IMPLEMENTATION GUIDE

Enhanced Platform Awareness Performance Benefits using RIFT.ware*

Intel Corporation
Datacenter Network
Solutions Group

Authors

Shivapriya Hiremath
Solution Software Engineer,
Intel Corporation

Jason Xia
Solution Software Engineer,
Intel Corporation

1.0 Introduction

RIFT.io is a Burlington-based networking startup that developed the RIFT.ware* network service (NS) virtualization platform. RIFT.ware delivers management and orchestration (MANO) and automation of virtual network services, applications, and functions with scale. RIFT.ware is a model-driven, European Telecommunications Standards Institute-compliant network functions virtualization (NFV) MANO solution that serves as a common MANO platform across multiple cloud platforms (i.e., NFV infrastructure) and multi-vendor network functions and services.

Intel Corporation and RIFT.io have deep collaboration in the field of NFV MANO. In this document, we present the three use case demonstrations of RIFT.io MANO with the OpenStack* as virtualized infrastructure manager used to deploy virtual network functions (VNFs) on Intel® architecture platforms. We compare the scalability and performance of platforms that are optimized with Enhanced Platform Awareness (EPA) features and technologies, with the platforms that are not optimized for performance.

Enhanced Platform Awareness (EPA) is an umbrella term for contributions provided by Intel Corporation and others to the OpenStack. EPA enables the service provider to benefit from the underlying platform capabilities by:

- Providing OpenStack a better view of underlying hardware capabilities.
- OpenStack intelligence to filter and match platforms with specific capabilities.
- Matching virtual machine (VM) workload to platform capabilities.

In this document, we showcase the performance benefits of EPA with the following demonstrations:

- **Demo 1: Throughput with and without EPA-based optimizations and Data Plane Development Kit-accelerated Open vSwitch* (OVS-DPDK*)**. We achieved **5×** higher throughput in packet forwarding using optimal Data Plane Development Kit* (DPDK*) deployments over deployments without DPDK.
- **Demo 2: IP Security (IPSec) performance measurement with and without the Intel® QuickAssist Technology**. In our setup, Intel QuickAssist Technology improved the Internet Key Exchange (IKE) rekey rate by **20×**, and enabled to use **20×** more IPSec tunnels.
- **Demo 3: Performance benefit with Cache Allocation Technology (CAT)**. Noise workloads cause cache conflicts to VNFs (RIFT.ware Trafgen* and RIFT.ware Trafsink*) along with the application-level impact. We show how cache conflicts are resolved with CAT and more intelligent placement of workload.

The audiences of this document are network administrators, cloud architects, software-defined networking (SDN) and NFV strategists, VNF suppliers and builders, and other influencers at network suppliers, cloud and telecommunications service providers, and enterprises.

More information on performance optimizations of Intel® architecture servers with EPA features can be found at https://networkbuilders.intel.com/network-technologies/solution-blueprints.

Table of Contents

2.0 Overview

To show how various technologies can speed up the NFV infrastructures, we set up three demonstrations. Table 1 presents mapping of the selected technologies or EPA features to these demonstrations. We used RIFT.ware* 4.2.1 software components supplied by RIFT.io for VNFs.

Table 1. Overview on the technologies and EPA features used in the demonstrations.

TECHNOLOGY / EPA FEATURE		DEMO 1A	DEMO 1B	DEMO 2A	DEMO 2B	DEMO 3
Intel® Ethernet Server Adapter X520-DA2		√	√	√	√	√
Intel® Ethernet Converged Network Adapter XL710-QDA2						√
CPU Optimization	Huge pages		√	√	√	√
	CPU Pinning		√	√	√	√
	CPU Thread Pinning		√	√	√	√
Data Plane Development Kit (DPDK)			√	√	√	√
Open vSwitch* acceleration			√			
Intel® Data Direct I/O Technology (Intel® DDIO)			√	√	√	√
Single Root I/O Virtualization (SR-IOV)						√
Intel® QuickAssist Technology					√	
Cache Allocation Technology (CAT)						√

For each demonstration, we used one OpenStack controller node, one or more OpenStack compute node, and the Cobbler* server that is used for installing the software stacks on the host machines (i.e., setting up the controller and compute nodes). Figure 1 shows how each demonstration is deployed.

Table 2 presents in detail the most important hardware components of each server. The host machines marked orange are the controller nodes and the Cobbler installation server. The host machines marked blue, green, and yellow are the compute nodes used in Demo 1, Demo 2, and Demo 3, respectively. The port numbers that were used on the switches are denoted with white rectangles.

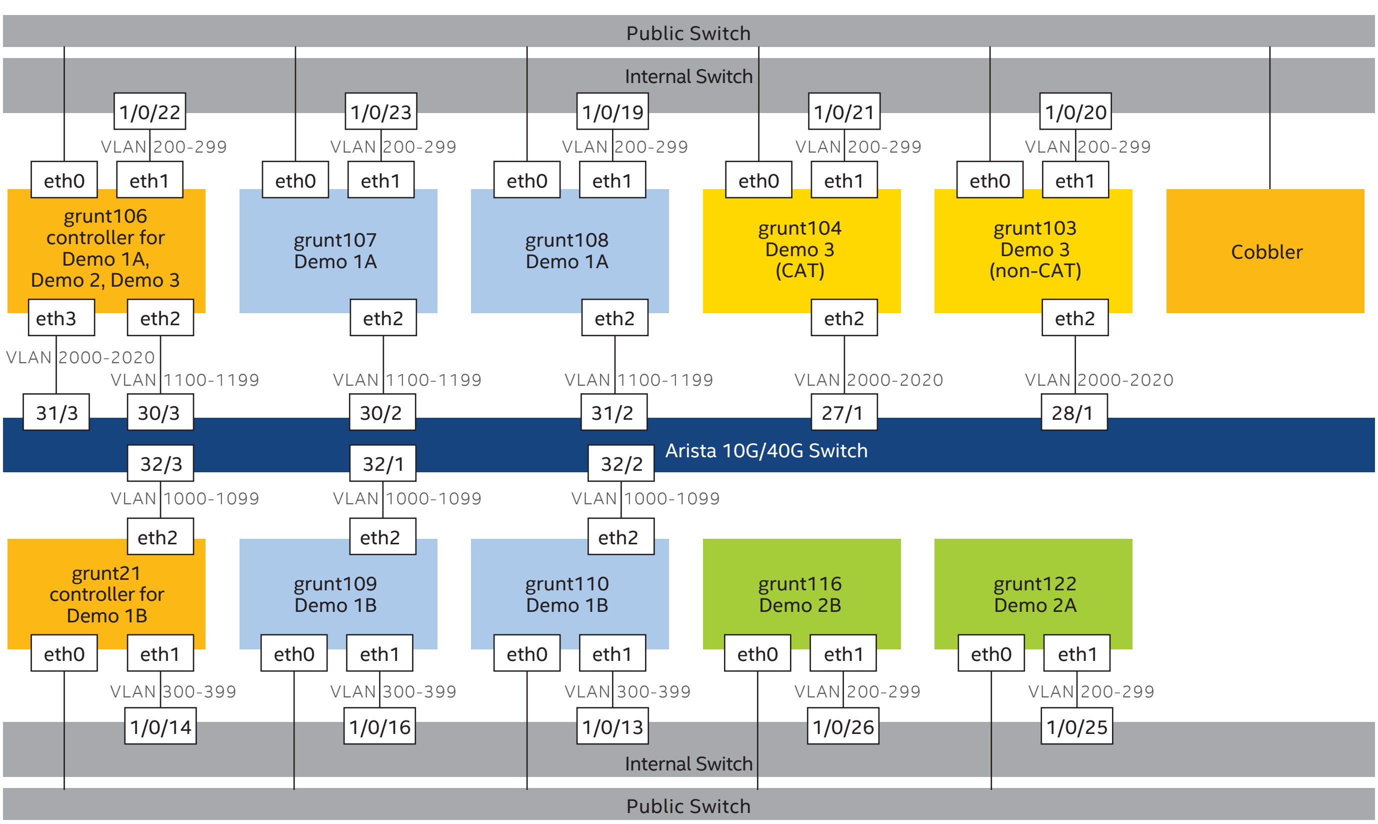


Figure 1. Physical topology for demonstrations.

