunk and not the header chunk of
the packet whatever that means.

However we still need to guard against a loop in our routing table
setup between multiple guests.

Eric

Subject: Re: L2 network namespace benchmarking
Posted by Rick Jones on Wed, 28 Mar 2007 18:08:28 GMT
View Forum Message <> Reply to Message

> If I read the results right it took a 32bit machine from AMD with
> a gigabit interface before you could measure a throughput difference.
> That isn't shabby for a non-optimized code path.

Just some paranoid ramblings - one needs to look beyond just whether or
not the performance of a bulk transfer test (eg TCP_STREAM) remains able
to hit link-rate. One has to also consider the change in service demand
(the normalization of CPU util and throughput). Also, with
functionality like TSO in place, the ability to pass very large things
down the stack can help cover for a multitude of path-length sins. And
with either multiple 1G or 10G NICs becoming more and more prevalent, we
have another one of those "NIC speed vs CPU speed" switch-overs, so
maintaining single-NIC 1 gigabit throughput, while necessary, isn't
(IMO) sufficient.

Soooo, it becomes very important to go beyond just TCP_STREAM tests when evaluating these sorts of things. Another test to run would be the TCP_RR test. TCP_RR with single-byte request/response sizes will "bypass" the TSO stuff, and the transaction rate will be more directly affected by the change in path length than a TCP_STREAM test. It will also show-up quite clearly in the service demand. Now, with NICs doing interrupt coalescing, if the NIC is strapped "poorly" (IMO) then you may not see a change in transaction rate - it may be getting limited artifically by the NIC's interrupt coalescing. So, one has to fall-back on service demand, or better yet, disable the interrupt coalescing.

Otherwise, measuring peak aggregate request/response becomes necessary.

rick jones
don't be blinded by bit-rate

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

---

Subject: Re: L2 network namespace benchmarking
Posted by Daniel Lezcano on Wed, 28 Mar 2007 19:47:54 GMT
View Forum Message <> Reply to Message

Rick Jones wrote:
>> If I read the results right it took a 32bit machine from AMD with
>> a gigabit interface before you could measure a throughput difference.
>> That isn't shabby for a non-optimized code path.
>
> Just some paranoid ramblings - one needs to look beyond just whether
> or not the performance of a bulk transfer test (eg TCP_STREAM) remains
> able to hit link-rate. One has to also consider the change in service
> demand (the normalization of CPU util and throughput). Also, with
> functionality like TSO in place, the ability to pass very large things
> down the stack can help cover for a multitude of path-length sins.
> And with either multiple 1G or 10G NICs becoming more and more
> prevalent, we have another one of those "NIC speed vs CPU speed"
> switch-overs, so maintaining single-NIC 1 gigabit throughput, while
> necessary, isn't (IMO) sufficient.
>
> Soooo, it becomes very important to go beyond just TCP_STREAM tests
> when evaluating these sorts of things. Another test to run would be
> the TCP_RR test. TCP_RR with single-byte request/response sizes will
> "bypass" the TSO stuff, and the transaction rate will be more directly
> affected by the change in path length than a TCP_STREAM test. It will

> also show-up quite clearly in the service demand. Now, with NICs
> doing interrupt coalescing, if the NIC is strapped "poorly" (IMO) then
> you may not see a change in transaction rate - it may be getting
> limited artifically by the NIC's interrupt coalescing. So, one has to
> fall-back on service demand, or better yet, disable the interrupt
> coalescing.
>
> Otherwise, measuring peak aggregate request/response becomes necessary.
>
>
> rick jones
> don't be blinded by bit-rate
Thanks Rick,

Do you have any pointer to help on benchmarking the network, perhaps a
checklist or some scripts for netperf ?

Regards.
   -- Daniel

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

---

Subject: Re: L2 network namespace benchmarking
Posted by Rick Jones on Wed, 28 Mar 2007 20:12:03 GMT
View Forum Message <> Reply to Message

> Do you have any pointer to help on benchmarking the network, perhaps a
> checklist or some scripts for netperf ?

There are some scripts in doc/examples but they are probably a bit long
in the tooth by now.

The main writeup _I_ have on netperf would be the manual, which was
recently updated for the 2.4.3 release.

http://www.netperf.org/svn/netperf2/tags/netperf-2.4.3/doc/netperf.html

or the current top of trunk:

http://www.netperf.org/svn/netperf2/trunk/doc/netperf.html

There is also a netperf-talk@netperf.org mailing list which one can join
and have discussions about netperf, and a netperf-dev@netperf.org if one
wants to discuss actual netperf (netperf2 or netperf4) development.

rick jones

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

---

Subject: Re: L2 network namespace benchmarking
Posted by Benjamin Thery on Thu, 29 Mar 2007 07:37:58 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <daniel.lezcano@free.fr> writes:

[...]

