Test neutron network performance locally

Table of Contents

For a long time I want to do testing about the performance of the networks created by neutron. In particularly, when I see the *nine devices that the virtual machine's traffic must pass described by Neutron administrator guide, I have a stronger motivation to do it. This post records the whole tes ting process I have done.

In the front parts of this post, Two virtual machines are launched with 'virsh' mannually. They are attached to Linux bridges, OpenvSwitch bridges or hybrid bridges. The hybrid way is to simulate the nova's LibvirtHybridOVSBridgeDriver vif driver. After that, I test the LibvirtHybridOVSBridgeDriver driver with devstack. And then I come up with a new vif driver to avoid the performance of the LibvirtHybridOVSBridgeDriver.

The testing process is organized into seven different cases. Each test case has its own network fabric. I test two kind of traffics in each test case. One is from host to virtual machine, the other is from a virtual machine to the other virtual machine. The testing is done with iperf tool, and the data is collected. The result is shown at the end of this post.

Linux bridge

In this network fabric, the virtual machines are attached to one Linux bridge:

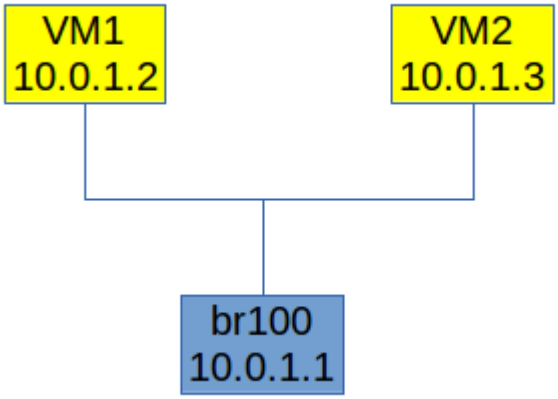
```
$ sudo brctl show
bridge name bridge id           STP enabled interfaces
br100       8000.fe5400a04c09   no          vnet0

vnet1
```

The virtual machine's interface definition in its domain file is like:

```
<interface type='bridge'>
        <mac address='52:54:00:a0:4c:09'/>
        <source bridge='br100'/>
        <model type='virtio'/>
        <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
    </interface>
```

The network fabric is shown as following figure:



Host to virtual machine

Start iperf as server on virtual machine VM1 and wait for connection, the iperf will print the bandwidth after testing:

```
# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.1.2 port 5001 connected with 10.0.1.1 port 58596
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  95.6 GBytes  27.4 Gbits/sec
```

Start iperf as client to connect to virtual machine one. the iperf will run for 30 seconds to push data to server and print the bandwidth:

```
# iperf -c 10.0.1.2 -i 2 -t 30
------------------------------------------------------------
Client connecting to 10.0.1.2, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.1 port 58596 connected with 10.0.1.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  6.00 GBytes  25.8 Gbits/sec
[  3]  2.0- 4.0 sec  6.24 GBytes  26.8 Gbits/sec
[  3]  4.0- 6.0 sec  6.85 GBytes  29.4 Gbits/sec
[  3]  6.0- 8.0 sec  6.69 GBytes  28.7 Gbits/sec
```

[3] 8.0-10.0 sec 6.24 GBytes 26.8 Gbits/sec
[3] 10.0-12.0 sec 6.78 GBytes 29.1 Gbits/sec

VM1 to VM2

vm1# iperf -c 10.0.1.3 -i 2 -t 30

Client connecting to 10.0.1.3, TCP port 5001

TCP window size: 23.5 KByte (default)