Table 2. Hardware bill of materials.

SERVER	USAGE	PROCESSOR	MEMORY	DISK	PLATFORM	BIOS	NETWORK INTERFACE CARD AND INTEL® QUICKASSIST TECHNOLOGY
Cobbler	Cobbler network installation	Intel® Xeon® processor E5-2680 v2, 2.8 GHz, 10 cores	128 GB	1 TB Seagate* SATA	Intel® Server Board S2600GZ	SE5C600.86B .02.03.0003	Intel® Ethernet Controller I350 (2× 1 Gbps) Intel® 82599 10 Gigabit Ethernet Controller (2× 10 Gbps)
grunt106	Controller for Demo 1A, Demo 2, Demo 3	Intel Xeon processor E5-2680 v2, 2.8 GHz, 10 cores	64 GB	1 TB Seagate SATA	Intel Server Board S2600GZ	SE5C600.86B .02.03.0003	Intel Ethernet Controller I350 (4× 1 Gbps) via LAN-on-motherboard (LOM) (2 ports were used) Intel® Ethernet Server Adapter X520-DA2 (2× 10 Gbps)
grunt107	Demo 1A	Intel® Xeon® processor E5-2699 v3, 2.3 GHz, 18 cores	128 GB	1 TB Seagate SATA	Intel® Server Board S2600WT2	SE5C610.86B.01.01 .0008.021120151325	Intel Ethernet Controller I350 (2× 1 Gbps) via LOM Intel Ethernet Server Adapter X520-DA2 (2× 10 Gbps)
grunt108	Demo 1A	Intel Xeon processor E5-2699 v3, 2.3 GHz, 18 cores	128 GB	1 TB Seagate SATA	Intel Server Board S2600WT2	SE5C610.86B.01.01 .0008.021120151325	Intel Ethernet Controller I350 (2× 1 Gbps) via LOM Intel Ethernet Server Adapter X520-DA2 (2× 10 Gbps)
grunt21	Controller, for Demo 1B	Intel Xeon processor E5-2680 v2, 2.8 GHz, 10 cores	128 GB	1 TB Seagate SATA	Intel Server Board S2600GZ	SE5C600.86B .02.03.0003	Intel Ethernet Controller I350 (4× 1 Gbps) via LOM (2 ports were used) Intel Ethernet Server Adapter X520-DA2 (2× 10 Gbps)
grunt109	Demo 1B	Intel Xeon processor E5-2699 v3, 2.3 GHz, 18 cores	128 GB	1 TB Seagate SATA	Intel Server Board S2600WT2	SE5C610.86B.01.01 .0008.021120151325	Intel Ethernet Controller I350 (2× 1 Gbps) via LOM Intel Ethernet Server Adapter X520-DA2 (2× 10 Gbps)
grunt110	Demo 1B	Intel Xeon processor E5-2699 v3, 2.3 GHz, 18 cores	128 GB	1 TB Seagate SATA	Intel Server Board S2600WT2	SE5C610.86B.01.01 .0008.021120151325	Intel Ethernet Controller I350 (2× 1 Gbps) Intel Ethernet Server Adapter X520-DA2 (2× 10 Gbps)
grunt116	Demo 2B	Intel® Xeon® processor E5-2699 v4, 2.2 GHz, 22 cores	128 GB	500 GB HP* SATA	HP ProLiant DL380 Gen9*	P89 v2.00 (12/27/2015)	Broadcom NetXtreme BCM5719* controller (4× 1 Gbps) via LOM 2× Intel® QuickAssist Adapter 8950
grunt122	Demo 2A	Intel Xeon processor E5-2699 v4, 2.2 GHz, 22 cores	128 GB	500 GB HP SATA	HP ProLiant DL380 Gen9	P89 v2.00 (12/27/2015)	Broadcom NetXtreme BCM5719 controller (4× 1 Gbps) via LOM
grunt103	Demo 3 (Non-CAT)	Intel Xeon processor E5-2699 v4, 2.2 GHz, 22 cores	64 GB	2× 600 GB Intel® SSD	HP ProLiant DL380 Gen9	P89 v2.00 (12/27/2015)	Broadcom NetXtreme BCM5719 controller (4× 1 Gbps) via LOM Intel® Ethernet Converged Network Adapter XL710-QDA2 (2× 40 Gbps)
grunt104	Demo 3 (CAT)	Intel Xeon processor E5-2699 v4, 2.2 GHz, 22 cores	64 GB	2× 600 GB Intel® SSD	Quanta D51B-1U*	S2B_3B04	Intel Ethernet Controller I350 (2× 1 Gbps) via LOM Intel Ethernet Converged Network Adapter XL710-QDA2 (2× 40 Gbps)

Table 3 presents the software components used for demonstrations.

Table 3. Software versions.

FUNCTION	SOFTWARE COMPONENT
BIOS	Refer to Table 2.
Network installer	Cobbler 2.6.3
Host operating system	Fedora 21* Server, 3.19.7-200.fc21.x86_64
Guest operating system	Fedora 20 Server, 3.12.9-301.fc20.x86_64
MANO	RIFT.ware* 4.2.1
VNF orchestration	OpenStack* Kilo 2015.1.1
Packet processing acceleration	Data Plane Development Kit 2.2
Switching	Open vSwitch* 2.3.1-git4750c96
Traffic sink VNF	RIFT.ware Trafsink* 4.2.1
Traffic generator VNF	RIFT.ware Trafgen* 4.2.1
Converged access gateway VNF	RIFT.ware Converged Access Gateway* 4.2.1
On-premises data gateway VNF	RIFT.ware Premises Gateway* 4.2.1
Load balancer VNF	RIFT.ware Scriptable Load Balancer* 4.2.1

2.1 Demo 1: Impact of EPA-Based Optimizations and OVS-DPDK on Performance

The network service (NS) of the Demo 1A uses a regular Open vSwitch*. In the Demo 1B, NS will use DPDK-accelerated Open vSwitch (OVS-DPDK) with EPA features enabled, including non-uniform memory access (NUMA) architecture awareness, CPU pinning, huge pages, etc.

