1.0 Introduction

Since 2015, Intel Corporation has cooperated with several communication service providers to define optimal Network Functions Virtualization Infrastructures (NFVIs) that would help virtualize their workloads.

As part of this initiative, Intel helped a prominent video, phone, and cable Internet access provider to build a performant generic infrastructure that could be used for different workloads such as video on demand and high-speed, data-related network functions. This infrastructure had to be capable of supporting edge and access services across technologies such as Data Over Cable Service Interface Specifications (DOCSIS), Ethernet, or passive optical networks.

Network services utilizing such an infrastructure are often based on the TCP/IP communication. An example of such a service is a TCP speed test. Many Internet customers use a speed test server as a tool to compare the actual speed they are experiencing with the speed they signed up for. These speed test servers are also based on the TCP; therefore the TCP performance is critical in such a network infrastructure.

This proof-of-concept (PoC) solution is built from three commercial-off-the-shelf (COTS) Intel® Xeon® processor-based servers and the open source software platform, primarily based on OpenStack that provides the cloud computing environment. New network functions virtualization (NFV) extensions integrated into OpenStack Kilo include the support for non-uniform memory access (NUMA) topology awareness, CPU affinity in virtual machines (VM), and huge pages, to improve overall OpenStack VM performance.

For fast packet processing, we integrated the Open vSwitch* with the Data Plane Development Kit (DPDK) on the host machines. Furthermore, we used features such as multi-queue, and implemented a patch to enable the TCP segmentation offload in DPDK-accelerated Open vSwitch (OVS-DPDK) that helped achieve an additional performance boost.

The primary audience for this document are architects and engineers planning to implement and optimize their TCP-based NFVIs. While this document discusses the solution in detail, it is not a large-scale solution. This guide also provides the installation steps of the software components and the performance optimizations implemented to deliver an optimal virtualized infrastructure capable of handling communications-grade Network Functions Virtualization workloads.

In the Technical Brief companion document (https://builders.intel.com/docs/networkbuilders/implementing_a_TCP_broadband_speed_test_in_the_cloud_for_use_in_an_NFV_infrastructure.pdf) we present three examples of test scenarios that were built on this infrastructure to reflect real-world setups and share the test results. Our findings show that open source software can be successfully used to achieve a reliable and flexible topology that enables delivery of robust and highly efficient virtualized infrastructures.
# Table of Contents

1.0 Introduction .................................................................................................................. 1

2.0 Solution Overview ......................................................................................................... 3

3.0 Installation Guide .......................................................................................................... 4

3.1 Enable Hardware Features ............................................................................................ 4

3.2 Prepare Host Machines for the OpenStack* Installation ............................................. 4

3.3 Install the OpenStack Using Packstack* ..................................................................... 6

3.4 Enable the networking-ovs-dpdk Plug-In .................................................................... 7

3.4.1 Prepare the OpenStack Nodes ................................................................................. 7

3.4.2 Clone the Required Repositories ........................................................................... 9

3.4.3 Install the OVS-DPDK ......................................................................................... 9

3.5 Post-Installation Configuration .................................................................................. 12

3.6 Enable Traffic Monitoring on the vHost-user and DPDK Interfaces. ......................... 13

3.6.1 Configure the sFlow* in OVS-DPDK. ................................................................... 13

3.6.2 Install the InMon sFlowTrend* Data Collector ...................................................... 13

4.0 Performance Optimizations ......................................................................................... 14

4.1 Optimize the Host ....................................................................................................... 14

4.1.1 Isolate the CPU Cores ....................................................................................... 14

4.1.2 Enable 1 GB Huge Pages ................................................................................... 15

4.1.3 Enable the TCP Segmentation Offload in OVS-DPDK ............................................ 15

4.1.4 Enable the Multiqueue Feature for vHost-user and Physical DPDK Interfaces ........ 16

4.1.5 Enable Core Pinning and NUMA Awareness in the OpenStack Compute .............. 17

4.2 Optimize the Guest .................................................................................................... 17

4.2.1 The 'extra_specs' Properties for OpenStack VMs ................................................ 17

4.2.2 Enable Multiqueue for VirtIO Interfaces ............................................................... 17

4.2.3 Upgrade the CentOS* 7 Kernel to version 4.5.5 on the Guest ................................ 17

4.2.4 Additional TCP Performance Tunings on the CentOS* 7 Guest. ......................... 17

Appendix A: The Packstack Answer File .......................................................................... 18

Appendix B: Create OpenStack Networks and Router ...................................................... 21

Appendix C: Basic Configuration of Brocade 5600 vRouter .............................................. 22

Appendix D: Abbreviations ................................................................................................. 22
2.0 Solution Overview

The NFVI used for this proof-of-concept (PoC) solution consists of three COTS Intel® Xeon® processor-based servers, running the Fedora® 21 Server operating system (OS). OpenStack® Kilo was installed on these servers to provide a cloud computing platform.

One server is configured as the OpenStack controller that also includes the OpenStack Networking* functions, whereas the remaining two servers are configured as OpenStack compute nodes.

Figure 1 shows the physical topology and the software stack of the PoC.

Each server has four network interfaces that, through a top-of-rack switch, provide connectivity to the networks described in Table 1.

Table 1. Networks used in the PoC.

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>NETWORK DESCRIPTION</th>
<th>COMPUTE NODES NIC</th>
<th>CONTROLLER NODE NIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>Flat provider network used for Internet/remote access to the hosts and OpenStack* virtual machines (VMs).</td>
<td>Intel® Ethernet Server Adapter I350-T4</td>
<td></td>
</tr>
<tr>
<td>VLAN</td>
<td>802.1Q tagged network mapped to the existing physical virtual local area network (VLAN) provider network. This network simulates the subscribers.</td>
<td>Intel® Ethernet Server Adapter X520-DA2</td>
<td>Intel® Ethernet Converged Network Adapter X710-DA4</td>
</tr>
<tr>
<td>Management</td>
<td>Management network used for accessing and managing OpenStack services.</td>
<td>Intel Ethernet Server Adapter I350-T4</td>
<td></td>
</tr>
<tr>
<td>Data (VxLAN)</td>
<td>Virtual extensible local area network (VxLAN) tunnel used for east-west traffic between tenant VMs on different hosts.</td>
<td>Intel Ethernet Server Adapter X520-DA2</td>
<td>Intel Ethernet Converged Network Adapter X710-DA4</td>
</tr>
</tbody>
</table>
Table 2 presents the specification of the hardware used in the solution.

### Table 2. Specification of the hardware components.

<table>
<thead>
<tr>
<th>HARDWARE</th>
<th>SPECIFICATION</th>
</tr>
</thead>
</table>
| Controller / Neutron host server | • 2x Intel® Xeon® processor E5-2680 v3, 2.50 GHz, total of 48 logical cores with Intel® Hyper-Threading Technology  
                                 | • 128 GB, DDR4-2133 RAM  
                                 | • Intel® Ethernet Server Adapter I350-T4 (using Intel® Ethernet Controller I350)  
                                 | • Intel® Ethernet Converged Network Adapter X710-DA4 (using Intel® Ethernet Controller XL710-AM1)  
                                 | • 200 GB HDD  |
| Compute 1 host server     | • 2x Intel® Xeon® processor E5-2680 v2, 2.80 GHz, total of 40 logical cores with Intel Hyper-Threading Technology  
                                 | • 64 GB, DDR3-1600 RAM  
                                 | • Intel Ethernet Server Adapter I350-T4 (using Intel Ethernet Controller I350)  
                                 | • Intel® Ethernet Server Adapter X520-DA2 (using Intel® 82599ES 10 Gigabit Ethernet Controller)  
                                 | • 1 TB HDD  |
| Compute 2 host server     |                                                                                                                                               |
| Top-of-rack switch        | • Extreme Networks Summit* X670V-48t-BF-AC 10GbE Switch, SFP+ Connections                                                                       |

### 3.0 Installation Guide

This chapter contains the instructions for installation and configuration of the software stack.

#### 3.1 Enable Hardware Features

Before starting to install the OS, enable the following features in the BIOS of all host machines:

- Intel® Virtualization Technology
- Intel® Hyper-Threading Technology (Intel® HT Technology)
- Intel® Turbo Boost Technology

#### 3.2 Prepare Host Machines for the OpenStack* Installation

**Note:** The instructions for installing Fedora 21 Server are not within the scope of this document; however, this section contains some information that user needs to follow during OS installation or configuration.

1. Install the following packages while installing the OS.
   - C development tools and libraries
   - Development tools
   - Virtualization
2. Create custom partitioning as presented in Table 3.

### Table 3. Solution partitioning schema.

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biosboot</td>
<td>2 MB</td>
</tr>
<tr>
<td>/boot</td>
<td>2 GB</td>
</tr>
<tr>
<td>/swap</td>
<td>Double the size of physical memory</td>
</tr>
<tr>
<td>/ (root partition)</td>
<td>Remaining space</td>
</tr>
</tbody>
</table>
3. After the OS is installed, configure the network interfaces on the host machines with the proper IP addresses. On each host machine, eno1, eno2, and eno3 interfaces are used for the External, Management, and VxLAN tunnel networks respectively. These interfaces are assigned with static IP addresses as mentioned in Table 4. On the VLAN interface, no assignment of IP address is required on any node.