>> * When do you expect to have the network namespace into mainline ?
> My current goal is to finish my rebase against 2.6.linus_lastest in
> the next couple of days after having figured out how to deal with sysfs.

Great news!
I also have some questions about this updated version:

- Have you integrated the bug fixes and cleanups(*) Daniel wrote for
your previous netns patchset (and the few glitches I reported too)?

(*) available in LXC8 patchset

- Do you already have a public git repository set up for the rebase?
- If it is private, any plan to make it public soon? (That would be great)

> I have been doing reviewing in more code then I know what to do with,
> and fighting some very strange bugs during the stabilization window.
> Which has kept me from doing additional development. Plus I have
> had a cold.

I hope you're getting better... and you'll be able to provide us the
updated patchset very soon :)

[...]

> If I read the results right it took a 32bit machine from AMD with
> a gigabit interface before you could measure a throughput difference.
> That isn't shabby for a non-optimized code path.

Indeed the throughput difference is not significant.
This is very good to see that it stays constant when using the container.
What I'm more worried about is the CPU load increase. But it seems

we've identified some of the culprits.

This afternoon I had a look at why the bridge setup isn't better than
the route setup (section 2.3 and 2.4 of Daniel's report).

In the bridge case, we encounter the same problems as the routes case.
  The oprofile profile is the same: the most demanding routines are
pskb_expand_head and csum_partial_copy_generic.
pskb_expand_head() is also called by skb_cow(), but this time
skb_cow() is called by netfilter's nf_bridge_copy_header().

We can avoid this copy by removing option CONFIG_BRIDGE_NETFILTER.
This copy is made even if netfilter is not used on the host.
Maybe some optimizations can be made in netfilter's code to prevent this.

Regards,
Benjamin

--
B e n j a m i n T h e r y - BULL/DT/Open Software R&D

   http://www.bull.com

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

---

Subject: Re: L2 network namespace benchmarking
Posted by ebiederm on Thu, 29 Mar 2007 13:01:20 GMT
View Forum Message <> Reply to Message

Benjamin Thery <benjamin.thery@bull.net> writes:

> Eric W. Biederman wrote:
>> Daniel Lezcano <daniel.lezcano@free.fr> writes:
>
> [...]
>
>>> * When do you expect to have the network namespace into mainline ?
>> My current goal is to finish my rebase against 2.6.linus_lastest in
>> the next couple of days after having figured out how to deal with sysfs.
>
> Great news!
> I also have some questions about this updated version:
>

> - Have you integrated the bug fixes and cleanups(*) Daniel wrote for
> your previous netns patchset (and the few glitches I reported too)?

About half of them so far. It is my intention to incorporate all of them.
They weren't all trivial to include. A couple of Daniel's patches
address a real issue in the wrong way so I have to give them some more
thought.

> (*) available in LXC8 patchset
>
> - Do you already have a public git repository set up for the rebase?
> - If it is private, any plan to make it public soon? (That would be great)
Yes. Where the current one is now.

>> I have been doing reviewing in more code then I know what to do with,
>> and fighting some very strange bugs during the stabilization window.
>> Which has kept me from doing additional development. Plus I have
>> had a cold.
>
> I hope you're getting better... and you'll be able to provide us the
> updated patchset very soon :)

Hopefully. I think I have fixed my last non network regression I know
about for 2.6.21-rcX. Which means I can begin to focus again.

> [...]
>
>> If I read the results right it took a 32bit machine from AMD with
>> a gigabit interface before you could measure a throughput difference.
>> That isn't shabby for a non-optimized code path.
>
> Indeed the throughput difference is not significant.
> This is very good to see that it stays constant when using the container.
> What I'm more worried about is the CPU load increase. But it seems
> we've identified some of the culprits.

Yes, and the good news is that they all seem to be in getting the
packets to the network namespace.

> This afternoon I had a look at why the bridge setup isn't better than
> the route setup (section 2.3 and 2.4 of Daniel's report).
>
> In the bridge case, we encounter the same problems as the routes case.
> The oprofile profile is the same: the most demanding routines are
> pskb_expand_head and csum_partial_copy_generic.
> pskb_expand_head() is also called by skb_cow(), but this time
> skb_cow() is called by netfilter's nf_bridge_copy_header().
>

> We can avoid this copy by removing option CONFIG_BRIDGE_NETFILTER.
> This copy is made even if netfilter is not used on the host.
> Maybe some optimizations can be made in netfilter's code to prevent this.

Sounds reasonable. I guess the next step is to get some numbers with
CONFIG_BRIDGE_NETFILTER disabled. (So we don't hit that case and just
in case there are more). I suspect the bridging code has a small
enough user base right now it just hasn't been optimized much.

Eric

_____

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