Each NS has 2 VNFs—RIFT.ware Trafgen* and RIFT.ware Trafsink*. Each VNF is a multiple-VM VNF consisting of four VMs. Each VM has 8 GB of memory and 2 virtual CPUs (vCPUs). The setup requirements for the hosts include:

- 1 host for OpenStack controller node with Open vSwitch
- 2 hosts for Demo 1A VNFs (8+ vCPUs, 32+ GB memory) with Open vSwitch
- 2 hosts for Demo 1B VNFs (8+ vCPUs, 32+ GB memory) with OVS-DPDK

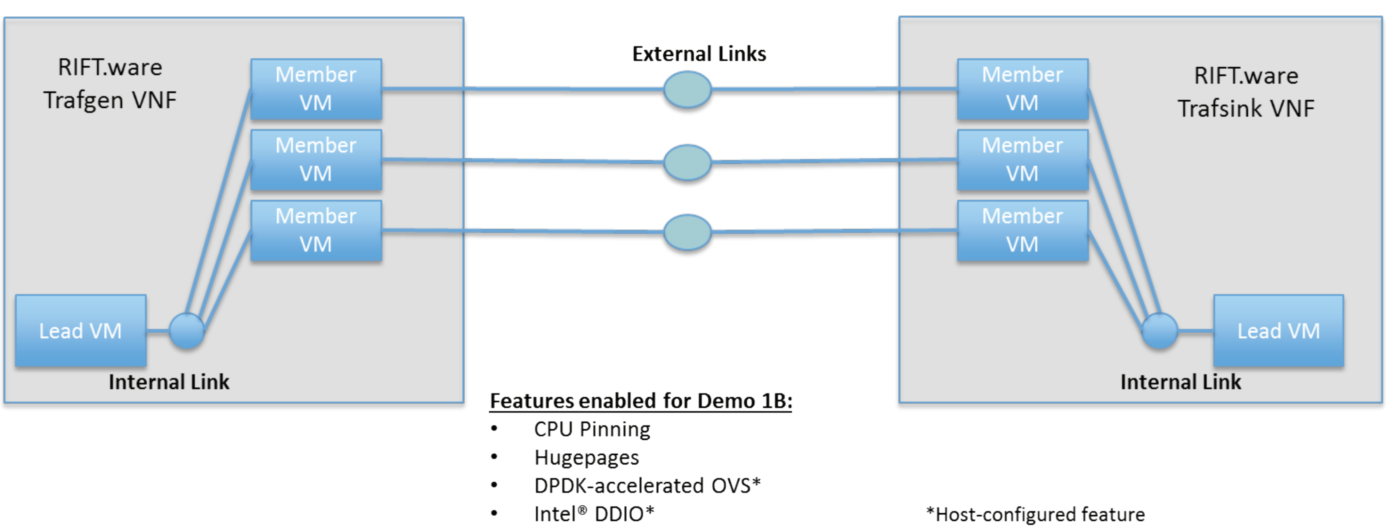


Figure 2. Logical connectivity for Demo 1.

2.2 Demo 2: Impact of Intel® QuickAssist Technology on Performance

The Demo 2A and 2B showcase the impact of Intel QuickAssist Technology. Demo 2A is run on the host that does not use this technology, while Demo 2B is run on the host with 2× Intel® QuickAssist Adapter 8950.

Each NS has two VNFs: RIFT.ware Converged Access Gateway* (CAG) and RIFT.ware Premises Gateway* (PGW). Only IKE traffic, available by initiating IPSec tunnel transactions, flows between the VNFs. The demo uses 4096-bit Diffie-Hellman keys. Each VNF is a single-VM VNF, and each VM has 16 GB of memory and 6 vCPUs. The external virtual link is VirtIO. The setup requirements for the hosts include:

- 1 host for OpenStack controller node
- 1 host for Demo 2A VNFs (12+ vCPUs, 32+ GB memory)
- 1 host for Demo 2B VNFs (12+ vCPUs, 32+ GB memory) with 2× Intel QuickAssist Adapter 8950

2.3 Demo 3: Impact of Cache Allocation Technology on Performance

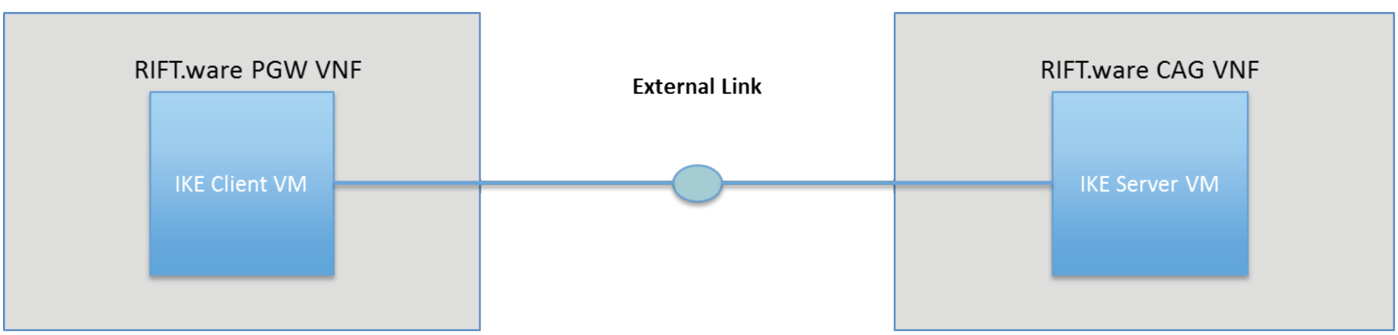
This demo uses one NS with three VNFs: RIFT.ware Trafgen*, RIFT.ware Scriptable Load Balancer* (RIFT.ware SLB*), and RIFT.ware Trafsink*. A noisy neighbor program is started on the host where the RIFT.ware SLB runs. In this demonstration, CAT is enabled and disabled on that host.

The VNFs are single-VM VNFs with 6 vCPUs and 32 GB of memory for each. There is approximately 40 Gbps of traffic between VNFs which is enough amount of data to examine cache usage and obtain key performance indicators.

The demo uses Intel® Ethernet Converged Network Adapter XL710-QDA2 with SR-IOV. Each VM has SR-IOV interfaces.

The setup requirements for the hosts are as follows:

- 1 host for OpenStack controller node
- 1 host for RIFT.ware Trafgen and RIFT.ware Trafsink VNFs (using 12 vCPUs and 64 GB of memory)
- 1 host for RIFT.ware SLB VNF (6 vCPUs and 32 GB memory) with CAT-capable CPU

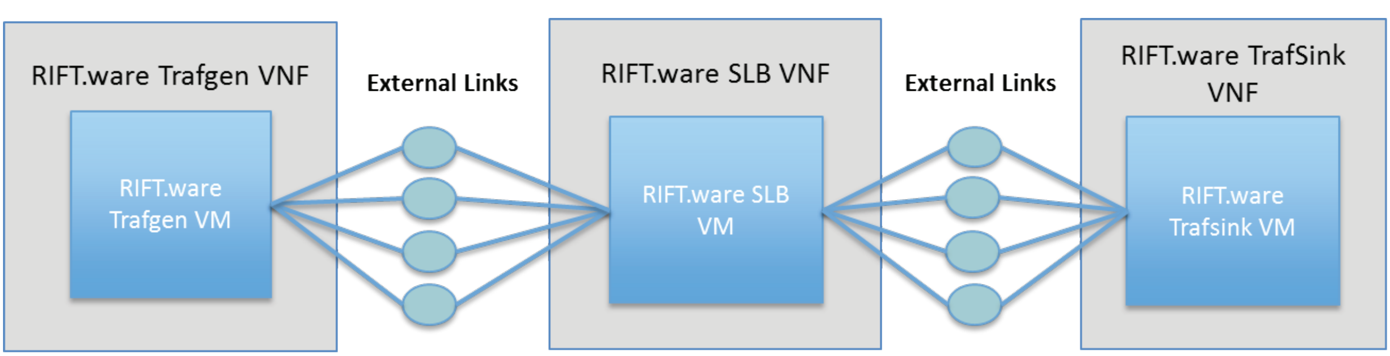


Features enabled for Demo 2B:

- Intel® QuickAssist Technology
- Intel® DDIO*

* Host-configured feature

Figure 3. Logical connectivity for Demo 2.