**Table 4. The IP addresses of the setup.**

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>EXTERNAL IP ADDRESS</th>
<th>MANAGEMENT IP ADDRESS</th>
<th>VXLAN TUNNEL IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>10.250.100.101</td>
<td>172.16.101.2</td>
<td>172.16.111.2</td>
</tr>
<tr>
<td>Compute 1</td>
<td>10.250.100.102</td>
<td>172.16.101.3</td>
<td>172.16.111.3</td>
</tr>
<tr>
<td>Compute 2</td>
<td>10.250.100.126</td>
<td>172.16.101.4</td>
<td>172.16.111.4</td>
</tr>
<tr>
<td>OpenStack dashboard</td>
<td><a href="http://10.250.100.101/dashboard/auth/login/?next=/dashboard/">http://10.250.100.101/dashboard/auth/login/?next=/dashboard/</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Network</td>
<td>10.250.100.0/24, Untagged</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN2 Network</td>
<td>20.20.20.0/24, VLAN ID=2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public DNS1</td>
<td>8.8.4.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public DNS2</td>
<td>4.2.2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public DNS3</td>
<td>4.2.2.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the Fedora 21 OS, the network script files are located in the `/etc/sysconfig/network-scripts` directory. Since the NetworkManager service is not used, the following line is added into the network script file of each interface.

```
NM_CONTROLLED=no
```

A sample network script file with a static IP address assigned on the management interface on the controller node is shown below.

```
TYPE=Ethernet
BOOTPROTO=static
IPADDR=172.16.101.2
NETMASK=255.255.255.0
DEFROUTE=no
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
NAME=eno2
DEVICE=eno2
UUID=58215fc4-845e-4e0d-af51-588beb53f536
ONBOOT=yes
HWADDR=EC:F4:BB:C8:58:7A
PEERDNS=yes
PEERROUTES=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
NM_CONTROLLED=no
```

4. Once all IP addresses are configured, disable the NetworkManager and enable the network service on all the host machines in the following order.

```
# systemctl disable NetworkManager
# systemctl enable network
# systemctl stop NetworkManager
# systemctl restart network
```
5. Set the host name on all the host machines by editing the `/etc/hostname` files. Additionally, provide all the host names of the setup with their management IP addresses into the `/etc/hosts` files on each host machine. An example is shown below.

```
172.16.101.2 controller controller.localdomain
172.16.101.3 compute1 compute1.localdomain
172.16.101.3 compute2 compute2.localdomain
```

6. Update the software packages on each of the host machines.

```
# yum --y update
```

7. Disable Security-Enhanced Linux* (SELinux) and the firewall on all the host machines. Edit the `etc/sysconfig/selinux` file and set SELINUX=disabled to permanently disable SELinux. The following commands can be used to disable the firewall service and, temporarily, SELinux.

```
# setenforce 0
# sestatus
# systemctl disable firewalld.service
# systemctl stop firewalld.service
```

8. Uncomment the `PermitRootLogin` line in the `/etc/ssh/sshd_config` file.

```
PermitRootLogin=yes
```

**Note:** Remote login as root is not advisable from the security standpoint.

9. Reboot all the host machines.

### 3.3 Install the OpenStack Using Packstack*

To install the OpenStack Kilo using Packstack, perform the following steps.

1. Set up RDO repositories on all of the nodes.

```
# yum install --y https://repos.fedorapeople.org/openstack/openstack-kilo/rdo-release-kilo-1.noarch.rpm
```

2. Install Packstack on the controller node.

```
# yum install openstack-packstack
```

3. Update packages on all of the nodes.

```
# yum --y update
```

4. Run Packstack to generate the answer file. Packstack uses the answer file to install and configure the desired OpenStack services.

```
# packstack --gen-answer-file=answerfile.txt
```

The answer file used in this setup is presented in Appendix A: The Packstack Answer File. Edit the answer file based on your setup requirements and network configurations.

5. Run Packstack with the edited answer file on the controller node.

```
# packstack --answer-file=./answerfile.txt
```

6. After OpenStack is installed, verify that all of the installed OpenStack services are running.

```
# cd /root
# source keystonerc_admin
# openstack-status
```

Check the `openstack-status` on the controller node to make sure that the services are active and all the service parameters (for example, both compute nodes, keystone user-list, glance image-list, and so on) are listed.

7. Update the `/etc/neutron/13_agent.ini` file on the controller node and remove the `br-ex` from the line as shown below.

```
external_network_bridge =
```

8. Update `/etc/neutron/plugins/ml2/ml2_conf.ini` file on the controller node as shown below.

```
[m12]
type_drivers = vxlan,flat,gre,vlan,local
tenant_network_types = vxlan,flat,gre,vlan,local
mechanism_drivers = openvswitch,linuxbridge,l2population
extension_drivers = port_security

[m12_type_flat]
flat_networks = external
network_vlan_ranges = physnet1:2:1000

[ovs]
bridge_mappings = external:br-ex,physnet1:br-vlan
```

9. Restart the `neutron-server` and `neutron-l3-agent` services on the controller node.

10. Update the `/etc/httpd/conf.d/15-horizon_vhost.conf` file on the controller node so that the external IP address can be used to access the OpenStack dashboard.

```
ServerAlias 172.16.101.2
ServerAlias *
```

11. Reload the `httpd` service on the controller node.

```
# service httpd reload
```

12. Log in to verify that the OpenStack dashboard is accessible. The credentials set in the answer file are admin/password. In the PoC example, the URL to access the dashboard is `http://10.250.100.101/dashboard/auth/login/?next=/dashboard/`.

---

**Figure 2. The OpenStack* login screen.**
13. In the OpenStack dashboard, click **Project -> Access & Security**, and then modify the default security group to add new rules for ingress and egress traffic types such as transmission control protocol (TCP), Internet Control Message Protocol (ICMP), and User Datagram Protocol (UDP). The final set of rules should be similar to the sample security rules shown below.

![Figure 3. Managing security group rules in the OpenStack* dashboard.](image)

Execute the following steps to verify the networking and connectivity within the OpenStack cloud setup.

14. Create the public and private networks in OpenStack. Refer to the **Appendix B: Create OpenStack Networks and Router** for commands used to create different types of OpenStack networks.

15. Download the CirrOS* image from [http://docs.openstack.org/image-guide/content/ch_obtaining_images.html](http://docs.openstack.org/image-guide/content/ch_obtaining_images.html), and import it to the OpenStack Image Service*.

In the following example, the CirrOS image is downloaded into the `/etc/demouser/Downloads` directory.

```bash
# cd /home/demouser/Downloads/
# glance image-create --name="cirros" --is-public=true --container-format=bare --disk-format=qcow2 --progress < cirros-0.3.4-x86_64-disk.img
```

16. Launch two VMs on the private network, and then check whether the VMs are getting IP addresses through the Dynamic Host Configuration Protocol (DHCP). Check whether the VMs can ping each other. The credentials for the CirrOS image are `cirros/cubswin`.

17. Create another private network, and spawn a new CirrOS VM on this new subnet.

18. Create the OpenStack router as presented in **Appendix B: Create OpenStack Networks and Router** or by using OpenStack dashboard, and attach it to both the private networks and the public network. Create a floating IP address within the public network and associate it to one of the VMs previously spawned on the private network.

Alternatively, you may want to install the Brocade 5600 vRouter*. The respective configuration steps are presented in **Appendix C: Basic Configuration of Brocade 5600 vRouter**.

**Note:** Floating IP address can be created through the OpenStack dashboard following **Compute -> Access & Security -> Floating IPs -> Allocate IP to project**; select **Public network** from a drop-down menu.

19. Ping the associated floating IP address from the host machine and any other machine connected to the same public network.

20. Check whether the VMs on different private networks can ping each other and any IP address on the Internet.

21. To create an OpenStack virtual local area network (VLAN) with a specific VLAN ID and a classless inter-domain routing (CIDR), use the commands provided in the **Appendix B: Create OpenStack Networks and Router**.

### 3.4 Enable the networking-ovs-dpdk Plug-in

#### 3.4.1 Prepare the OpenStack Nodes

Perform the following steps on the controller node.

1. Install missing packages.
   ```bash
   # yum install openstack-neutron openstack-neutron-ml2 python-neutronclient openstack-neutron-openvswitch
   ``

   **Note:** The packages mentioned above may already be installed by Packstack.

2. Edit the following parameters in the `/etc/sysctl.conf` file as presented below.
   ```bash
   net.ipv4.ip_forward=1
   net.ipv4.conf.all.rp_filter=0
   net.ipv4.conf.default.rp_filter=0
   ```

3. Commit the changes.
   ```bash
   # sysctl -p
   ```

4. Recreate the MySQL* database for the OpenStack Networking services. Enter the MySQL shell.
   ```bash
   # mysql -u root -p
   ```

   Execute the following set of MySQL commands.
   ```sql
   DROP DATABASE neutron;
   CREATE DATABASE neutron;
   GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'intel';
   GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'intel';
   ```

5. Edit the following parameters in the `/etc/neutron/plugins/ml2/ml2_conf.ini` file as presented below.
   ```ini
   [ml2]
   type_drivers = flat,vlan,gre,vxlan
   tenant_network_types = vxlan, vlan
   mechanism_drivers = openvswitch
   [ml2_type_flat]
   ```
flat_networks = external

[ml2_type_vlan]
... 
vlan_ranges = physnet1:2:1000

[ml2_type_vxlan]
... 
vni_ranges = 1001:2000

[securitygroup]
... 
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall. OVSHybridIptablesFirewallDriver
enable_ipset = True

[ovs]
... 
local_ip = 172.16.111.2
bridge_mappings = external:br-ex, physnet1:br-vlan

[agent]
... 
tunnel_types = vxlan

6. Create a symbolic link.

```
# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

7. Edit the `/usr/lib/systemd/system/neutron-openvswitch-agent.service` file and replace the following entry

```
```

with the line below.

```
```

8. Populate the database with the new configuration.

```
# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

9. Verify that all the Open vSwitch bridges exist before restarting the OpenStack Networking services.

```
# systemctl daemon-reload
# systemctl restart neutron*
```

Perform the following steps on each compute node.

10. Edit the following parameters in the `/etc/sysctl.conf` file as presented below.

```
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
```

11. Apply the changes.

```
# sysctl -p
```

12. Install networking components.

```
# yum install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch
```

**Note:** The packages mentioned above may already be installed by Packstack.

13. Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file as presented below.

```
[ml2]
... 
type_drivers = vxlan,vlan,flat
tenant_network_types = vxlan,vlan,flat
mechanism_drivers = openvswitch
[ml2_type_flat]
... 
flat_networks = external
[ml2_type_vxlan]
... 
network_vlan_ranges = physnet1:2:1000
[securitygroup]
... 
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.firewall.NoopFirewallDriver
[ovs]
... 
bridge_mappings = external:br-ex,physnet1:br-vlan
local_ip = 172.16.111.3
tunnel_types = vxlan
tunnel_id_ranges = 32769:34000
enable_tunneling = True
[agent]
... 
tunnel_types = vxlan
```
14. Create a symbolic link.