---

[3] local 10.0.1.2 port 40280 connected with 10.0.1.3 port 5001

[ID] Interval Transfer Bandwidth

[3] 0.0- 2.0 sec 5.62 GBytes 24.1 Gbits/sec

[3] 2.0- 4.0 sec 5.51 GBytes 23.7 Gbits/sec

[3] 4.0- 6.0 sec 5.89 GBytes 25.3 Gbits/sec

[3] 6.0- 8.0 sec 6.12 GBytes 26.3 Gbits/sec

vm2#iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.2 port 40280

[ID] Interval Transfer Bandwidth

[4] 0.0-30.0 sec 83.6 GBytes 23.9 Gbits/sec

OVS bridge with Linux bridge connected via veth

In this network fabric, the virtual machines are attached to one Linux bridge irrespectively:

```
$ sudo brctl show
bridge name bridge id           STP enabled interfaces
br1         8000.e2e3ea921a4b   no          qvb1
                                            vnet0
br2         8000.daede0593691   no          qvb2
                                            vnet1
```

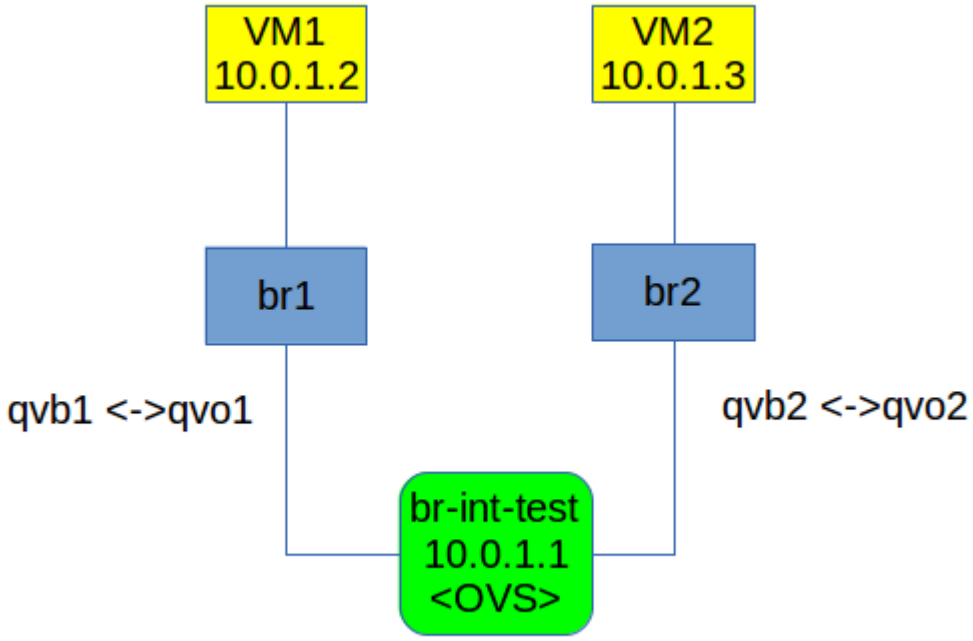
And then the Linux bridges are connected to a OVS bridge via veth devices qvb1 ↔ qvo1 and qvb2 ↔ qvo2:

```
$ sudo ovs-vsctl show
e5e2f9b3-a938-48cf-8754-6c70f7e251bc
    Bridge br-int-test
        Port "qvo2"
            Interface "qvo2"
        Port br-int-test
            Interface br-int-test
                type: internal
        Port "qvo1"
            Interface "qvo1"
```

The virtual machine's interface definition in its domain file is like:

```
    <interface type='bridge'>
      <mac address='52:54:00:a0:4c:09'/>
      <source bridge='br1'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
    </interface>
```

The network fabric is like this:



Host to virtual machine

Start iperf server on virtual machine and wait for connection:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.1 port 38999

[4] 0.0-30.0 sec 3.94 GBytes 1.13 Gbits/sec

Start the iperf client on host:

$ iperf -c 10.0.1.3 -i 2 -t 30

Client connecting to 10.0.1.3, TCP port 5001

TCP window size: 22.9 KByte (default)

---

[3] local 10.0.1.1 port 38999 connected with 10.0.1.3 port 5001

[ID] Interval Transfer Bandwidth

[3] 0.0- 2.0 sec 286 MBytes 1.20 Gbits/sec

[3] 2.0- 4.0 sec 247 MBytes 1.03 Gbits/sec

[3] 4.0- 6.0 sec 253 MBytes 1.06 Gbits/sec

[3] 6.0- 8.0 sec 238 MBytes 1.00 Gbits/sec

VM1 to VM2:

Start iperf server on VM2:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.2 port 58664

[ID] Interval Transfer Bandwidth

[4] 0.0-30.0 sec 12.0 GBytes 3.43 Gbits/sec

Start iperf client on VM2:

iperf -c 10.0.1.3 -i 2 -t 30

Client connecting to 10.0.1.3, TCP port 5001

TCP window size: 23.5 KByte (default)

---

[3] local 10.0.1.2 port 58664 connected with 10.0.1.3 port 5001

[ID] Interval Transfer Bandwidth

[3] 0.0- 2.0 sec 662 MBytes 2.78 Gbits/sec

[3] 2.0- 4.0 sec 809 MBytes 3.39 Gbits/sec

[3] 4.0- 6.0 sec 792 MBytes 3.32 Gbits/sec

[3] 6.0- 8.0 sec 807 MBytes 3.38 Gbits/sec

[3] 8.0-10.0 sec 833 MBytes 3.50 Gbits/sec

[3] 10.0-12.0 sec 817 MBytes 3.43 Gbits/sec

[3] 12.0-14.0 sec 836 MBytes 3.51 Gbits/sec

[3] 14.0-16.0 sec 906 MBytes 3.80 Gbits/sec

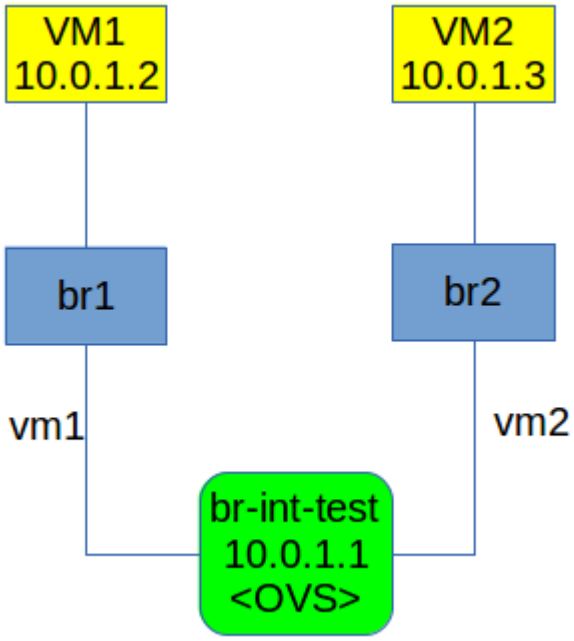
OVS bridge with Linux bridge connected via OVS internal port

This fabric is like previous one. However, the Linux bridge is connected to OVS bridge via OVS internal port, instead of the veth devices.

```
# ovs-vsctl add-port br-int-test vm2 -- set interface vm2 type=internal
# ovs-vsctl add-port br-int-test vm1 -- set interface vm1 type=internal
# ovs-vsctl show
e5e2f9b3-a938-48cf-8754-6c70f7e251bc
    Bridge br-int-test
        Port br-int-test
            Interface br-int-test
                type: internal
        Port "vm1"
            Interface "vm1"
                type: internal
        Port "vm2"
            Interface "vm2"
                type: internal
    ovs_version: "1.11.0"
# brctl show
bridge name bridge id           STP enabled interfaces
br1         8000.62ad95cf4b15   no          vm1
                                            vnet0
br2         8000.fe5400f8fa26   no          vm2
```

vnet1

The fabric is like:



Host to virtual machine

Start iperf server on virtual machine VM2:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.1 port 42314

[ID] Interval Transfer Bandwidth

[4] 0.0-30.0 sec 91.1 GBytes 26.1 Gbits/sec

and start iperf client on host:

$ iperf -c 10.0.1.3 -i 2 -t 30

```
------------------------------------------------------------
Client connecting to 10.0.1.3, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.1 port 42314 connected with 10.0.1.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  6.28 GBytes  27.0 Gbits/sec
[  3]  2.0- 4.0 sec  5.68 GBytes  24.4 Gbits/sec
[  3]  4.0- 6.0 sec  6.44 GBytes  27.7 Gbits/sec
[  3]  6.0- 8.0 sec  5.51 GBytes  23.7 Gbits/sec
[  3]  8.0-10.0 sec  5.94 GBytes  25.5 Gbits/sec
[  3] 10.0-12.0 sec  5.76 GBytes  24.7 Gbits/sec
[  3] 12.0-14.0 sec  6.25 GBytes  26.8 Gbits/sec
[  3] 14.0-16.0 sec  6.13 GBytes  26.3 Gbits/sec
[  3] 16.0-18.0 sec  6.25 GBytes  26.9 Gbits/sec
[  3] 18.0-20.0 sec  6.53 GBytes  28.0 Gbits/sec
```

VM1 to VM2

Start iperf server on VM2:

```
# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.1.3 port 5001 connected with 10.0.1.2 port 44556
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  74.0 GBytes  21.2 Gbits/sec
```

Start iperf client on VM1:

```
# iperf -c 10.0.1.3 -i 2 -t 30
------------------------------------------------------------
Client connecting to 10.0.1.3, TCP port 5001
TCP window size: 23.5 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.2 port 44556 connected with 10.0.1.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  4.73 GBytes  20.3 Gbits/sec
[  3]  2.0- 4.0 sec  4.76 GBytes  20.5 Gbits/sec
[  3]  4.0- 6.0 sec  4.79 GBytes  20.6 Gbits/sec
[  3]  6.0- 8.0 sec  4.64 GBytes  19.9 Gbits/sec
[  3]  8.0-10.0 sec  5.03 GBytes  21.6 Gbits/sec
[  3] 10.0-12.0 sec  4.93 GBytes  21.2 Gbits/sec
[  3] 12.0-14.0 sec  5.08 GBytes  21.8 Gbits/sec
[  3] 14.0-16.0 sec  4.80 GBytes  20.6 Gbits/sec
[  3] 16.0-18.0 sec  4.86 GBytes  20.9 Gbits/sec
```

OVS bridge

In this fabric, the virtual machines are connected to OVS bridge directly. The interface definition of VMs' domain files is like:

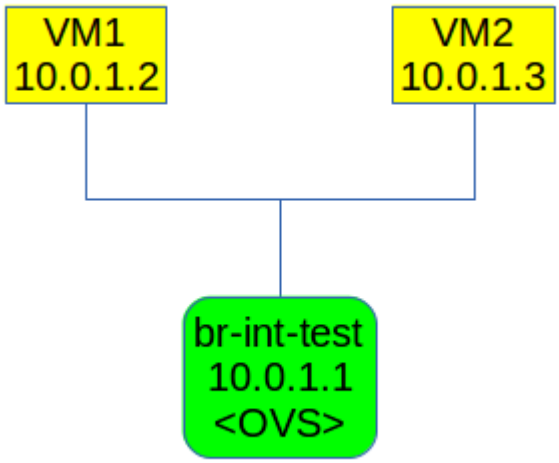
```
 <interface type='bridge'>
        <mac address='52:54:00:a0:4c:09'/>
        <source bridge='br-int-test'/>
        <virtualport type='openvswitch'>
          <parameters interfaceid='6a7bbe68-f4fe-f8fb-cf09-f2bb933e005f'/>
        </virtualport>
        <target dev='vnet0'/>
        <model type='virtio'/>
        <alias name='net0'/>
        <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
```

```
    </interface>
```

When the virtual machines are started, we have an OVS bridge like the following:

```
# ovs-vsctl show
e5e2f9b3-a938-48cf-8754-6c70f7e251bc
    Bridge "br-int-test"
        Port "vnet0"
            Interface "vnet0"
        Port "vnet1"
            Interface "vnet1"
        Port "br-int-test"
            Interface "br-int-test"
                type: internal
ovs_version: "1.11.0"
```