Features enabled for Demo 3:

- CAT
- Intel® DDIO*

* Host-configured feature

Figure 4. Logical Connectivity for Demo 3.

3.0 Installation Guide

3.1 BIOS Configuration

Prior to starting the installation process, configure the following BIOS settings on all of the 'grunt' servers that will be used as OpenStack nodes. The following configuration steps correspond to the BIOS of the Intel® Server Board S2600WT2.

Note: Depending on the manufacturer and the model of the server's mainboard, you may experience different BIOS with specific user interface that may require taking different steps to achieve similar configuration.

- **MAIN→QUIET BOOT**: `<Disabled>`
- **ADVANCED→POWER & PERFORMANCE→CPU POWER AND PERFORMANCE POLICY**: `<Performance>`
- **ADVANCED→PCI CONFIGURATION→NIC CONFIGURATION**: Disable preboot execution environment (PXE) on NIC ports except for the first one.
- **ADVANCED→PROCESSOR CONFIGURATION→INTEL (R) HYPER-THREADING TECH**: `<Disabled>`

ADVANCED→PROCESSOR CONFIGURATION→INTEL (R) VIRTUALIZATION TECHNOLOGY: `<Enabled>`

- **ADVANCED→INTEGRATED IO CONFIGURATION→INTEL (R) VT FOR DIRECTED I/O**: `<Enabled>`
- **ADVANCED→INTEGRATED IO CONFIGURATION→ COHERENCY SUPPORT**: `<Enabled>`
- **SERVER MANAGEMENT→Resume on AC Power Loss**: `<Last State>`
- **BOOT MANAGER**: Change boot order to: Network Interface, SATA, EFI Shell
- **ADVANCED BOOT OPTIONS→System Boot Timeout**: `[5]`

3.2 Install Operating System and the OpenStack through the Cobbler

Note: Beside the following steps, you can also refer to Use Cobbler Image to Deploy OpenStack section in the OpenStack Installation Guide by RIFT.io.

3.2.1 Install OS and Configure the Network

Execute the following steps on Cobbler server.

1. Install Fedora* 21 Server operating system (OS) on Cobbler server with virtualization packages such as `qemu-kvm`, `libvirt-client`, and so on.
2. Configure the network for Internet access. Then connect Cobbler server and all the 'grunt' servers to the public switch on the same local area network (LAN). The 'grunt' servers will use PXE network boot process to install OS and OpenStack through the Cobbler server.

3.2.2 Create and Start Cobbler VM

Execute the following steps on the Cobbler server.

1. Create a directory and download the Cobbler files.

```
# mkdir –p /kvm/cobbler
# cd /kvm/cobbler
# wget http://repo.riftio.com/releases/open.riftio.com/4.2.1/cobbler.xml -O /kvm/cobbler/cobbler.xml
# wget http://repo.riftio.com/releases/open.riftio.com/4.2.1/cobbler.qcow2 -O /kvm/cobbler/cobbler.qcow2
```

2. Configure the network for Cobbler VM.

Create a bridge for the installation of Cobbler VM. Add the corresponding physical port into the bridge.

```
# brctl addbr br-cobbler
#bridge for cobbler installation
# brctl addif br-cobbler enp1s0f1  #port connected to controller and compute nodes
```

Edit the `/kvm/cobbler/cobbler.xml` file according to the network configuration above; modify the interface `type='bridge'` section.

```
        <interface type='bridge'>
    <mac address='52:54:00:13:d4:22'/>
    <source bridge='br-cobbler'/>
              <model type='virtio'/>
              <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
        </interface>
```

3. Create, start, and log in to the Cobbler VM.

```
# virsh define /kvm/cobbler/cobbler.xml
# virsh start cobbler
# virsh console cobbler
```

Note: If you encounter "`unsupported configuration: Unable to find security driver for label selinux`" when executing `virsh start cobbler` command, disable the security-enhanced Linux (SELinux), and edit the `/etc/libvirt/qemu.conf` file to remove the `<seclabel type='dynamic' model='selinux'>` entry.

3.2.3 Configure Cobbler VM Networks

Execute the following steps on the Cobbler VM.

Note: Log in to the Cobbler VM with `root/riftIO` as user/password.

1. Modify IP addresses for eth0 by editing the `/etc/sysconfig/network-scripts/ifcfg-eth0` file.

```
DEVICE="eth0"
BOOTPROTO=static
IPADDR=192.168.12.111
NETWORK=192.168.12.0
NETMASK=255.255.255.0
GATEWAY = 192.168.12.1
ONBOOT="yes"
TYPE="Ethernet"
HWADDR="52:54:00:13:d4:22"
```

2. Remove the static routers related to `10.95.0.0/16` in the `/etc/rc.d/rc.local start-up script file.`

3. Restart the `network` service.

```
# service network restart
```

3.2.4 Configure Cobbler Software

Execute the following steps on the Cobbler VM.

1. Modify the Cobbler's settings in the `/etc/cobbler/settings` file.

```
server: 192.168.12.111 //ip address of
cobbler VM
next_server: 192.168.12.111 //ip address
of dhcp server, same as cobbler VM in
our case
manage_dhcp: 1 //cobbler will manage dhcp
manage_tftp: 1 //cobbler will manage tftp
manage_tftpd: 1
```

2. Modify the `/etc/cobbler/dhcp.template` file.

```
subnet 192.168.12.0 netmask 255.255.255.0 {
//dhcp subnet for cobbler
    option routers
192.168.12.1; //gateway router
    option subnet-mask
255.255.255.0; //subnet-mask
    range dynamic-bootp
192.168.12.200 192.168.12.253; //ip range
for dynamic allocation
    default-lease-time 21600;
    max-lease-time 43200;
    next-server $next_
server; //never touch it
```

Note: To support PXE network boot process, Cobbler will enable the Dynamic Host Configuration Protocol (DHCP) server to allocate the IP addresses for Cobbler clients (the controller and compute nodes). Cobbler will use the `dhcp.template` file to update the `/etc/dhcp/dhcpd.conf` file.

3. Make a backup of the existing Cobbler settings, download the `cobbler2_06042016.tgz` file, and update with new settings.

```
# cd /var/lib/cobbler
# git init .
# git add .
# git commit -m "init"
# git checkout -b intel_idf
# git checkout -b intel_kpi2016
# wget http://repo.riftio.com/releases/
open.riftio.com/4.2.1/cobbler2_06042016.tgz
# tar xzvf cobbler2_06042016.tgz
```

4. Pull the updates to the Cobbler using the `reposync` command on each of the following repositories:

- `kilo`
- `openstack-kilo`
- `rift-misc`
- `riftware-test`

Log in to the Cobbler graphical user interface (GUI) at https://192.168.12.111/cobbler_web with `cobbler/cobbler` as user/password. Go to **CONFIGURATION→REPOS**, and select one of the repository. Click **BATCH ACTIONS**, select **REPOSYNC**, and click G**O**. Go to **COBBLER→EVENTS**, and verify the status. Once completed, repeat these steps for the next repository.

5. Remove extra and unneeded packages (`ejabberd, qemu-kvm-tools, qemu, libvirt, libvirt-daemon-qemu`) from the `/var/lib/cobbler/snippests/rift-grunt-fc21-packages` configuration file.

6. Edit the `/var/lib/cobbler/snippests/rift-post-kilo` configuration file to change all `wheel.riftio.com` URLs to `wheel.eng.riftio.com`. Change all `8881` ports to `80`, if the firewall blocks the TCP 8881 port.