```
# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

15. Edit the /usr/lib/systemd/system/neutron-openvswitch-agent.service file and replace the following entry

```
```

with the line below.

```
```

16. Restart all the OpenStack Networking services.

```
# systemctl daemon-reload
# systemctl restart neutron*
```

Perform the following step on the controller node.

17. Recreate all the OpenStack networks as given in Appendix B: Create OpenStack Networks and Router.

### 3.4.3 Install the OVS-DPDK

1. Change the directory to the DPDK directory, and then edit the following lines in the config/common_linuxapp file.

```
CONFIG_RTE_BUILD_COMBINE_LIBS=y
CONFIG_RTE_LIBRTE_VHOST=y
```

2. Build the DPDK.

```
# export RTE_TARGET=x86_64-native-linuxapp-gcc
# make install T=$RTE_TARGET DESTDIR=install
```

3. Change the directory to the Open vSwitch directory, and then build the Open vSwitch with DPDK.

```
# ./boot.sh
# ./configure --with-dpdk=<DPDK_DIR>/<TARGET> --prefix=/usr --with-rundir=/var/run/openvswitch
# make CFLAGS='-O3 -march-native'
# make install
```

4. Change the directory to the networking-ovs-dpdk directory, and install the ovs-dpdk agent.

```
# yum install python-pip
# python setup.py install
```

5. Stop the native openvswitch service.

```
# systemctl stop openvswitch
```


```
# systemctl stop neutron-openvswitch-agent.service
```

7. Change the directory to the networking-ovs-dpdk directory, and then copy the files as shown below.

```
# cd ~/networking-ovs-dpdk
# cp devstack/ovs-dpdk/ovs-dpdk-init /etc/init.d/ovs-dpdk
# cp devstack/ovs-dpdk/ovs-dpdk-conf /etc/default/ovs-dpdk
```

8. Edit the /etc/default/ovs-dpdk file to match your environment. Use the content below as an example, and adjust paths, huge pages, and other settings.

```
RTE_SDK=/root/source/dpdk
RTE_TARGET=x86_64-native-linuxapp-gcc
```

### 3.4.2 Clone the Required Repositories

1. Clone the networking-ovs-dpdk git repository on both the compute and controller nodes.

```
# git clone https://github.com/openstack/networking-ovs-dpdk.git
# cd networking-ovs-dpdk
# git checkout origin/stable/kilo
```

2. Clone the dpdk repository only on the compute nodes.

```
# git clone http://dpdk.org/git/dpdk
# cd dpdk
# git checkout v2.2.0
```

**Note:** You can check out the v16.04 tag of dpdk repository to enable some of the performance optimizations. Refer to the section 4.0 Performance Optimizations for more information.

3. Clone the ovs repository only on compute nodes.

```
# git clone https://github.com/openvswitch/ovs.git
# cd ovs
# git checkout v2.5.0
```

**Note:** You can check out the newer Open vSwitch version to enable some of the performance optimizations. Refer to the section 4.0 Performance Optimizations for more information and detailed instructions.
9. Create a backup of the qemu-kvm executable file.

   # mv /usr/bin/qemu-kvm /usr/bin/qemu-kvm.orig

10. Create a new qemu-kvm executable script that includes the support for DPDK vhost-user ports for newly created VMs on this node. To do so, create a new qemu-kvm file

   # touch /usr/bin/qemu-kvm

Open the newly created /usr/bin/qemu-kvm file, paste the following code, and then save it.

```bash
#!/bin/bash -

VIRTIO_OPTIONS="csum=off,gso=off,guest
    ts04=off,guest_ts06=off,guestecn=off,guest_ufo=off"

VHOST_FORCE="vhostforce=on"

SHARE="share=on"

add_mem=False

i=0

while [ $# -gt 0 ]; do
    case "$1" in
        -netdev)
            args[i]="$1"
            (( i++ ))
            shift
            if [[ "$1" == "vhost-user" ]];
                then
                    args[i]="${1},${VHOST_FORCE}

                (( i++ ))
                shift
                fi
        ;;
        -device)
            args[i]="$1"
            (( i++ ))
            shift
            if [[ "$1" == "virtio-net-pci*" ]];
                then
                    args[i]="${1},${VIRTIO_OPTIONS}

                (( i++ ))
                shift
                fi
        ;;
        -object)
            args[i]="$1"
            (( i++ ))
    esac

done

if [ -e /usr/local/bin/qemu-system-x86_64 ]; then
    exec /usr/local/bin/qemu-system-x86_64 "${args[@]}"
elif [ -e /usr/libexec/qemu-kvm.orig ]; then
    exec /usr/libexec/qemu-kvm.orig "${args[@]}"
fi
```

11. Add execution permissions to the qemu-kvm file and the networking-ovs-dpdk plug-in executable files.

   # chmod +x /usr/bin/qemu-kvm
   # chmod +x /usr/bin/networking-ovs-dpdk-agent

12. Edit the OpenStack Networking neutron ml2 agent settings.

On the compute node, open the /etc/neutron/plugins/ml2/ml2_conf.ini file, and then edit the mechanism_drivers parameter as shown below.

```ini
[DEFAULT] ...
    mechanism_drivers = ovspqdk

[securitygroup] ..
    firewall_driver = neutron.agent.firewall.NoopFirewallDriver
```

On the controller node, open the /etc/neutron/plugins/ml2/ml2_conf.ini file, and then add the ovspqdk entry to the mechanism_drivers parameter as shown below.

```ini
[DEFAULT] ...
    mechanism_drivers = openvswitch, ovspqdk

[securitygroup] ..
    firewall_driver = neutron.agent.firewall.NoopFirewallDriver
```
In the same file on both the compute and controller nodes, configure the VxLAN tunnel settings.

```ini
[ovs]
  ...
  local_ip = IP_OF_THE_INTERFACE_USED_FOR_TUNNEL
[agent]
  ...
  tunnel_types = vxlan
```

13. Edit the `/etc/libvirt/qemu.conf` file, and then change the user and group parameters to qemu.

```bash
user = "qemu"
group = "qemu"
```

Set the `hugetlbfs_mount` location to match your system settings.

```bash
hugetlbfs_mount = "/mnt/huge"
```

14. Due to errors in the ovs-dpdk script, edit the `/etc/init.d/ovs-dpdk` file.

At line 191, change:

```bash
sudo ip link $nic 0 down
```

to:

```bash
sudo ip link set dev $nic down
```

At line 376, change:

```bash
while [ ! $(grep "unix.*connected" ${OVS_LOG_DIR}/ovs-vswitchd.log) ]; do
```

to:

```bash
while [ ! "$(grep 'unix.*connected' ${OVS_LOG_DIR}/ovs-vswitchd.log)" ]; do
```

Insert the following lines after line 410:

```bash
  echo "vhostuser sockets cleanup"
  rm -f $OVS_DB_SOCKET_DIR/vhu*
```

Save the file, and then exit.

15. Initialize the ovs-dpdk service.

At this point, it is recommended that you remove and recreate manually the Open vSwitch database file `conf.db` to avoid any issues with configuration of the Open vSwitch in the next steps.

Kill any Open vSwitch-related process running in your system, such as `ovs-vswitchd` and `ovsdb-server`.

```bash
# rm /usr/local/etc/openvswitch/conf.db
# ovsdb-tool create /etc/openvswitch/conf.db
```

Run the service initialization.

```bash
# service ovs-dpdk init
```

16. Run the ovs-dpdk service.

```bash
# service ovs-dpdk start
```

**Note:** To identify possible issues, pay attention to the output of this command, and check also the ovs-vswitchd logs located in the `/etc/default/ovs-dpdk` directory.

Check the status of the ovs-dpdk with the following command.

```bash
# systemctl status ovs-dpdk
```

**Note:** Automatic binding of igb_uio to the interfaces by the ovs-dpdk service was not fully tested and might not be working. If this happens, a solution is to disable this feature by commenting out the following parts of the `/etc/init.d/ovs-dpdk` script.

```bash
319  # bind_nics
...]
403  #if uio diver is not loaded
404  # echo "loading OVS_INTERFACE_DRIVER diver"
405  # if [[ "$OVS_INTERFACE_DRIVER" == "igb_uio" ]]; then
406  # load_igb_uio_module
407  # elif [[ "$OVS_INTERFACE_DRIVER" == "vfio-pci" ]]; then
408  # load_vfio_pci_module
409  # fi
...]
427  # echo "binding nics to linux_dirver"
428  # unbind_nics
429  #
430  # echo "unloading OVS_INTERFACE_DRIVER"
431  # if [[ "$OVS_INTERFACE_DRIVER" == "igb_uio" ]]; then
432  # remove_igb_uio_module
433  # elif [[ "$OVS_INTERFACE_DRIVER" =~ "vfio-pci" ]]; then
434  # remove_vfio_pci_module
435  # fi
```

17. Bind the DPDK interfaces to the igb_uio driver, and manually create the Open vSwitch bridges for these interfaces.

Execute the following commands to bind the interface to the igb_uio driver.

```bash
# modprobe uio
# modprobe cuse
# modprobe fuse
```

Change the directory to the DPDK directory, and then load the DPDK igb_uio driver.

```bash
# insmod x86_64-native-linuxapp-gcc/kmod/igb_uio.ko
```

**Note:** For a different DPDK target, replace the `x86_64-native-linuxapp-gcc` in the above command with the respective one.

