Subject: L2 network namespace benchmarking (resend with Service Demand)
Posted by Daniel Lezcano on Fri, 30 Mar 2007 14:16:28 GMT
View Forum Message <> Reply to Message

Hi,

as suggested Rick, I added the Service Demand results to the matrix.

Cheers.

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)	SD (us/KB)
on xeon	5.99	941.38	2.084
on athlon	28.17	844.82	5.462

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 SD (us/KB)
on xeon	5.93 / -1 %	941.32 / 0 % 2.066
on athlon	28.89 / +2.5 %	842.78 / -0.2 % 5.615

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 SD (us/KB)
on xeon	6.23 / +4 %	941.35 / 0 % 2.168

| on athlon | 28.83 / +2.3 % | 850.76 / +0.7 %
| 5.552 |

 ---------- ---
| 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
| SD (us/KB) |
 ---------- ---

| on xeon | 17.14 / +186.1 % | 941.34 / 0 %
| 5.966 |

| on athlon | 49.99 / +77.45 % | 838.85 / +0.7 %
| 9.763 |

 ---------- ---
| 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
|  SD (us/KB) |
 ----------- ------------------------------------------------------------
-------------
| on xeon   |     19.14 / +299 %      |       941.18  / 0 %
|    6.863    |
 -------------------------------------------------------------------------
-------------
| on athlon |     49.98 / +77.42 %    |       831.65 / -1.5 %
|    9.846    |
 ---------------------------------------------------------------------------------------



 ----------- --------------------------------
| 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
|  SD (us/KB) |
 ----------- ------------------------------------------------------------
-------------
| on xeon   |     6.04 / +0.8 %       |       941.33  / 0 %
|    2.104    |
 -------------------------------------------------------------------------
-------------
| on athlon |     28.45 / +1 %        |       840.76 /  -0.5 %
|    5.545    |
 ---------------------------------------------------------------------------------------
```

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	SD (us/KB)
on xeon	6.02 / +0.5 %	941.34 / 0 %	2.097
on athlon	27.93 / -0.8 %	833.53 / -1.3 %	5.490

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	SD (us/KB)
on xeon	5.60 / -6.5 %	941.42 / 0 %	1.949
on athlon	27.73 / -1.5 %	835.11 / +1.5 %	5.440

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	SD (us/KB)
on xeon	16.25 / +171 %	941.31 / 0 %	5.657
on athlon	49.99 / +77 %	828.94 / -1.9 %	9.880

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	SD (us/KB)
on xeon	18.39 / +207 %	941.30 / 0 %	6.400
on athlon	49.94 / +77 %	823.75 / -2.5 %	9.933

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 (resend with Service Demand)
Posted by ebiederm on Fri, 06 Apr 2007 08:03:47 GMT
View Forum Message <> Reply to Message

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

> Hi,
>
> as suggested Rick, I added the Service Demand results to the matrix.

A couple of random thoughts in trying to understand the numbers you are seeing.

- Checksum offloading?

  You have noted that with the bridge netfilter support disabled you are still seeing additional checksum overhead. Just like you are

seeing in the routing case.

Is it possible the problem is simply that etun doesn't support checksum offloading, while your normal test hardware does?

- Tagged VLANs?

Currently you have tested bridging and routing to get the packets to a network namespace. Could you test tagged vlans?

I'm just curious if we have anything in the network stack today that will multiplex a NIC without measurable overhead.

- Without NETNS?

We should probably see if we can setup the same configuration we are testing without network namespaces (just multiple interfaces on the same machine) and see if we can still measure the same overhead. Just to confirm the overhead is not a network namespace related thing.

I know we can configure the same case with bridging and I am fairly confident that we will see the same overhead without network namespaces.

Of the top of my head I am insufficiently clever to think how we could configure the routing case without network namespaces, although we might be able to force it and if so it would be interesting to measure.

I will work to get the etun setup races fixed and to fix whatever obvious feature deficiencies it has (like no configurable MTU support) and see if I can get that pushed upstream. That should make it easier for other people to reproduce what we are seeing.

Eric

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

---

Subject: Re: L2 network namespace benchmarking (resend with Service Demand)
Posted by Benjamin Thery on Fri, 06 Apr 2007 11:19:36 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:

>
>> Hi,
>>
>> as suggested Rick, I added the Service Demand results to the matrix.
>
> A couple of random thoughts in trying to understand the numbers you are
> seeing.
>
> - Checksum offloading?
>
> You have noted that with the bridge netfilter support disabled you
> are still seeing additional checksum overhead. Just like you are
> seeing in the routing case.
>
> Is it possible the problem is simply that etun doesn't support
> checksum offloading, while your normal test hardware does?

Looks like you are 100% correct.
I feel a bit stupid I didn't think about this "small" difference
between real NIC and etun.

If I turn off checksum offloading on my physical NIC, the checksum
"overhead" (load) measured by oprofile is about the same in both case:
when running netperf through a real NIC or through an etun tunnel first.

Benjamin

> - Tagged VLANs?
>
> Currently you have tested bridging and routing to get the packets to
> a network namespace. Could you test tagged vlans?
>
> I'm just curious if we have anything in the network stack today that
> will multiplex a NIC without measurable overhead.
>
> - Without NETNS?
>
> We should probably see if we can setup the same configuration we are
> testing without network namespaces (just multiple interfaces on the
> same machine) and see if we can still measure the same overhead.
> Just to confirm the overhead is not a network namespace related
> thing.
>
> I know we can configure the same case with bridging and I am fairly
> confident that we will see the same overhead without network
> namespaces.
>
> Of the top of my head I am insufficiently clever to think how we

> could configure the routing case without network namespaces,
> although we might be able to force it and if so it would be
> interesting to measure.
>
> I will work to get the etun setup races fixed and to fix whatever
> obvious feature deficiencies it has (like no configurable MTU support)
> and see if I can get that pushed upstream. That should make it easier
> for other people to reproduce what we are seeing.
>
> Eric
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

--
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 (resend with Service Demand)
Posted by ebiederm on Fri, 06 Apr 2007 14:25:48 GMT
View Forum Message <> Reply to Message

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

> Eric W. Biederman wrote:

>> A couple of random thoughts in trying to understand the numbers you are
>> seeing.
>>
>> - Checksum offloading?
>>
>> You have noted that with the bridge netfilter support disabled you
>> are still seeing additional checksum overhead. Just like you are
>> seeing in the routing case.
>>
>> Is it possible the problem is simply that etun doesn't support
>> checksum offloading, while your normal test hardware does?
>
> Looks like you are 100% correct.

> I feel a bit stupid I didn't think about this "small" difference between real
> NIC and etun.
>
> If I turn off checksum offloading on my physical NIC, the checksum "overhead"
> (load) measured by oprofile is about the same in both case: when running netperf
> through a real NIC or through an etun tunnel first.

Interesting. You can also 'enable' checksum offloading when using etun with
ethtool. Which should just tell the kernel not to do checksumming. A
bad idea in general but it might be useful in confirming where the
performance overhead is coming from, and when used with routing I
believe it is safe. When used with bridging I don't know.

Thinking about it the ideal situation is to preserve skb->ip_summed it
if came from another device, instead of unconditionally setting it.
I need to take a good hard look at etun_xmit and make certain we
are dotting all of the i's and crossing all of the t's for best
performance and compatibility with the rest of the network stack.

Eric

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