7. Modify the snippet profiles.

   The Cobbler VM is prepackaged with two snippet profiles, `intel-kpi-demo.cfg and intel-kpi-demo-dpdk.cfg`. These two profiles can be used to install the OS and OpenStack Kilo.

   Edit node details in these two snippet profiles, with the controller IP address, virtual LAN (VLAN) information, private IP address, and floating IP subnet, through a command line or GUI.

- `/var/lib/cobbler/snippets/intel-kpi-demo.cfg`:

```
case $name in

grunt106)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
    ;;

grunt107)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
    ;;

grunt108)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
```

```
    ;;

grunt103)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
    ;;

grunt104)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
    ;;

grunt116)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
    ;;

grunt122)
    CONTROLLER=192.168.12.106
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=200:299
    PRIVATE_IP=66.0.0.0
    FLOATING_IP=192.168.12.0
    ;;

*)
    ;;
esac
```

- /var/lib/cobbler/snippets/intel-kpi-demo-dpdk.cfg:

```
case $name in

grunt21)
    CONTROLLER=192.168.12.121
    BRGIF=3
    OVSDPDK=N
    TRUSTED=N
    QAT=N
    HUGEPAGE=0
    VLAN=300:399
    IP_KEY=121
    ;;

grunt109)
    CONTROLLER=192.168.12.121
    BRGIF=3
    OVSDPDK=Y
    TRUSTED=N
    QAT=N
    HUGEPAGE=60000
    VLAN=300:399
    IP_KEY=121
    KERNEL_OPT="isolcpus=2-17,20-35"
    VCPU_PIN_SET="5-17,22-35"
    ;;

grunt110)
    CONTROLLER=192.168.12.121
    BRGIF=3
    OVSDPDK=Y
    TRUSTED=N
    QAT=N
    HUGEPAGE=60000
    VLAN=300:399
    IP_KEY=121
    KERNEL_OPT="isolcpus=2-17,20-35"
    VCPU_PIN_SET="5-17,22-35"
    ;;

*)
    ;;
esac
```

Note: For best performance, make sure that all OVS-DPDK virtual CPUs (vCPUs), VM vCPUs and eth2 are on the same NUMA node, and in the `intel-kpi-demo-dpdk.cfg` snippet profile for Demo 1B, vCPUs 0-4 are used by OVS-DPDK. Thus, do not allocate them to VMs in the `KERNEL_OPT` and `VCPU_PIN_SET` values.

8. Start and enable the following services.

```
# systemctl start httpd.service
# systemctl start xinetd.service
# systemctl start cobblerd.service
# systemctl start dhcpd.service

# systemctl enable httpd.service
# systemctl enable xinetd.service
# systemctl enable cobblerd.service
```

9. Starting with the controller node, add the system nodes to Cobbler. Other nodes that use the same profile as the controller node are set up as compute nodes and are attached to the named controller node.

   Add the system node with a correct IP address, media access control (MAC) address, and profile, through the command line or Cobbler GUI.

```
# cobbler system add
--name=grunt106 --hostname=grunt106
--mac=00:1e:67:feb2:09ef:a1b4
--interface=eth0 --ip-
address=192.168.12.106 --profile=intel-kpi-
demo
```

```
# cobbler system add
--name=grunt107 --hostname=grunt107
--mac=00:1e:67:d1:a1:0d --interface=eth0
--ip-address=192.168.12.107 --profile=intel-
kpi-demo
# cobbler system add
--name=grunt108 --hostname=grunt108
--mac=00:1e:67:fe:f2:bb --interface=eth0
--ip-address=192.168.12.108 --profile=
intel-kpi-demo
# cobbler system add
--name=grunt103 --hostname=grunt103
--mac=1c:98:ec:2b:4b:20 --interface=eth0
--ip-address=192.168.12.103 --profile=
intel-kpi-demo
# cobbler system add
--name=grunt104 --hostname=grunt104
--mac=2c:60:0c:66:fe:aa --interface=eth0
--ip-address=192.168.12.104 --profile=
intel-kpi-demo
# cobbler system add
--name=grunt116 --hostname=grunt116
--mac=1c:98:ec:2b:db:98 --interface=eth0
--ip-address=192.168.12.116 --profile=
intel-kpi-demo
# cobbler system add
--name=grunt122 --hostname=grunt122
--mac=1c:98:ec:2b:3c:18 --interface=eth0
--ip-address=192.168.12.122 --profile=
intel-kpi-demo
# cobbler system add --name=grunt21
--hostname=grunt21 --mac=00:1e:67:b2:5e:73
--interface=eth0 --ip-
address=192.168.12.121 --profile=intel-kpi-
demo-dpdk
# cobbler system add
--name=grunt109 --hostname=grunt109
--mac=00:1e:67:cf:bf:27 --interface=eth0
--ip-address=192.168.12.109 --profile=intel-
kpi-demo-dpdk
# cobbler system add
--name=grunt110 --hostname=grunt110
--mac=00:1e:67:cf:b6:da --interface=eth0
--ip-address=192.168.12.110 --profile=intel-
kpi-demo-dpdk
```

10. Update the Cobbler.

    Anytime you change the Cobbler snippet file or add a Cobbler system, make sure to update the Cobbler; otherwise, your configuration will not work.

```
# cobbler check
# cobbler sync
```

3.2.5 Install OS and OpenStack on All Hosts

1. Configure BIOS as in section 3.1; make sure to enable PXE for the first Ethernet port.
2. Reboot the controller nodes (grunt106, grunt21) to install the OS and OpenStack.
3. Reboot the compute nodes to install the OS and OpenStack.

3.3 Install RIFT.ware from RPM Package Manager

Note: Beside the following steps, you can also refer to Install RIFT.ware from an RPM section in the RIFT.ware Installation Guide.

3.3.1 Download RIFT.ware Binary Software.

1. Using a browser, go to http://repo.riftio.com/releases/open.riftio.com/4.2.1/ and download the `rift-ui-latest.qcow2` runtime VM image that is suitable for hosting RIFT.ware.
2. Open https://open.riftio.com/download/ and download the `id_grunt` secure shell (SSH) key file for accessing the rift-ui VM as root user.

3.3.2 Prepare the OpenStack for RIFT.ware

1. Using a browser, navigate to dashboards of all controller nodes (in this example, http://192.168.12.106 and http://192.168.12.121).
2. Create the `demo` project and the related user.
    - Log in with `admin/mypasswd` as user/password.
    - Go to the **IDENTITY** → **PROJECTS** tab, and click **CREATE PROJECT**. Then, enter demo for **NAME**, and click **CREATE PROJECT**.
    - Go to the **IDENTITY** → **USERS** tab, and click **CREATE USER**. Then, enter demo for **USER NAME** and set the respective password. Select demo for **PRIMARY PROJECT**. Select admin for **ROLE**, and click **CREATE USER**.
    - Go to the **IDENTITY** → **PROJECTS** tab, and find the demo project. Click **MANAGE MEMBERS**, add admin to **PROJECT MEMBERS**, and click **SAVE**.
3. Add security rules.
    - Log in as `demo` user.
    - Go to **PROJECT** → **COMPUTE** → **ACCESS & SECURITY** tab, find the default rule, and click **MANAGE RULES**.
    - Click **ADD RULE**. Select: `ALL TCP` for **RULE**, `Ingress` for **DIRECTION**, and then click **ADD**.
    - Click **ADD RULE** again, select: `ALL TCP` for **RULE**, `Egress` for **DIRECTION**, and then click **ADD**.
4. Create virtual router for the demo project.
    - Log in as `demo` user.
    - Go to **PROJECT** → **NETWORK** → **ROUTERS** tab, and click **CREATE ROUTER**.
    - Enter `router_demo` for **ROUTER NAME**, select public for **EXTERNAL NETWORK**, and click **CREATE ROUTER**.
    - Click just created `router_demo`. Go to **INTERFACES** tab, and click ADD INTERFACE.
    - Select `private` for **SUBNET**, and click **ADD INTERFACE**.
    - Go to **PROJECT** → **NETWORK** → **NETWORK TOPOLOGY** tab, and verify if `router_demo` connects to the public and private networks.