18. Execute the following command to check the current binding status of all the interfaces.

```bash
# ./tools/dpdk_nic_bind.py --status
```
19. Bind the interfaces to the DPDK driver if needed. The interfaces must be in down status; otherwise, binding will fail. To bring the interfaces down execute the following command.

```
# ip l s dev <Interface-Name> down
```

20. The following command brings down the eno4 interface.

```
# ip l s dev eno4 down
```

To bind the interface to the DPDK driver, execute the command below.

```
# /root/dpdk/dpdk-nic-bind.py -b igb_iuo
<PCI_ADDRESS_OF_NIC_TO_BIND>
```

```
# /root/dpdk/tools/dpdk_nic_bind.py -b igb_uio 0000:04:00.0
```

To bind the interface back to the regular Linux driver, execute the command below.

```
# /root/dpdk/tools/dpdk-nic-bind.py -b
<DRIVER_NAME>
<PCI_ADDRESS_OF_NIC_TO_BIND>
```

21. Run the ovs-dpdk service.

```
# service ovs-dpdk start
```

### 3.5 Post-Installation Configuration

1. To create the Open vSwitch bridges with DPDK interfaces use the following commands. Table 5 shows the mapping of DPDK interfaces.

```
# ovs-vsctl add-br br-ex -- set bridge br-ex datapath_type=netdev
# ovs-vsctl add-port br-ex dpdk0 -- set Interface dpdk0 type=dpdk
# ovs-vsctl add-br br-vxlan -- set bridge br-vxlan datapath_type=netdev
# ovs-vsctl add-port br-vxlan dpdk1 -- set Interface dpdk1 type=dpdk
# ovs-vsctl add-br br-vlan -- set bridge br-vlan datapath_type=netdev
# ovs-vsctl add-port br-vlan dpdk2 -- set Interface dpdk2 type=dpdk
```

**Table 5. Mapping of DPDK interfaces.**

<table>
<thead>
<tr>
<th>DPDK INTERFACE NAME</th>
<th>PREVIOUS NAME</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpdk0</td>
<td>eno1</td>
<td>External network</td>
</tr>
<tr>
<td>dpdk1</td>
<td>eno3</td>
<td>VxLAN network</td>
</tr>
<tr>
<td>dpdk2</td>
<td>eno4</td>
<td>VLAN network</td>
</tr>
</tbody>
</table>

**Note:** The DPDK interfaces are sorted by the Peripheral Component Interconnect* (PCI*) address—the higher value of a PCI address results in a higher interface number.

Check the status of the Open vSwitch.

```
# ovs-vsctl show
```

If there are issues with adding the DPDK port to the bridge, restart the ovs-dpdk service after binding the DPDK interfaces using the command below.

```
# systemctl restart ovs-dpdk
```

2. Set the administrative status to up on all the Open vSwitch bridges except for the br-int.

**Note:** This step may be required after creating new Open vSwitch bridges and restarting the ovs-dpdk service.

The following sample command brings the br-vlan bridge up.

```
# ip link set dev br-vlan up
```

Use the following commands to assign an IP address to the VxLAN bridge.

```
# ip address add 172.16.111.3/24 dev br-vxlan
```

3. Once all the bridges are created and configured, start the networking-ovs-dpdk-agent.

```
# screen /usr/bin/networking-ovs-dpdk-agent \
--config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini
```

4. It is recommended that you run networking-ovs-dpdk-agent in the nohup, screen (as provided in the example above), or tmux session.

5. Restart the openstack-nova-compute service on the compute nodes.

```
# systemctl restart openstack-nova-compute
```

6. On the controller node, restart all the OpenStack Networking services.

```
# systemctl restart neutron*
```

7. On the controller node, check whether all of the OpenStack Networking and Compute services are running.

```
# neutron agent-list
# cd /root
# source keystonerc_admin
# openstack-status
```

There might also be an old Open vSwitch agent visible on the compute nodes. Make sure to delete manually all the entries with the agent_type as Open vSwitch agent. To delete the old agent, execute the following command.

```
# neutron agent-delete <id-of-the-non-dpdk-agent>
```

8. On the controller node, create a new flavor or update an already existing one, and set the extra_specs parameter as shown below. These updated flavors will be used for all OpenStack VMs.

```
# nova flavor-key <flavor-id> set "hw:mem_page_size"="large"
```
3.6 Enable Traffic Monitoring on the vHost-user and DPDK Interfaces

3.6.1 Configure the sFlow* in OVS-DPDK

Traffic monitoring using sFlow* was not supported on physical DPDK and vHost-user interfaces. To enable this feature, we implemented a patch and published it to the ovs-dev mailing list. To enable sFlow on DPDK interfaces, perform following steps.

1. Download the patch from the ovs-dev mailing list: [https://mail.openvswitch.org/pipermail/ovs-dev/2016-April/070317.html](https://mail.openvswitch.org/pipermail/ovs-dev/2016-April/070317.html), and then apply the patch on the Open vSwitch sources.

   ```bash
   # git am 0001-netdev-dpdk-add-sflow-support-for-vhost-user-ports.patch
   ``

   Alternatively, use the command below.

   ```bash
   # git apply 0001-netdev-dpdk-add-sflow-support-for-vhost-user-ports.patch
   ``

2. Recompile and reinstall the Open vSwitch.

   ```bash
   # ./boot.sh
   # ./configure --with-dpdk=<DPDK_DIR>/TARGET --prefix=/usr --with-rundir=/var/run/openvswitch CFLAGS='-O3 -march-native'
   # make
   # make install
   ``

3. Restart the ovs-dpdk service, and then start the networking-ovs-dpdk-agent.

   ```bash
   # service ovs-dpdk restart
   # screen /usr/bin/networking-ovs-dpdk-agent
   # set-config-file /etc/neutron/neutron.conf
   # set-config-file /etc/neutron/plugins/ml2/ml2_conf.ini
   ``

On all of the nodes, configure the Open vSwitch with the following commands.

4. Define the following variables in the shell environment.

   ```bash
   COLLECTOR_IP=172.16.101.2
   COLLECTOR_PORT=6343
   AGENT_IP=en02
   HEADER_BYTES=128
   SAMPLING_N=64
   POLLING_SECS=10
   ``

   **Note:** COLLECTOR_IP is the IP address of the management interface on the controller node. AGENT_IP is the name of the management interface of the currently configured node.

5. Enable the sFlow agent on all the Open vSwitch bridges. The following command creates the sFlow agent and assigns it to the br-int bridge. This command also returns the universally unique identifier (UUID) of the sFlow agent, which may be reused for other Open vSwitch bridges.

   ```bash
   # ovs-vsctl -- --id=@sflow create sflow agent=${AGENT_IP} \
   target="${COLLECTOR_IP}:${COLLECTOR_PORT}" header=${HEADER_BYTES} \
   sampling=${SAMPLING_N} polling=${POLLING_SECS} -- set bridge br-int sflow=@sflow
   ``

   a733c94f-0364-43f2-9c14-a5a163b4add8

6. Use the following command to enable the sFlow agent on all the remaining bridges.

   ```bash
   # ovs-vsctl set bridge BR_NAME sflow=RETURNED_SFLOW_UUID
   ``

   The commands below present the exemplary executions.

   ```bash
   # ovs-vsctl set bridge br-ex sflow=a733c94f-0364-43f2-9c14-a5a163b4add8
   # ovs-vsctl set bridge br-vlan sflow=a733c94f-0364-43f2-9c14-a5a163b4add8
   ``

   **Note:** Repeat the above steps on each host where the sFlow Open vSwitch agent must be enabled.

3.6.2 Install the InMon sFlowTrend* Data Collector

Perform the following steps on the controller node.

1. Download the installation script.

   ```bash
   ``

2. Set the execution permissions, and then run the script.

   ```bash
   # chmod +x sFlowTrend-unix-6_2.sh
   # ./sFlowTrend-unix-6_2.sh
   ``

3. Follow the installation steps displayed on the screen.

4. Once the installation is completed, use the web browser to navigate to the InMon sFlowTrend* dashboard at `http://CONTROLLER_IP:8087/sflowtrend`.

5. The InMon sFlowTrend dashboard should be accessible and show all the collected network statistics from the Open vSwitch bridges and interfaces, as shown in Figure 4.

   **Figure 4.** InMon sFlowTrend* monitor dashboard.
4.0 Performance Optimizations

This chapter provides the optimization instructions that enable the NFVI to operate with optimal performance.

4.1 Optimize the Host

4.1.1 Isolate the CPU Cores

First, isolate the CPU cores from the Linux scheduler so that the OS cannot use it for housekeeping or other OS-related tasks. These isolated cores can then be dedicated to the Open vSwitch, DPDK poll mode drivers (PMDs), and OpenStack VMs.

Optimal performance is achieved when CPU cores that are isolated and assigned to the Open vSwitch, PMD threads, OpenStack VMs, memory banks, and the NIC, are connected to the same NUMA node. This helps avoid the usage of costly cross-NUMA node links and therefore boosts the performance.

To check what NUMA node the NIC is connected to, execute the following command.

```
# cat /sys/class/net/<interface_name>/device/numa_node
```

The output of this command indicates the NUMA node number, 0 or 1, in case of a two-socket system.

To list the associations between the CPU cores and NUMA nodes, execute the following commands.

```
# yum install numactl
# numactl –hardware
```

The output of the command shows that cores 0–9 and 20–29 belong to the NUMA node 0, and cores 10–19 and 30–39 belong to the NUMA node 1.

**Note:** The CPU cores belonging to the NUMA node 1 are not listed in the Table 6 as these cores were not used in the PoC. However, these cores can be assigned to the OpenStack Compute service in order to allocate more resources to OpenStack VMs.

Intel HT Technology increases the number of independent instructions in the CPU pipeline, because when it is enabled, every single physical CPU core appears as two virtual processors in the OS. These virtual processors are referred to as hyper-threaded or logical cores (LCs). Two hyper-threaded cores that belong to the same physical core are called sibling cores. In this setup, there is an offset of 20 between each of the sibling cores, as presented in Figure 5.

To achieve the optimal performance of DPDK PMD threads, three CPU pinning alternatives were tested:

- Two LCs (LC2 and LC22) on same physical core assigned to DPDK PMD threads (0x400004).
- One LC (LC2) assigned to one PMD thread (0x4).
- Two LCs (LC2 and LC3) on different physical cores assigned to PMD threads (0xC).

Figure 5 shows the graphical interpretation of these alternatives. The hexadecimal values are the masks that denote which virtual cores were assigned to PMD threads. This notation is often used as a parameter passed to a configuration file or a command. Table 6 shows that physical cores 2 and 3 were selected due to the best performance achieved.