They are connected like the following figure:



Host to virtual machine

On VM2, we run:

```
# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.1.2 port 5001 connected with 10.0.1.1 port 58905
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec   102 GBytes  29.1 Gbits/sec
```

On host, we run:

```
$ iperf -c 10.0.1.2 -i 2 -t 30
------------------------------------------------------------
Client connecting to 10.0.1.2, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.1 port 58905 connected with 10.0.1.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  6.90 GBytes  29.6 Gbits/sec
[  3]  2.0- 4.0 sec  6.46 GBytes  27.8 Gbits/sec
[  3]  4.0- 6.0 sec  6.74 GBytes  29.0 Gbits/sec
[  3]  6.0- 8.0 sec  7.18 GBytes  30.8 Gbits/sec
[  3]  8.0-10.0 sec  6.61 GBytes  28.4 Gbits/sec
[  3] 10.0-12.0 sec  7.13 GBytes  30.6 Gbits/sec
[  3] 12.0-14.0 sec  6.81 GBytes  29.2 Gbits/sec
[  3] 14.0-16.0 sec  5.95 GBytes  25.5 Gbits/sec
[  3] 16.0-18.0 sec  7.03 GBytes  30.2 Gbits/sec
[  3] 18.0-20.0 sec  7.14 GBytes  30.7 Gbits/sec
[  3] 20.0-22.0 sec  6.94 GBytes  29.8 Gbits/sec
[  3] 22.0-24.0 sec  7.12 GBytes  30.6 Gbits/sec
```

VM1 to VM2

On VM2:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.2 port 34248

[ID] Interval Transfer Bandwidth

[4] 0.0-30.0 sec 80.7 GBytes 23.1 Gbits/sec

On VM1:

iperf -c 10.0.1.3 -i 2 -t 30

Client connecting to 10.0.1.3, TCP port 5001

TCP window size: 23.5 KByte (default)

---

[3] local 10.0.1.2 port 34248 connected with 10.0.1.3 port 5001

[ID] Interval Transfer Bandwidth

[3] 0.0- 2.0 sec 5.37 GBytes 23.1 Gbits/sec

[3] 2.0- 4.0 sec 5.87 GBytes 25.2 Gbits/sec

[3] 4.0- 6.0 sec 5.22 GBytes 22.4 Gbits/sec

[3] 6.0- 8.0 sec 4.79 GBytes 20.6 Gbits/sec

[3] 8.0-10.0 sec 5.29 GBytes 22.7 Gbits/sec

[3] 10.0-12.0 sec 5.80 GBytes 24.9 Gbits/sec

[3] 12.0-14.0 sec 5.65 GBytes 24.3 Gbits/sec

Linux bridge with OVS bridge connected by OVS

internal port

This fabric connects each virtual machine to one OVS bridge irrespectively, and then add each OVS bridge's internal port into a Linux bridge.

The interface definition of virtual machine is like:

```
<interface type='bridge'>
  <mac address='52:54:00:a0:4c:09'/>
  <source bridge='br-int-1'/>
  <virtualport type='openvswitch'>
    <parameters interfaceid='6a7bbe68-f4fe-f8fb-cf09-f2bb933e005f'/>
  </virtualport>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

And the interfaces of OVS and Linux bridge are like this:

```
# ovs-vsctl show
e5e2f9b3-a938-48cf-8754-6c70f7e251bc
    Bridge "br-int-2"
        Port "vnet1"
            Interface "vnet1"
        Port "br-int-2"
            Interface "br-int-2"
                type: internal
    Bridge "br-int-1"
        Port "br-int-1"
            Interface "br-int-1"
                type: internal
        Port "vnet0"
            Interface "vnet0"
ovs_version: "1.11.0"
```

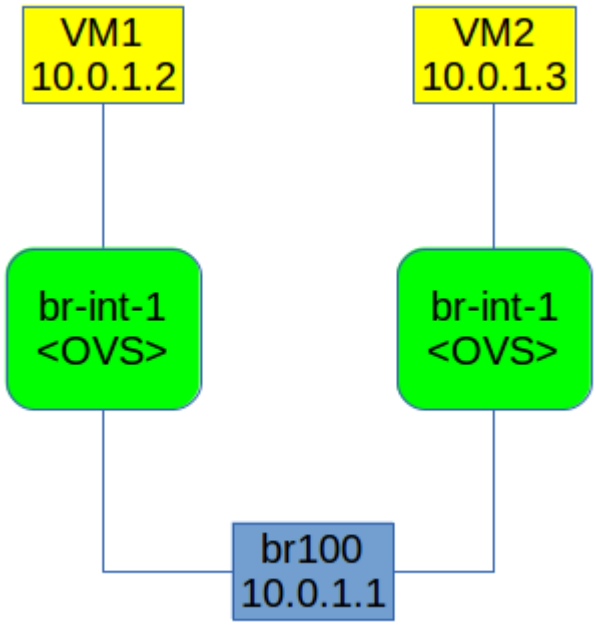
brctl show

bridge name bridge id STP enabled interfaces

br100 8000.a236a5523a46 no br-int-1

br-int-2

This network structure is shown as following:



Host to virtual machine

Run on VM2:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.1 port 42231

[4] 0.0-30.0 sec 90.2 GBytes 25.8 Gbits/sec

Run on host:

$ iperf -c 10.0.1.3 -i 2 -t 30

Client connecting to 10.0.1.3, TCP port 5001

TCP window size: 22.9 KByte (default)

---

[3] local 10.0.1.1 port 42231 connected with 10.0.1.3 port 5001

[ID] Interval Transfer Bandwidth

[3] 0.0- 2.0 sec 5.69 GBytes 24.4 Gbits/sec

[3] 2.0- 4.0 sec 6.24 GBytes 26.8 Gbits/sec

[3] 4.0- 6.0 sec 6.16 GBytes 26.5 Gbits/sec

[3] 6.0- 8.0 sec 5.99 GBytes 25.7 Gbits/sec

VM1 to VM2

Start iperf server on VM2:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.1.3 port 5001 connected with 10.0.1.2 port 42533

[ID] Interval Transfer Bandwidth

[4] 0.0-30.0 sec 70.6 GBytes 20.2 Gbits/sec

Start client on VM1:

iperf -c 10.0.1.3 -i 2 -t 30

Client connecting to 10.0.1.3, TCP port 5001

TCP window size: 23.5 KByte (default)

---

[3] local 10.0.1.2 port 42533 connected with 10.0.1.3 port 5001

[ID]	Interval	Transfer	Bandwidth
[3]	0.0- 2.0 sec	4.78 GBytes	20.5 Gbits/sec
[3]	2.0- 4.0 sec	4.56 GBytes	19.6 Gbits/sec
[3]	4.0- 6.0 sec	4.53 GBytes	19.4 Gbits/sec
[3]	6.0- 8.0 sec	5.04 GBytes	21.7 Gbits/sec
[3]	8.0-10.0 sec	4.74 GBytes	20.3 Gbits/sec
[3]	10.0-12.0 sec	4.65 GBytes	20.0 Gbits/sec
[3]	12.0-14.0 sec	4.81 GBytes	20.7 Gbits/sec
[3]	14.0-16.0 sec	4.90 GBytes	21.0 Gbits/sec
[3]	16.0-18.0 sec	4.75 GBytes	20.4 Gbits/sec

Devstack test

In this case, ML2 plugin and OVS agent is used. The fabric is similar to one in the case “OVS bridge with Linux bridge connected via veth”.

We customize the localrc:

```
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
Q_PLUGIN=ml2
Q_AGENT=openvswitch
DATABASE_PASSWORD=root
RABBIT_PASSWORD=password
SERVICE_TOKEN=token
SERVICE_PASSWORD=password
ADMIN_PASSWORD=password
SCREEN_LOGDIR=$DEST/logs/screen
```

After running the stack.sh, run following commands to open ports of VMs:

```
$ neutron security-group-rule-create --protocol icmp --direction ingress default
```

$ neutron security-group-rule-create --protocol tcp --port-range-min 0 --port-range-max 65000 --direction ingress default

Host to virtual machine

Run on VM2:

```
# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.4 port 5001 connected with 10.0.0.1 port 49297
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  3.53 GBytes  1.01 Gbits/sec
```

Run iperf client on host's router namespace:

```
$ sudo ip netns exec qrouter-63b8359f-a774-4e2d-83ff-5f0172644b2b iperf -c 10.0.0.4 -i 2 -t 30
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 49297 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 2.0 sec   224 MBytes   941 Mbits/sec
[  5]  2.0- 4.0 sec   218 MBytes   916 Mbits/sec
[  5]  4.0- 6.0 sec   228 MBytes   957 Mbits/sec
[  5]  6.0- 8.0 sec   235 MBytes   985 Mbits/sec
[  5]  8.0-10.0 sec   260 MBytes  1.09 Gbits/sec
[  5] 10.0-12.0 sec   280 MBytes  1.18 Gbits/sec
[  5] 12.0-14.0 sec   226 MBytes   946 Mbits/sec
[  5] 14.0-16.0 sec   243 MBytes  1.02 Gbits/sec
[  5] 16.0-18.0 sec   238 MBytes   999 Mbits/sec
[  5] 18.0-20.0 sec   247 MBytes  1.03 Gbits/sec
```

[5] 20.0-22.0 sec 250 MBytes 1.05 Gbits/sec

VM1 to VM2

Run iperf server on VM2:

```
# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.3 port 5001 connected with 10.0.0.4 port 37903
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  7.85 GBytes  2.25 Gbits/sec
```

Run iperf client on VM1:

```
# iperf -c 10.0.0.3 -i 2 -t 30
------------------------------------------------------------
Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 23.5 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.4 port 37903 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  548 MBytes  2.30 Gbits/sec
[  3]  2.0- 4.0 sec  542 MBytes  2.27 Gbits/sec
[  3]  4.0- 6.0 sec  536 MBytes  2.25 Gbits/sec
[  3]  6.0- 8.0 sec  516 MBytes  2.16 Gbits/sec
[  3]  8.0-10.0 sec  552 MBytes  2.32 Gbits/sec
[  3] 10.0-12.0 sec  510 MBytes  2.14 Gbits/sec
[  3] 12.0-14.0 sec  530 MBytes  2.22 Gbits/se
```

Devstack test with OVS internal port

It is obvious the veth connection solution is bad. So I tried to write a new vif driver named LibvirtHybridOVSBridgeDriverViaInternalPort which uses the OVS internal port as connection between Linux bridge and OVS bridge.

Add the follwing line in the localrc file used by previous case:

NOVA_VIF_DRIVER=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriverViaInternalPort

This case's network structure is the same to the one in the case “OVS bridge with Linux bridge connected via OVS internal port”.

Host to virtual machine

On virtual machine, run iperf server:

```
# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.5 port 5001 connected with 10.0.0.1 port 57274
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  95.4 GBytes  27.3 Gbits/sec
```

In the host's route namespace run:

```
$ sudo ip netns exec qrouter-1021848e-d727-4085-8883-de3ddca677eb iperf -c 10.0.0.5 -i 2 -t 30
------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 57274 connected with 10.0.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 2.0 sec  6.83 GBytes  29.3 Gbits/sec
[  5]  2.0- 4.0 sec  6.14 GBytes  26.4 Gbits/sec
[  5]  4.0- 6.0 sec  6.11 GBytes  26.3 Gbits/sec
[  5]  6.0- 8.0 sec  6.76 GBytes  29.0 Gbits/sec
[  5]  8.0-10.0 sec  6.29 GBytes  27.0 Gbits/sec
[  5] 10.0-12.0 sec  6.88 GBytes  29.5 Gbits/sec
```

VM1 to VM2

On VM2 run:

iperf -s

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.0.5 port 5001 connected with 10.0.0.3 port 49414

[ID] Interval Transfer Bandwidth

[4] 0.0-30.0 sec 69.2 GBytes 19.8 Gbits/sec

On VM1 run:

iperf -c 10.0.0.5 -i 2 -t 30

Client connecting to 10.0.0.5, TCP port 5001

TCP window size: 23.5 KByte (default)

---

[3] local 10.0.0.3 port 49414 connected with 10.0.0.5 port 5001

[ID] Interval Transfer Bandwidth

[3] 0.0- 2.0 sec 4.15 GBytes 17.8 Gbits/sec

[3] 2.0- 4.0 sec 4.63 GBytes 19.9 Gbits/sec

[3] 4.0- 6.0 sec 4.48 GBytes 19.3 Gbits/sec

[3] 6.0- 8.0 sec 5.05 GBytes 21.7 Gbits/sec

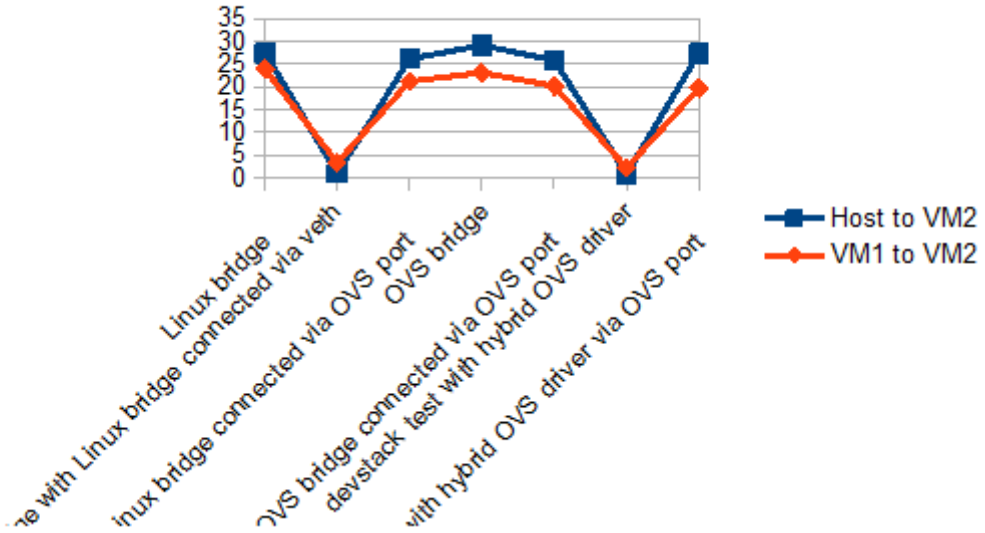
[3] 8.0-10.0 sec 4.98 GBytes 21.4 Gbits/sec

[3] 10.0-12.0 sec 4.94 GBytes 21.2 Gbits/sec

Conclusion

Collect the bandwidth data of all the “iperf -s”, we can have a table and a figure below.

		Bandwidth output of iperf -s	
		Host to VM2	VM1 to VM2
1	Linux bridge	27.4	23.9
2	OVS bridge with Linux bridge connected via veth	1.13	3.43
3	OVS bridge with Linux bridge connected via OVS port	26.1	21.2
4	OVS bridge	29.1	23.1
5	Linux bridge with OVS bridge connected via OVS port	25.8	20.2
6	devstack test with hybrid OVS driver	1.01	2.25
7	devstack test with hybrid OVS driver via OVS port	27.3	19.8



We can get these points:

1. The network bandwidth is not influenced much by the so-called *nine devices passage. We can prove it from the fact that the bandwidth of cases 3, 5 and 7 are near to the bandwidth of cases 1 and 4;

2. The veth pair is reducing the bandwidth a lot. The cases 2 and 6 are using veth, where the bandwidth is almost ten times worse than other solution;

3. The OVS bridge is as good as Linux bridge which can be seen by comparing case 1 & case 4.