3.3.3 Upload VM Image to OpenStack

1. Log in as `demo` user.
2. Go to **PROJECT** → **COMPUTE** → **IMAGES** tab, and click **CREATE IMAGE**.
3. Enter `rift-ui-img` for **NAME**, select **IMAGE FILE** for **IMAGE SOURCE**, click **CHOOSE FILE** to pick the `rift-ui-latest.qcow2` downloaded in section 3.3.1, check **PUBLIC**, and then click **CREATE IMAGE**.

3.3.4 Initiate VM

1. Initiate RIFT.ware Launchpad VM.
   - Log in as `demo` user.
   - Go to **PROJECT** → **COMPUTE** → **INSTANCES** tab, and click **LAUNCH INSTANCE**.
   - Enter `rift-ui` for **INSTANCE NAME**, select `m1.medium` for **FLAVOR**, and Boot from image for **INSTANCE BOOT SOURCE**. Select `rift-ui-img` (1.0 GB) uploaded in the section 3.3.3 for `IMAGE NAME`.
   - On the **NETWORKING** tab, add `private` for selected networks.
   - Click **LAUNCH**.
2. Associate the floating IP for the VM. Use ping to verify a connection.
   - Go to **PROJECT** → **COMPUTE** → **INSTANCES** tab, find just created `rift-ui` instance, and select **ASSOCIATE FLOATING IP**.
   - Click + to allocate the floating IP, select `public` for **POOL**, and click **ALLOCATE IP**.
   - Select the allocated IP (here, it is 192.168.12.50), and click **ASSOCIATE** button.
   - From Cobbler VM, ping the 192.168.12.50 IP address to verify the connection.
3. Access the VM with the SSH key file.
   - Upload the `id_grunt` file (downloaded at section 3.3) to the Cobbler VM.
   - Add execution permission for this file, and then use it to connect with SSH to the VM.

```
# mv ~/Downloads/id_grunt ~/.ssh/
# chmod 600 ~/.ssh/id_grunt
# ssh –i ~/.ssh/id_grunt root@192.168.12.50
```

3.3.5 Install and Start RIFT.ware

1. Configure the network according to your network environment, such as the name server, proxy, yum repo file, to make sure the VM can access the Internet.
2. Clean up the older RIFT.ware packages.

```
# yum remove 'rift*'
```

3. Install `sudo`.

```
# yum install sudo
```

4. Configure the yum repository information.

```
# rpm –Uvh http://repo.riftio.com/
releases/riftware-release-latest.rpm
```

5. Download the RIFT.ware 4.2.1 repository file.

```
# curl -o /etc/yum.repos.d/riftware-4.2-
intel_4.2.1.repo http://repo.riftio.com/
releases/repos/intel_4.2.1.repo
# yum makecache
```

6. Select the repository code by running the following series of commands.

```
# yum-config-manager --disable RIFT.ware
# yum-config-manager --disable RIFT.ware-
4.1
# yum-config-manager --disable RIFT.ware-
4.2
# yum-config-manager --disable RIFT.ware-
4.2-testing
# yum-config-manager --disable RIFT.ware-
4.2-nightly
# yum-config-manager --enable RIFT.ware-
4.2-intel_4.2.1
```

7. Optionally, install the `riftware-loganalyzer` package to view the syslog data through a browser interface.

```
# yum install riftware-loganalyzer
```

8. Install the `riftware-launchpad` package with all the dependencies.

```
# yum install riftware-launchpad
```

9. After the installation is completed, start the RIFT.ware Launchpad services.

```
# service rwlp start
```

4.0 Configuration

4.1 Physical Switches

1. Configure the physical internal switch.

   Execute the following commands to configure all the ports of the internal switch that connect VLAN 200-299 with eth1 ports on grunt106, grunt107, grunt108, grunt103, grunt104, grunt116, and grunt122.

```
configure terminal
vlan 200-299
exit
interface Ethernet 1/0/19-1/0/26
switchport mode trunk
switchport trunk allowed vlan 200-299
```

   Execute the following commands to configure all the ports of the internal switch that connect VLAN300–399 with eth1 ports on grunt21, grunt109, and grunt110.

```
configure terminal
vlan 300-399
exit
interface Ethernet 1/0/13,1/0/14,1/0/16
switchport mode trunk
switchport trunk allowed vlan 300-399
```

2. Configure the Arista* 10G/40G switch.

   Execute the following commands to configure the ports that connect Demo 1A hosts with VLAN 1100–1199.

```
configure terminal
vlan 1100-1199
exit
interface Ethernet 30/2, 30/3, 31/2
switchport mode trunk
switchport trunk allowed vlan 1100-1199
```

   Execute the following commands to configure the ports that connect Demo 1B hosts with VLAN 1000–1099.

```
configure terminal
vlan 1000-1099
exit
interface Ethernet 32/1, 32/2, 32/3
switchport mode trunk
switchport trunk allowed vlan 1000-1099
```

   Execute the following commands to configure the ports that connect Demo 3 hosts with VLAN 2000–2020.

```
configure terminal
vlan 2000-2020
exit
interface Ethernet 27/1, 28/1
speed forced 40gfull
switchport mode trunk
switchport trunk allowed vlan 2000-2020
interface Ethernet 31/1
switchport mode trunk
switchport trunk allowed vlan 2000-2020
```

4.2 Demo 1A

4.2.1 Controller Node (grunt106)

1. Update domain name service (DNS) for the private network.

```
# . keystonerc_admin
```

```
# neutron subnet-update --dns-nameserver
8.8.4.4 private-subnet
```

2. Tune MariaDB*.

   Edit the /etc/systemd/system/mariadb.service.d/limits.conf file to include the following content, or create it, if the file does not exist.

```
[Service]
LimitNOFILE=40000
```

   Edit the /etc/security/limits.d/95-rift.conf file to include the following content, or create it, if the file does not exist.

```
* soft nofile 1024000
* hard nofile 1024000
* soft nproc 10240
* hard nproc 10240
```

3. Update the /etc/nova/nova.conf file with the following content.

```
scheduler_default_filters=AggregateInstanc
eExtraSpecsFilter,AvailabilityZoneFilter,R
amFilter,ComputeFilter,AvailabilityZoneFil
ter,ComputeCapabilitiesFilter,ImageProper
tiesFilter,CoreFilter,PciPassthroughFilter
,NUMATopologyFilter,ServerGroupAntiAffinit
yFilter,ServerGroupAffinityFilter
```

4. Create an Open vSwitch bridge on the controller node.

```
# ifup eth2
# ip link set eth2 promisc on
# ovs-vsctl add-br br-demo1pnet2
# ovs-vsctl add-port br-demo1pnet2 eth2
```

5. Update the /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini file to add the demo1pnet2 provider network.

```
:::
bridge_mappings = physnet1:br-
eth1,demo1pnet2:br-demo1pnet2
:::
```