<table>
<thead>
<tr>
<th>NUMA NODE 0 CPU CORES</th>
<th>ASSIGNED TO</th>
<th>CONFIGURATION SETTINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 20</td>
<td>Housekeeping</td>
<td>Set the parameters below in the <code>/etc/default/grub</code> file. GRUB_CMDLINE_LINUX = ... isolcpus=1-19,21-39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alternatively, set the parameters below in the <code>/etc/default/ovs-dpdk</code> file. OVS_PMD_CORE_MASK=C</td>
</tr>
<tr>
<td>2, 3</td>
<td>OVS-DPDK PMD threads</td>
<td>Set the mask below in the <code>/etc/default/ovs-dpdk</code> file. OVS_PMD_CORE_MASK=C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alternatively, execute the following command. ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=C</td>
</tr>
<tr>
<td>1</td>
<td>ovs-vswitchd daemon</td>
<td>Set the mask below in the <code>/etc/default/ovs-dpdk</code> file. OVS_CORE_MASK=2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set the CPU core numbers in the <code>/etc/nova/nova.conf</code> file. vcpu_pin_set = 4-9,24-29</td>
</tr>
</tbody>
</table>
### 4.1.2 Enable 1 GB Huge Pages

1 GB huge pages were used for OpenStack VMs to reduce translation lookaside buffer (TLB) misses by the memory management hardware and the CPU on x86_64 architectures. To enable 1 GB huge pages, execute the following steps on all the compute nodes.

1. Add the following line to the `/etc/libvirt/qemu.conf` file.
   ```
   hugetlbfs_mount="/mnt/huge"
   ```

2. Add the following line in the `/etc/fstab` file.
   ```
   hugetlbfs /mnt/huge hugetlbfs defaults 0 0
   ```

3. Create the mount directory for huge pages.
   ```bash
   # mkdir -p /mnt/huge
   ```

4. Add the following line to the `/etc/sysctl.conf` file.
   ```
   vm.nr_hugepages = 60
   ```

5. Edit the `/etc/default/grub` file to set the huge pages.
   ```
   GRUB_CMDLINE_LINUX="… hugepagesz=1G hugepages=60 default_hugepagesz=1G"
   ```

6. Update the GRUB2 configuration.
   ```bash
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

   **Note:** The `grub.cfg` file location may vary. You can use the following command to locate it.
   ```bash
   # locate grub.cfg
   ```

7. Reboot the host machine.
   ```bash
   # reboot
   ```

8. Verify the settings.
   ```bash
   # cat /proc/meinfo | grep Huge
   AnonHugePages:          0 kB
   HugePages_Total:       60
   HugePages_Free:        44
   HugePages_Rsvd:        0
   HugePages_Surp:        0
   Hugepagesize:     1048576 kB
   ```

   ```bash
   # dmesg | grep -o “isolcpus.*”
   isolcpus=2-19,22-39 iommu=on
   isolcpus=2-19,22-39 iommu=on
   ```

### 4.1.3 Enable the TCP Segmentation Offload in OVS-DPDK

We implemented and published a patch to enable TCP segmentation offload (TSO) support in OVS-DPDK. The patch enables successful feature negotiation of TSO (and implicitly, transmit checksum offloading) between the hypervisor and the OVS-DPDK vHost-user back end so that TSO may be enabled on a per-port basis in the VM using the standard Linux `ethtool` utility. Furthermore, the patch also increases the maximum permitted frame length for OVS-DPDK-netdevs to 64 KB—a necessity to accommodate oversized frames received—and introduces the support for handling "offload" frames.

Note that the TSO feature in OVS-DPDK is experimental. It is only validated on OpenStack-deployed flat and VLAN networks. The guest may only take advantage of TSO if OVS is connected to a NIC that supports that functionality. The mechanism by which offloading was achieved works as follows: When OVS dequeues a frame from a TSO-enabled guest port using the DPDK vHost library, the library sets specific offload flags in the metadata that DPDK uses to represent a frame (known as ‘mbuf’). Upon receipt of an offload mbuf, Open vSwitch sets additional offload flags and attribute values in the mbuf before passing it to the DPDK NIC driver for transmission. The driver examines and interprets the mbuf’s offload flags and corresponding attributes to facilitate TCP segmentation on the NIC.

With the enablement of TSO for OVS-DPDK-netdevs in Open vSwitch, the segmentation of guest-originated, oversized TCP frames moves from the guest operating system’s software TCP/IP stack to the NIC hardware. The benefits of this approach are many. First, offloading segmentation of a guest’s TCP frames to hardware significantly reduces the compute burden on the VM’s virtual CPU. Consequently, when the guest does not need to segment frames itself, its virtual CPU can take advantage of the additionally available computational cycles to perform more meaningful work.
Second, with TSO enabled, Open vSwitch does not need to receive, process, and transmit a large number of smaller frame segments, but rather a smaller amount of significantly larger frames. In other words, the same amount of data can be handled with significantly reduced overhead. Finally, the reduction in the number of small packets, which are sent to the NIC for transmission, results in the reduction of PCI bandwidth usage. The cumulative effect of these enhancements is a massive improvement in TCP throughput for DPDK-accelerated Open vSwitch.

To enable TSO in OVS-DPDK, execute the following steps.

1. Stop the ovs-dpdk service.
   ```
   # service ovs-dpdk stop
   ```

2. Unload the igb_uio module.
   ```
   # rmmod igb_uio
   ```

3. Change the directory to the source directory of Open vSwitch.
   ```
   # cd ~/ovs
   ```

4. Check out the TSO patch with a compatible commit.
   ```
   # git checkout cae7529c16e312524bc6b76182e080c97428e2e0
   Note: This will change the Open vSwitch version to 2.5.90.
   ```

5. Download the TCP segmentation patch from the ovs-dev mailing list at https://mail.openvswitch.org/pipermail/ovs-dev/2016-June/316414.html, and then apply the patch.
   ```
   # git am 0001-netdev-dpdk-add-TSO-support-for-vhostuser-ports.patch
   ```
   Alternatively, use the command below.
   ```
   # git apply 0001-netdev-dpdk-add-TSO-support-for-vhostuser-ports.patch
   ```

6. Check out the DPDK v16.04, which is required to use the TSO feature.
   ```
   # cd ~/dpdk
   # git checkout v16.04
   ```

7. Recompile the DPDK libraries.
   ```
   # make install T=x86_64-native-linuxapp-gcc DESTDIR=install
   ```

8. Recompile, and then reinstall the Open vSwitch.
   ```
   # cd ~/ovs
   # ./boot.sh
   # ./configure --with-dpdk=<DPDK_DIR>/<TARGET> --prefix=/usr --with-rundir=/var/run/openvswitch CFLAGS='=-O3 -march-native'
   # make
   # make install
   ```

9. Load the igb_uio driver.
   ```
   # insmod /dpdk/x86_64-native-linuxapp-gcc/kmod/igb_uio.ko
   ```

10. Bind the network interfaces to the igb_uio driver as described in section 3.4.3 Install the OVS-DPDK.

11. Restart the ovs-dpdk service, and then run the networking-ovs-dpdk agent.
    ```
    # service ovs-dpdk restart
    # screen /usr/bin/networking-ovs-dpdk-agent \
    --config-file /etc/neutron/neutron.conf \
    --config-file /etc/neutron/plugins/ml2/ml2_conf.ini
    ```

12. Enable the offload features in qemu-kvm wrapper. Edit the /usr/bin/qemu-kvm file, and change the following line
    ```
    VIRTIO_OPTIONS="csum=off,gso=off,guest_tso4=off,guest_tso6=off,guest_ecn=off,guest_ufo=off"
    ```
    with the line below.
    ```
    VIRTIO_OPTIONS="csum=on,gso=on,guest_tso4=on,guest_tso6=on,guest_ecn=on,guest_ufo=on"
    ```

### 4.1.4 Enable the Multiqueue Feature for vHost-user and Physical DPDK Interfaces

1. Enable multiple queues in the qemu-kvm wrapper. Edit the /usr/bin/qemu-kvm file on both compute nodes. Add multiqueue settings in the following lines.
   ```
   ...
   VIRTIO_OPTIONS="...,mq=on,vectors=18"
   VHOST_FORCE="...,queues=8"
   ...
   ```
   Note: The value of vectors parameter must be equal to 2 × queues + 2.

2. Configure the number of DPDK queues in the Open vSwitch to increase the network performance. In the Open vSwitch 2.5.0 and older, this option is set globally with the command below.
   ```
   # ovs-vsctl set Open_vSwitch . other_config:dpdk-rxqs=8
   ```
   In the Open vSwitch 2.5.90 or newer, this option is configured on a per-port basis.

   It was reported that the multiquesue feature used with OpenStack Kilo and Open vSwitch 2.5.90 or newer is not working properly. The issue appears when the number of queues specified in Quick Emulator* (QEMU*) is higher than the default value, 1.

   To work around the problem, identify the VM’s port ID with the following commands.
   ```
   # nova list
   # neutron port-list | grep –e "20.20.20.105"
   ```
   Make sure to note the first 10 characters of port-id for your desired VM. This string will be used to identify the interface on OVS-DPDK.
Once the port is identified, use the Open vSwitch command line interface and set the options:n_rxq parameter manually on the Open vSwitch interface in order to set a specific number of queues on that interface.