6. Update the /etc/neutron/plugins/ml2/ml2_conf.ini file to update the VLAN range for the demo1pnet2 provider network.

```
:::
network_vlan_ranges =physnet1:200:299,demo
1pnet2:1100:1199
:::
```

7. Restart OpenStack Networking* services.

```
# systemctl restart neutron-server.
service
# systemctl restart neutron-openvswitch-
agent.service
# systemctl restart neutron-ovs-cleanup.
service
# systemctl restart neutron-l3-agent.
service
# systemctl restart neutron-dhcp-agent.
service
```

4.2.2 Compute Nodes (grunt107, grunt108)

1. Create an Open vSwitch bridge on the Demo 1A compute nodes.

```
# ifup eth2
# ip link set eth2 promisc on
# ovs-vsctl add-br br-demo1pnet2
# ovs-vsctl add-port br-demo1pnet2 eth2
```

2. Update the /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini file.

```
:::
bridge_mappings=physnet1:br-
eth1,demo1pnet2:br-demo1pnet2
:::
```

3. Restart the neutron-openvswitch-agent.service service.

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

4.3 Demo 1B

4.3.1 Controller Node (grunt21)

1. Update the DNS for the private network.

```
# . keystonerc_admin
# neutron subnet-update —dns-nameserver 
8.8.4.4 private-subnet
```

2. Tune MariaDB*.

Edit the `/etc/systemd/system/mariadb.service.d/limits.conf` file to include the following content, or create it, if the file does not exist.

```
[Service]
LimitNOFILE=40000
```

Edit the `/etc/security/limits.d/95-rift.conf` file to include the following content, or create it, if the file does not exist.

```
* soft nofile 1024000
* hard nofile 1024000
* soft nproc 10240
* hard nproc 10240
```

3. Add an Open vSwitch bridge for the 10 GbE network for the DHCP path.

```
# ovs-vsctl add-br br-eth2
# ovs-vsctl add-port br-eth2 eth2
```

4. Update the `ml2_conf.ini` file.

```
:::
[ml2_type_vlan]
network_vlan_ranges = physnet1:300:399, 
physdpdk:1000:1099
:::
[ovs]
bridge_mappings = physnet1:br-eth1, 
physdpdk:br-eth2
```

5. Restart all the OpenStack Networking services.

```
# su – stack
# screen –r stack
```

Use **CTRL-A-“** to list the services, and select all OpenStack Networking services. Use **CTRL-C** to stop each service, and then use the **UP** arrow key to get the command to start the service again.

4.3.2 Compute Nodes (grunt109, grunt110)

1. Add an Open vSwitch bridge for the 10 GbE network.

```
# ovs-vsctl add-br br-eth2
# ovs-vsctl add-port br-eth2 eth2
```

2. Update the ml2_conf.ini file.

```
:::
[ml2_type_vlan]
network_vlan_ranges = physnet1:300:399, 
physdpdk:1000:1099
:::
[ovs]
bridge_mappings = physnet1:br-eth1, 
physdpdk:br-eth2
```

3. Restart all the OpenStack Networking services.

```
# su – stack
# screen –r stack
```

Use **CTRL-A-“** to list the services, and select all OpenStack Networking services. Use **CTRL-C** to stop each service, and then use the **UP** arrow key to get the command to start the service again.

4.4 Demo 2A

4.4.1 Controller Node (grunt106)

No additional configuration is needed.

4.4.2 Computer Node (grunt122)

1. Enable huge pages by adding the following line to the `/etc/sysctl.conf` file.

```
vm.nr_hugepages=32780
```

2. Reboot the compute node.

4.5 Demo 2B

4.5.1 Controller Node (grunt106)

Based on the configuration in section 4.2.1,

1. Update the `/etc/nova/nova.conf` file.

```
:::
pci_alias={“vendor_id”:”8086”, “product_
id”:”0443”, “name”:”PCI_QAT”}
:::
```

2. Restart the following OpenStack Compute* services.

```
# openstack-service restart nova-api
# openstack-service restart nova-compute
# openstack-service restart nova-
scheduler
# openstack-service restart nova-
conductor
```

4.5.2 Computer Node (grunt116)

1. Install the `qat_host` service.

```
# yum install qat_host –y
```

2. Enable huge pages by adding the following line to the `/etc/sysctl.conf` file.

```
vm.nr_hugepages=32780
```

3. Update the `/etc/nova/nova.conf` file.

```
:::
pci_passthrough_whitelist = [{“vendor_
id”:”8086”, “product_id”:”0443”}]
:::
```

4. Reboot the compute node.

4.6 Demo 3

4.6.1 Controller Node (grunt106)

Based on the configuration in section 4.5.1,

1. Add an Open vSwitch bridge.

```
# ovs-vsctl add-br br-demo3-eth2
# ovs-vsctl add-port br-demo3-eth2 eth3
```

2. Update the /etc/neutron/openvswitch/ovs_neutron_plugin.ini file to add the demo3pnet2 provider network.

```
:::
bridge_mappings =physnet1:br-
eth1,demo1pnet2:br-
demo1pnet2,demo3pnet2:br-demo3-eth2
:::
```

3. Update the /etc/neutron/plugins/ml2/ml2_conf.ini file.

```
:::
mechanism_drivers=openvswitch,sriovnicsw
itch
:::
network_vlan_ranges= physnet1:200:299,demo
1pnet2:1100:1199,demo3pnet2:2000:2020
:::
```

4. Update the /usr/lib/systemd/system/neutron-server.service file.

```
:::
ExecStart=/usr/bin/neutron-server
--config-file /usr/share/neutron/neutron-
dist.conf --config-dir /usr/share/neutron/
server --config-file /etc/neutron/neutron.
conf --config-file /etc/neutron/plugin.
ini --config-file /etc/neutron/plugins/ml2/
ml2_conf.ini --config-dir /etc/neutron/
conf.d/neutron-server --log-file /var/log/
neutron/server.log
:::
```

5. Restart the following services.

```
# systemctl daemon-reload
# systemctl restart neutron-server.
service
# systemctl restart neutron-openvswitch-
agent.service
# systemctl restart neutron-ovs-cleanup.
service
# systemctl restart neutron-l3-agent.
service
# systemctl restart neutron-dhcp-agent.
service
```

6. Add the following commands to the rc.local script to ensure that the eth3 port is up for the DHCP path.

```
:::
ip link set eth3 up
sleep 40
:::
```

4.6.2 Compute Node (grunt 103, 104)

1. Add the i40evf driver to the blacklist.

```
# echo “blacklist i40evf” >> /etc/
modprobe.d/blacklist.conf
```

2. Update the rc.local script.

```
:::
ip link set eth4 up
echo 4 > /sys/class/net/eth4/device/sriov_
numvfs
sleep 40
:::
```

3. Update boot kernel option in the /etc/default/grub file.

```
:::
GRUB_CMDLINE_LINUX="iommu=pt intel_
iommu=on nomodeset crashkernel=192M
rd.lvm.lv=vg0/LogVol00 "
:::
```

4. Update the GNU GRand Unified Bootloader (GRUB).

```
# grub2-mkconfig –o /boot/grub2/grub.cfg
```

5. Create an Open vSwitch bridge.

```
# ovs-vsctl add-br br-demo3-eth2
# ovs-vsctl add-port br-demo3-eth2 eth4
```

6. Install SR-IOV network interface card (NIC) agent.

```
# yum install openstack-neutron-sriov-
nic-agent.noarch –y
```

7. Create the /etc/neutron/plugins/ml2/ml2_conf_sriov.ini file with the following content.

```
[ml2_sriov]
supported_pci_vendor_dev = 8086:154c
agent_required = True
[sriov_nic]
physical_device_mappings =
demo3pnet2:eth4
```