```
# ovs-vsctl set interface vhu9fbcbc75-30 options:n_rxq=8
```

To apply the changes, execute the soft reboot of the instance.

### 4.1.5 Enable Core Pinning and NUMA Awareness in the OpenStack Compute

1. On all of the compute nodes, edit the `/etc/nova/nova.conf` file, and then update the vcpu_pin_set setting.

   ```
   vcpu_pin_set = 4-9,24-29
   ```

2. Restart the `openstack-nova-compute.service`.

   ```
   # systemctl restart openstack-nova-compute.service
   ```

3. On the controller node, create the optimized NUMA-aware OpenStack flavor by specifying the number of CPU cores, memory size, and storage capacity.

   ```
   # nova flavor-create <flavor-name> <id> <mb-of-ram> <gb-of-storage> <cpus-number>
   ```

4. Set the following extra_specs to use the resources from the selected NUMA node.

   ```
   # nova flavor-key <id> set hw:numa_mem.0=<mb-of-ram>
   # nova flavor-key <id> set hw:mem_page_size="large"
   # nova flavor-key <id> set hw:cpu_policy="dedicated"
   # nova flavor-key <id> set hw: numa_mempolicy="strict"
   # nova flavor-key <id> set hw:numa_nodes=1
   # nova flavor-key <id> set hw:numa_cpus.0=0,1,...,<cpus-number>
   # nova flavor-key <id> set hw:cpu_threads_policy="prefer"
   ```

**Note:** The above commands create a flavor utilizing NUMA node 0 resources only. To use the NUMA node 1, set the hw:numa_cpus.1 and hw:numa_mem.1 keys instead of hw:numa_cpus.0 and hw:numa_mem.0 respectively.

### 4.2 Optimize the Guest

#### 4.2.1 The 'extra_specs' Properties for OpenStack VMs

To make use of features like CPU affinity, huge pages, and single NUMA node topology in VMs, we set the 'extra_specs' property applicable to OpenStack Compute flavors. Table 7 shows the extra_specs parameters that were used to instantiate VMs in this setup.

<table>
<thead>
<tr>
<th>EXTRA_SPECS</th>
<th>VALUE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw:cpu_policy</td>
<td>dedicated</td>
<td>Guest virtual CPUs will be strictly pinned to a set of host physical CPUs.</td>
</tr>
<tr>
<td>hw:mem_page_size</td>
<td>large</td>
<td>Guest will use 1 GB HugePages.</td>
</tr>
<tr>
<td>hw:numa_mempolicy</td>
<td>preferred</td>
<td>Memory resources will be provided according to extra_specs, but if more resources are needed, these will be taken from the other NUMA node.</td>
</tr>
<tr>
<td>hw:numa_mem.0</td>
<td>4096</td>
<td>Mapping memory size to the NUMA node 0.</td>
</tr>
<tr>
<td>hw:numa_nodes</td>
<td>1</td>
<td>Number of NUMA nodes to expose to the guest.</td>
</tr>
<tr>
<td>hw:numa_cpus.0</td>
<td>0,1,2,3</td>
<td>Mapping of virtual CPUs list to the NUMA node 0.</td>
</tr>
<tr>
<td>hw:cpu_threads_policy</td>
<td>prefer</td>
<td>If the host has threads, the virtual CPU will be placed on the same core as a sibling core.</td>
</tr>
</tbody>
</table>

### 4.2.2 Enable Multiqueue for VirtIO Interfaces

After enabling the multiqueue feature on the host machine, the same number of queues need to be set inside the VM with the command below.

```
# ethtool -L eth0 combined NR_OF_QUEUES
```

**Note:** The interface name on the virtual machine may be different.

#### 4.2.3 Upgrade the CentOS* 7 Kernel to version 4.5.5 on the Guest

1. Install dependencies.

   ```
   # yum install wget
   # yum install linux-firmware
   ```

2. Download the RPM package of the kernel.

   ```
   # wget http://mirrors.coreix.net/elrepo-archive-archive/kernel/el7/x86_64/RPMS/kernel-ml-4.5.4-1.el7.elrepo.x86_64.rpm
   ```

3. Install the new kernel.

   ```
   # rpm -i kernel-ml-4.5.4-1.el7.elrepo.x86_64.rpm
   ```

4. Optionally, uninstall the old kernel.

   ```
   # rpm -e <kernel-package-name>
   ```

   The package name of an old kernel can be obtained with the command below.

   ```
   # rpm -qa kernel command.
   ```

5. Reboot the VM, and then select the 4.5.4 kernel in the GRUB boot menu if more than one entry is available.

#### 4.2.4 Additional TCP Performance Tunings on the CentOS* 7 Guest

1. Set the tuned-adm profile that can bring further TCP performance improvements.

   ```
   # tuned-adm profile latency-performance
   # tuned-adm profile network-throughput
   # tuned-adm profile throughput-performance
   ```

---

**Table 7. The extra_specs settings for OpenStack Compute flavors.**
Appendix A: The Packstack Answer File

The following is the Packstack answer file that is used in the PoC.

```
[general]
CONFIG_SSH_KEY=/root/.ssh/id_rsa.pub
CONFIG_DEFAULT_PASSWORD=password
CONFIG_MARIADB_INSTALL=y
CONFIG_GLANCE_INSTALL=y
CONFIG_CINDER_INSTALL=y
CONFIG_NOVA_INSTALL=y
CONFIG_NEUTRON_INSTALL=y
CONFIG_MANILA_INSTALL=n
CONFIG_SWIFT_INSTALL=y
CONFIG_CILOMETER_INSTALL=y
CONFIG_HEAT_INSTALL=y
CONFIG_SAHAIRA_INSTALL=n
CONFIG_TROVE_INSTALL=n
CONFIG_IRONIC_INSTALL=n
CONFIG_CLIENT_INSTALL=y
CONFIG_NTP_SERVERS=
CONFIG_NAGIOS_INSTALL=y
EXCLUDE_SERVERS=
CONFIG_DEBUG_MODE=n
CONFIG_CONTROLLER_HOST=172.16.101.2
CONFIG_COMPUTE_HOSTS=172.16.101.3,172.16.101.4
CONFIG_NETWORK_HOSTS=172.16.101.2
CONFIG_UNSUPPORTED=n
CONFIG_USE_SUBNETS=n
CONFIG_VCENTER_HOST=
CONFIG_VCENTER_USER=
CONFIG_VCENTER_PASSWORD=
CONFIG_VCENTER_CLUSTER_NAME=
CONFIG_STORAGE_HOST=172.16.101.2
CONFIG_SAHAIRA_HOST=
CONFIG_USE_EPEL=n
CONFIG_REPO=
CONFIG_ENABLE_RDO_TESTING=n
CONFIG_RH_USER=
CONFIG_SATELLITE_URL=
CONFIG_SATELLITE_PW=
CONFIG_RH_OPTIONAL=y
CONFIG_RH_PROXY=
CONFIG_RH_PROXY_PORT=
CONFIG_RH_PROXY_USER=
CONFIG_RH_PROXY_PW=
CONFIG_SATELLITE_USER=
CONFIG_SATELLITE_PW=
CONFIG_SATELLITE_AKEY=
CONFIG_SATELLITE_CACERT=
CONFIG_SATELLITE_PROFILE=
CONFIG_SATELLITE_FLAGS=
CONFIG_SATELLITE_PROXY=
CONFIG_SATELLITE_PROXY_USER=
CONFIG_SATELLITE_PROXY_PW=
CONFIG_SSL_CACERT_KEY_FILE=/etc/pki/tls/private/selfkey.key
CONFIG_SSL_CERT_DIR=/packstackca/
CONFIG_SSL_CACERT_SELFSIGN=y
CONFIG_SELFSIGN_CACERT_SUBJECT_C=--
CONFIG_SELFSIGN_CACERT_SUBJECT_ST=State
CONFIG_SELFSIGN_CACERT_SUBJECT_L=City
CONFIG_SELFSIGN_CACERT_SUBJECT_O=openstack
CONFIG_SELFSIGN_CACERT_SUBJECT_EXTEN=
CONFIG_AMQP_BACKEND=rabbitmq
CONFIG_AMQP_HOST=172.16.101.2
CONFIG_AMQP_ENABLE_SSL=n
CONFIG_AMQP_ENABLE_AUTH=n
CONFIG_AMQP_NSS_CERTDB_PW=password
CONFIG_AMQP_AUTH_USER=amqp_user
CONFIG_AMQP_AUTH_PASSWORD=password
CONFIG_MARIADB_HOST=172.16.101.2
CONFIG_MARIADB_USER=root
CONFIG_MARIADB_PW=password
CONFIG_KEYSTONE_DB_PW=password
CONFIG_KEYSTONE_REGION=RegionOne
CONFIG_KEYSTONE_ADMIN_TOKEN=9b45b4987ca04d889203287691c11f4
CONFIG_KEYSTONE_ADMIN_EMAIL=root@localhost
CONFIG_KEYSTONE_ADMIN_USERNAME=admin
CONFIG_KEYSTONE_ADMIN_PW=password
CONFIG_KEYSTONE_DEMO_PW=password
CONFIG_KEYSTONE_API_VERSION=v2.0
CONFIG_KEYSTONE_TOKEN_FORMAT=UUID
CONFIG_KEYSTONE_SERVICE_NAME=keystone
CONFIG_KEYSTONE_IDENTITY_BACKEND=sql
CONFIG_KEYSTONE_LDAP_URL=ldap://172.16.101.2
CONFIG_KEYSTONE_LDAP_USER_PW=
CONFIG_KEYSTONE_LDAP_USER_PASSWD=
CONFIG_KEYSTONE_LDAP_SUFFIX=
CONFIG_KEYSTONE_LDAP_QUERY_SCOPE=one
CONFIG_KEYSTONE_LDAP_PAGE_SIZE=-1
CONFIG_KEYSTONE_LDAP_USER_SUBTREE=
CONFIG_KEYSTONE_LDAP_USER_FILTER=
CONFIG_KEYSTONE_LDAP_USER_OBJECTCLASS=
CONFIG_KEYSTONE_LDAP_USER_ID_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_NAME_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_ID_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_NAME_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_ID_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_NAME_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_ENABLED_DEFAULT=True
```
CONFIG_KEYSTONE_LDAP_USER_ENABLED_INVERT=n
CONFIG_KEYSTONE_LDAP_USER_ATTRIBUTE_IGNORE=
CONFIG_KEYSTONE_LDAP_USER_DEFAULT_PROJECT_ID_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_ALLOW_CREATE=n
CONFIG_KEYSTONE_LDAP_USER_ALLOW_UPDATE=n
CONFIG_KEYSTONE_LDAP_USER_ALLOW_DELETE=n
CONFIG_KEYSTONE_LDAP_USER_PASS_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_USER_ENABLED_EMULATION_DN=
CONFIG_KEYSTONE_LDAP_USER_ADDITIONAL_ATTRIBUTE_MAPPING=
CONFIG_KEYSTONE_LDAP_GROUP_SUBTREE=
CONFIG_KEYSTONE_LDAP_GROUP_FILTER=
CONFIG_KEYSTONE_LDAP_GROUP_OBJECTCLASS=
CONFIG_KEYSTONE_LDAP_GROUP_ID_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_GROUP_NAME_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_GROUP_MEMBER_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_GROUP_DESC_ATTRIBUTE=
CONFIG_KEYSTONE_LDAP_GROUP_ATTRIBUTE_IGNORE=
CONFIG_KEYSTONE_LDAP_GROUP_ALLOW_CREATE=n
CONFIG_KEYSTONE_LDAP_GROUP_ALLOW_UPDATE=n
CONFIG_KEYSTONE_LDAP_GROUP_ALLOW_DELETE=n
CONFIG_KEYSTONE_LDAP_GROUP_ADDITIONAL_ATTRIBUTE_MAPPING=
CONFIG_KEYSTONE_LDAP_USE_TLS=n
CONFIG_KEYSTONE_LDAP_TLS_CACERTDIR=
CONFIG_KEYSTONE_LDAP_TLS_CACERTFILE=
CONFIG_KEYSTONE_LDAP_TLS_REQ_CERT=demand
CONFIG_GLANCE_DB_PW=password
CONFIG_GLANCE_KS_PW=password
CONFIG_GLANCE_BACKEND=file
CONFIG_CINDER_DB_PW=2e6edd7b3d834fe1
CONFIG_CINDER_KS_PW=b7c18d9ac44d4f67
CONFIG_CINDER_BACKEND=lvm
CONFIG_CINDER_VOLUMES_CREATE=y
CONFIG_CINDER_VOLUMES_SIZE=20G
CONFIG_CINDER_GLUSTER_MOUNTS=
CONFIG_CINDER_NFS_MOUNTS=
CONFIG_CINDER_NETAPP_LOGIN=
CONFIG_CINDER_NETAPP_PASSWORD=
CONFIG_CINDER_NETAPP_HOSTNAME=
CONFIG_CINDER_NETAPP_SERVER_PORT=80
CONFIG_CINDER_NETAPP_STORAGE_FAMILY=ontap_cluster
CONFIG_CINDER_NETAPP_TRANSPORT_TYPE=http
CONFIG_CINDER_NETAPP_STORAGE_PROTOCOL=nfs
CONFIG_CINDER_NETAPP_SIZE_MULTIPLIER=1.0
CONFIG_CINDER_NETAPP_EXPIRY_THRES_MINUTES=720
CONFIG_CINDER_NETAPP_THRES_AVL_SIZE_PERC_START=20
CONFIG_CINDER_NETAPP_THRES_AVL_SIZE_PERC_STOP=60
CONFIG_CINDER_NETAPP_NFS_SHARES=
CONFIG_CINDER_NETAPP_NFS_SHARES_CONFIG=/etc/cinder/shares.conf
CONFIG_CINDER_NETAPP_VOLUME_LIST=
CONFIG_CINDER_NETAPP_VFILER=
CONFIG_CINDER_NETAPP_PARTNER_BACKEND_NAME=
CONFIG_CINDER_NETAPP_VSERVER=
CONFIG_CINDER_NETAPP_CONTROLLER_IPS=
CONFIG_CINDER_NETAPP_SA_PASSWORD=
CONFIG_CINDER_NETAPP_ESERIES_HOST_TYPE=linux_dm_mp
CONFIG_CINDER_NETAPP_WEBSERVICE_PATH=/devmgr/v2
CONFIG_CINDER_NETAPP_STORAGE_POOLS=
CONFIG_MANILA_DB_PW=password
CONFIG_MANILA_KS_PW=password
CONFIG_MANILA_BACKEND=generic
CONFIG_MANILA_NETAPP_DRV_HANDLES_SHARESERVERS=false
CONFIG_MANILA_NETAPP_TRANSPORT_TYPE=https
CONFIG_MANILA_NETAPP_LOGIN=admin
CONFIG_MANILA_NETAPP_PASSWORD=
CONFIG_MANILA_NETAPP_SERVER_HOSTNAME=
CONFIG_MANILA_NETAPP_STORAGE_FAMILY=ontap_cluster
CONFIG_MANILA_NETAPP_SERVER_PORT=443
CONFIG_MANILA_NETAPP_AGGRATE_NAME_SEARCH_PATTERN=.*
CONFIG_MANILA_NETAPP_ROOT_VOLUME_NAME=root
CONFIG_MANILA_NETAPP_ROOT_VOLUME_AGGRATE=
CONFIG_MANILA_NETAPP_VSERVER=
CONFIG_MANILA_GENERIC_DRV_HANDLES_SHARESERVERS=true
CONFIG_MANILA_GENERIC_VOLUME_NAME_TEMPLATE=manila-share-%s
CONFIG_MANILA_GENERIC_SHARE_MOUNT_PATH=/shares
CONFIG_MANILA_SERVICE_IMAGE_LOCATION=https://www.dropbox.com/s/vi5oeh10glqckckh/ubuntu_1204_nfs_cifs.qcow2
CONFIG_MANILA_NETWORK_TYPE=neutron
CONFIG_MANILA_NETWORK_STANDALONE_GATEWAY=
CONFIG_MANILA_NETWORK_STANDALONE_NETMASK=
CONFIG_MANILA_NETWORK_STANDALONE_SEG_ID=
CONFIG_MANILA_NETWORK_STANDALONE_IP_RANGE=
CONFIG_MANILA_NETWORK_STANDALONE_IP_VERSION=4
CONFIG_IRONIC_DB_PW=password
CONFIG_IRONIC_KS_PW=password
CONFIG_NOVA_DB_PW=password
CONFIG_NOVA_KS_PW=password
CONFIG_NOVA_SCHED_CPU_ALLOC_RATIO=16.0
CONFIG_NOVA_SCHED_RAM_ALLOC_RATIO=1.5
CONFIG_NOVA_COMPUTE_MIGRATE_PROTOCOL=tcp
CONFIG_NOVA_COMPUTE_MANAGER=nova.compute.manager.ComputeManager
CONFIG_VNC_SSL_CERT=
CONFIG_VNC_SSL_KEY=
CONFIG_NOVA_COMPUTE_PRIVIF=eno2
CONFIG_NOVA_NETWORK_MANAGER=nova.network.manager.FlatDHCPManager
CONFIG_NOVA_NETWORK_PUBIF=eno1
CONFIG_NOVA_NETWORK_PRIVIF=eno2
CONFIG_NOVA_NETWORK_FIXEDRANGE=192.168.32.0/22
CONFIG_NOVA_NETWORK_FLOATRANGE=10.3.4.0/22
CONFIG_NOVA_NETWORK_AUTOASSIGNFLOATINGIP=n
CONFIG_NOVA_NETWORK_VLAN_START=100
CONFIG_NOVA_NETWORK_NUMBER=1
CONFIG_NOVA_NETWORK_SIZE=255
CONFIG_NEUTRON_KS_PW=password
CONFIG_NEUTRON_DB_PW=password
CONFIG_NEUTRON_L3_EXT_BRIDGE=br-ex
CONFIG_NEUTRON_METADATA_PW=password
CONFIG_LBAAS_INSTALL=n
CONFIG_NEUTRON_METERING.Agent_INSTALL=n
CONFIG_NEUTRON_FWAAS=n
CONFIG_NEUTRON_ML2_TYPE_DRIVERS=vxlan
CONFIG_NEUTRON_ML2_TENANT_NETWORK_TYPES=vxlan
CONFIG_NEUTRON_ML2_MECHANISM_DRIVERS=openvswitch
CONFIG_NEUTRON_ML2_FLAT_NETWORKS=external
CONFIG_NEUTRON_ML2_VLAN RANGES=physnet1:12:1:000
CONFIG_NEUTRON_ML2_TUNNEL_ID_RANGES=
CONFIG_NEUTRON_ML2_VXLAN_GROUP=239.1.1.1:100
CONFIG_NEUTRON_ML2_VNI RANGES=1001:2000
CONFIG_NEUTRON_L2_AGENT=openvswitch
CONFIG_NEUTRON_L2_INTERFACE_MAPPINGS=
CONFIG_NEUTRON_OVS_BRIDGE MAPPINGS=external:br-ex,physnet1:br-vlan
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:eno1,br-vlan:eno4
CONFIG_NEUTRON_OVS_TUNNEL_IF=eno3
CONFIG_NEUTRON_OVS_VXLAN_UDP_PORT=4789
CONFIG_HORIZON_SSL=n
CONFIG_HORIZON_SECRET_KEY=3bd14fee29614097
CONFIG_HORIZON_KS_PW=password
CONFIG_HORIZON_CLOUDWATCH_INSTALL=n
CONFIG_HORIZON_CFN_INSTALL=n
CONFIG_HORIZON_DOMAIN=heat
CONFIG_HORIZON_DOMAIN_ADMIN=heat_admin
CONFIG_HORIZON_DOMAIN_PASSWORD=password
CONFIG_PROVISION_DEMO=n
CONFIG_PROVISION_TEMPEST=n
CONFIG_PROVISION_DEMO FLOATRANGE=172.24.4.224/28
CONFIG_PROVISION_TEMPEST_REPO_URI=https://github.com/openstack/tempest.git
CONFIG_PROVISION_TEMPEST_REVISION=master
CONFIG_PROVISION_ALL_IN_ONE_OVS_BRIDGE=n
CONFIG_CEILOMETER_SECRET=3bd14fee29614097
CONFIG_CEILOMETER_KS_PW=c7836e856272421a
CONFIG_CEILOMETERCOORDINATION_BACKEND=redis
CONFIG_MONGODB_HOST=172.16.101.2
CONFIG_REDIS_MASTER_HOST=172.16.101.2
CONFIG_REDIS_MASTER_NAME=mymaster
CONFIG_SAHARA_DB_PW=PW_PLACEHOLDER
CONFIG_SAHARA_KS_PW=PW_PLACEHOLDER
CONFIG_TROVE_DB_PW=PWPLACEHOLDER
CONFIG_TROVE_KS_PW=PWPLACEHOLDER
CONFIG_TROVE_NOVA_USER=heat
CONFIG_TROVE_NOVA_TENANT=heat
CONFIG_TROVE_NOVA_PASSWORD=password
CONFIG_NAGIOS_PW=password
CONFIG_SWIFT_STORAGE_ZONES=1
CONFIG_SWIFT_STORAGE_REPLICAS=1
CONFIG_SWIFT_STORAGE_FSTYPE=ext4
CONFIG_SWIFT_HASH=6ec5f52eb214f4a
CONFIG_SWIFT_STORAGE_SIZE=2G
CONFIG_HEAT_DB_PW=password
CONFIG_HEAT_AUTH_ENC_KEY=2fffb5d89b5947a7
CONFIG_HEAT_KS_PW=password
CONFIG_HEAT_CLOUDWATCH_INSTALL=n
CONFIG_HEAT_CFNMOUNT_INSTALL=n
CONFIG_HEAT_DOMAIN=heat
CONFIG_HEAT_DOMAIN_ADMIN=heat_admin
CONFIG_HEAT_DOMAIN_PASSWORD=password
CONFIG_PROVISION_DEMO=n
CONFIG_PROVISION_TEMPEST=n
CONFIG_PROVISION_FLOATRANGE=172.24.4.224/28
CONFIG_PROVISION_IMAGE_NAME=cirros
CONFIG_PROVISION_IMAGE_URL=http://download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-disk.img
CONFIG_PROVISION_IMAGE_FORMAT=qcow2
CONFIG_PROVISION_IMAGE_SSL_USER=cirros
CONFIG_PROVISION_TEMPEST_USER=
CONFIG_PROVISION_TEMPEST_USER_PW=PW_PLACEHOLDER
CONFIG_PROVISION_ALL_IN_ONE_OVS_BRIDGE=n
CONFIG_CEILOMETER_SECRET=3bd14fee29614097
CONFIG_CEILOMETER_KS_PW=c7836e856272421a
CONFIG_CEILOMETERCOORDINATION_BACKEND=redis
CONFIG_MONGODB_HOST=172.16.101.2
CONFIG_REDIS_MASTER_HOST=172.16.101.2
CONFIG_REDIS_MASTER_NAME=mymaster
CONFIG_SAHARA_DB_PW=PW_PLACEHOLDER
CONFIG_SAHARA_KS_PW=PW_PLACEHOLDER
CONFIG_TROVE_DB_PW=PWPLACEHOLDER
CONFIG_TROVE_KS_PW=PWPLACEHOLDER
CONFIG_TROVE_NOVA_USER=heat
CONFIG_TROVE_NOVA_TENANT=heat
CONFIG_TROVE_NOVA_PASSWORD=password
CONFIG_NAGIOS_PW=password
Appendix B: Create OpenStack Networks and Router

Execute the following commands to create the OpenStack public network.

```bash
# neutron net-create public
--provider:network_type flat
--provider:physical_network external
--router:external=True
# neutron subnet-create public
10.2.125.0/24 --name public_subnet
--gateway 10.2.125.1 --allocation-pool
start=10.2.125.236,end=10.2.125.240
--enable-dhcp=False
```

Execute the following commands to create the OpenStack private network.

```bash
# neutron net-create private
# neutron subnet-create private
172.16.111.0/24
--name private_subnet
```

Execute the following commands to create the OpenStack router, add an interface to the private subnet, and set the default gateway.

```bash
# neutron router-create myrouter
# neutron router-gateway-set myrouter
public
# neutron router-interface-add myrouter
private_subnet
```

Execute the following commands to create the VLAN network with the VLAN identifier set to 2.

```bash
# neutron net-create net-vlan-2
--provider:network_type vlan
--provider:physical_network physnet1
--provider:segmentation_id 2
--router:external --shared
# neutron subnet-create net-vlan-2
20.20.20.0/24
--name net-vlan2-subnet --no-gateway
--allocation-pool
start=20.20.20.10,end=20.20.20.100
--enable-dhcp=False
```

The following table explains the values of parameters used in the commands above.

<table>
<thead>
<tr>
<th>ABBREVIATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Name of the external/public network</td>
</tr>
<tr>
<td>flat</td>
<td>Network type for the public network</td>
</tr>
<tr>
<td>external</td>
<td>Physical network for the public network</td>
</tr>
<tr>
<td>10.2.125.0/24</td>
<td>CIDR of the public network</td>
</tr>
<tr>
<td>public_subnet</td>
<td>Name of the public subnet</td>
</tr>
<tr>
<td>10.2.125.1</td>
<td>IP address of the gateway for the public subnet</td>
</tr>
<tr>
<td>10.2.125.236</td>
<td>Starting IP address in the allocation pool for the public subnet</td>
</tr>
<tr>
<td>10.2.125.240</td>
<td>The last IP address in the allocation pool for the public subnet</td>
</tr>
<tr>
<td>private</td>
<td>Name of the private network</td>
</tr>
<tr>
<td>172.16.111.0/24</td>
<td>CIDR of the private network</td>
</tr>
<tr>
<td>private_subnet</td>
<td>Name of the private subnet</td>
</tr>
<tr>
<td>myrouter</td>
<td>Name of the OpenStack router</td>
</tr>
<tr>
<td>net-vlan-2</td>
<td>Name of the VLAN network</td>
</tr>
<tr>
<td>physnet1</td>
<td>Physical network for the VLAN type network</td>
</tr>
<tr>
<td>2</td>
<td>VLAN identifier of VLAN network</td>
</tr>
<tr>
<td>net-vlan2-subnet</td>
<td>Name of the VLAN subnet</td>
</tr>
<tr>
<td>20.20.20.0/24</td>
<td>CIDR of the VLAN subnet</td>
</tr>
<tr>
<td>20.20.20.10</td>
<td>Starting IP in the allocation pool for the VLAN subnet</td>
</tr>
<tr>
<td>20.20.20.100</td>
<td>The last IP address in the allocation pool for the VLAN subnet</td>
</tr>
</tbody>
</table>
Appendix C: Basic Configuration of Brocade 5600 vRouter

The following steps describe how to configure the Brocade 5600 vRouter VM.

1. Download the Brocade 5600 vRouter qcow2 image file. The command below can be used to import this image to the OpenStack Image Service.

   ```
   # glance image-create --name="Brocade 5600 vRouter" --is-public=true --container-format=bare --disk-format=qcow2 --progress < vyatta_vrouter.qcow2
   ```

2. Use the OpenStack dashboard to spawn the Brocade 5600 vRouter VM on all the desired subnets.

3. Once the Brocade 5600 vRouter VM is booted, connect to its console using the OpenStack dashboard. Use the credentials vyatta/vyatta to log in to the router VM.

   In the following example, the Brocade 5600 vRouter VM is spawned on VLAN2 and public networks, and have two interfaces, dp0s4 and dp0s6. The interface dp0s4 is directly connected to the VLAN 2 subnet with an IP address of 20.20.20.10/24. The interface dp0s6 is directly connected to the public network and have an IP address of 10.250.100.236.

   ```
   vyatta@R1# set system name-server 8.8.4.4
   vyatta@R1# set system name-server 4.2.2.2
   vyatta@R1# set system name-server 4.2.2.1
   vyatta@R1# set interfaces dataplane dp0s6 address 10.250.100.236/24
   vyatta@R1# set service nat source rule 10 source address 20.20.20.0/24
   [edit]
   vyatta@R1# set service nat source rule 10 outbound-interface dp0s6
   [edit]
   vyatta@R1# set service nat source rule 10 translation address masquerade
   [edit]
   vyatta@R1# commit
   [edit]
   vyatta@R1# set protocols static route 0.0.0.0/0 next-hop 10.250.100.1 interace dp0s6
   vyatta@R1# commit
   [edit]
   ```

   The commands above set:

   - Public DNS, that is, 8.8.4.4, 4.2.2.2, and 4.2.2.1, in the router VM
   - Network address translation rules for the VLAN2 network
   - Default gateway in the router to 10.250.100.1 to provide Internet access to the VMs spawned on VLAN2 network

Appendix D: Abbreviations

<table>
<thead>
<tr>
<th>PARAMETER Value</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOS</td>
<td>Basic Input/Output System</td>
</tr>
<tr>
<td>CIDR</td>
<td>Classless Inter-Domain Routing</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial Off-the-Shelf</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DPDK</td>
<td>Data Plane Development Kit</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>LC</td>
<td>Logical Core</td>
</tr>
<tr>
<td>netdev</td>
<td>Network Device</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Functions Virtualization</td>
</tr>
<tr>
<td>NFVI</td>
<td>NFV Infrastructure</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NUMA</td>
<td>Non-Uniform Memory Architecture</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OVS-DPDK</td>
<td>DPDK-Accelerated Open vSwitch</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral Component Interconnect</td>
</tr>
<tr>
<td>PMD</td>
<td>Poll Mode Driver</td>
</tr>
<tr>
<td>PoC</td>
<td>Proof-of-Concept</td>
</tr>
<tr>
<td>QEMU</td>
<td>Quick Emulator</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLB</td>
<td>Transaction Lookaside Buffer</td>
</tr>
<tr>
<td>TSO</td>
<td>TCP Segmentation Offload</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UUID</td>
<td>Universally Unique Identifier</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual Local Area Network</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtual Network Function</td>
</tr>
<tr>
<td>vSwitch</td>
<td>Virtual Switch</td>
</tr>
<tr>
<td>VxLAN</td>
<td>Virtual eXtensible LAN</td>
</tr>
</tbody>
</table>
By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer. Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses. Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade. Intel does not control or audit third-party websites, software, data or other information referenced in this document. You should contact each third party to confirm whether the referenced data is accurate. No endorsement, sponsorship by, or association with Intel and any third parties is expressed nor should be inferred from references to third parties and their products and services in this document.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights. Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others. © 2016 Intel Corporation.