8. Update the /usr/lib/systemd/system/neutron-sriov-agent.service file to include the /etc/neutron/plugins/ml2/ml2_conf_sriov.ini file on the ExecStart list.

```
[Unit]
Description=OpenStack Neutron SR-IOV NIC
Agent
After=syslog.target network.target
[Service]
Type=simple
User=neutron
ExecStart=/usr/bin/neutron-sriov-nic-
agent --debug --config-file /usr/share/
neutron/neutron-dist.conf --config-file /
etc/neutron/neutron.conf --config-file /
etc/neutron/plugins/ml2/ml2_conf_sriov.
ini --log-file /var/log/neutron/sriov-nic-
agent.log
PrivateTmp=false
KillMode=process
[Install]
WantedBy=multi-user.target
```

9. Restart and enable the neutron-sriov-nic-agent.service.

```
# systemctl daemon-reload
# systemctl restart neutron-sriov-nic-agent
# systemctl enable neutron-sriov-nic-agent
```

10. Update the /etc/nova/nova.conf file.

```
:::
pci_passthrough_whitelist = {"devname":"eth4","physical_network":"demo3pnet2"}
:::
```

11. Update the ovs_neutron_plugin.ini file.

```
:::
bridge_mappings=physnet1:br-eth1,demo3pnet2:br-demo3-eth2
:::
```

12. Reboot the compute nodes.

```
# reboot
```

4.6.3 Additional Tools for the CAT Node (grunt104)

1. Install the pqos tool for CAT.

```
# mkdir /usr/src/cat
# cd /usr/src/cat
# wget http:// repo.riftio.com/releases/open.riftio.com/4.2.1/tools/cmt_cat_refcode.1.0.1.3-10.tgz
# tar xzvf cmt_cat_refcode.1.0.1.3-10.tgz
# cd pqos
# make all
```

2. Install the rwstream as the neighbor noise program.

```
# cd /usr/src
# wget http://repo.riftio.com/releases/open.riftio.com/4.2.1/tools/stream.tgz
# tar zxvf stream.tgz
# cd stream
# tar xvf stream-5.10.tar
# ./build.sh
# cp rwstream.service /etc/systemd/system/rwstream.service
# cp start_rwstream_server /opt/rift/rwstream/start_rwstream_server
# systemctl daemon-reload
# systemctl enable rwstream
```

4.7 Configure Host Aggregate

4.7.1 Example of Host Aggregate Configuration

1. Create the host aggregate.

```
# nova aggregate-create Trafgen
+----+------------+-------------------+-------+----------+
| Id | Name       | Availability Zone | Hosts | Metadata |
+----+------------+-------------------+-------+----------+
| 1  | Trafgen    | -                 |       |          |
+----+------------+-------------------+-------+----------+
```

2. Add host into the host aggregate.

```
# nova aggregate-add-host Trafgen grunt107
+----+------------+-------------------+-----------+----------+
| Id | Name       | Availability Zone | Hosts     | Metadata |
+----+------------+-------------------+-----------+----------+
| 1  | Trafgen    | -                 |'grunt107' |          |
+----+------------+-------------------+-----------+----------+
```

3. Set metadata for the host aggregate.

```
# nova aggregate-set-metadata Trafgen Trafgen=True
+----+------------+-------------------+-----------+--------------+
| Id | Name       | Availability Zone | Hosts     |   Metadata   |
+----+------------+-------------------+-----------+--------------+
| 1  | Trafgen    | -                 |'grunt107' |Trafgen=True  |
+----+------------+-------------------+-----------+--------------+
```

4.7.2 Configure Host Aggregate for grunt106

Follow the above steps to configure host aggregates for grunt106 as shown below.

- Trafgen

```
+----+---------+-------------------+-----------+-----------------+
| Id | Name    | Availability Zone | Hosts     | Metadata        |
+----+---------+-------------------+-----------+-----------------+
| 1  | Trafgen | -                 | 'grunt107' | 'Trafgen=True'  |
+----+---------+-------------------+-----------+-----------------+
```

- Trafsink

```
+----+----------+------------------+-----------+-----------------+
| Id | Name     | Availability Zone | Hosts    | Metadata        |
+----+----------+------------------+-----------+-----------------+
| 2  | Trafsink | -                | 'grunt108' | 'Trafsink=True' |
+----+----------+------------------+-----------+-----------------+
```

- CAT

```
+----+------+--------------------+-----------+-----------+
| Id  | Name | Availability Zone  | Hosts      | Metadata   |
+----+------+--------------------+-----------+-----------+
| 3   | CAT  | -                  | 'grunt104’ | 'CAT=True'  |
+----+------+--------------------+-----------+-----------+
```

- NonCAT

```
+----+--------+-------------------+-----------+---------------+
| Id | Name   | Availability Zone | Hosts      | Metadata        |
+----+--------+-------------------+-----------+---------------+
| 4  | NonCAT | -                 | 'grunt103’ | 'NonCAT=True'   |
+----+--------+-------------------+-----------+---------------+
```

- NonQAT

```
+----+--------+-------------------+-----------+---------------+
| Id | Name   | Availability Zone | Hosts      | Metadata        |
+----+--------+-------------------+-----------+---------------+
| 5  | NonQAT | -                 | 'grunt122’ | 'NonQAT=True'   |
+----+--------+-------------------+-----------+---------------+
```

4.7.3 Configure Host Aggregate for grunt21

Follow the above steps to configure host aggregates for grunt21 as shown below.

- Trafgen

```
+----+---------+-------------------+-----------+----------------+
| Id | Name    | Availability Zone | Hosts     | Metadata        |
+----+---------+-------------------+-----------+----------------+
| 1  | Trafgen | -                 | 'grunt109' | 'Trafgen=True'  |
+----+---------+-------------------+-----------+----------------+
```

- Trafsink

```
+----+----------+------------------+-----------+-----------------+
| Id | Name     | Availability Zone | Hosts    | Metadata        |
+----+----------+------------------+-----------+-----------------+
| 2  | Trafsink | -                | 'grunt110' | 'Trafsink=True' |
+----+----------+------------------+-----------+-----------------+
```

5.0 Operation

5.1 Create a Cloud Account in RIFT.ware Launchpad*

1. Access the RIFT.ware Launchpad* GUI through https://192.168.12.50:8000, where 192.168.12.50 is the floating IP address of the VM hosting RIFT.ware Launchpad. Sign in with `admin/admin` as username/password.

Note: If you follow the Install Rift.ware from an RPM section from the RIFT.ware Installation Guide instead of the section 3.3 Install RIFT.ware from RPM Package Manager, then RIFT.ware 4.2.2 instead of RIFT.ware 4.2.1 will be installed, and then you should access RIFT.ware Launchpad GUI through https://192.168.12.50:8443.

2. Create an OpenStack cloud account that manages the grunt106 controller for Demo 1A, Demo 2A, Demo 2B, Demo 3.
   - On the RIFT.ware Launchpad GUI, select **ACCOUNTS** from the drop-down menu.
   - On the accounts page, click **ADD CLOUD ACCOUNT**.
   - Type a name for the account.
   - Select **ACCOUNT TYPE** as OpenStack and provide the following **ACCOUNT DETAILS**.
     - **KEY:** `demo`
     - **SECRET:** respective password for demo user
     - **AUTHENTICATION URL:** http://192.168.12.106:5000/v3

Note: For the Authentication URL, use "v3", even if the Identity Service Endpoint on the OpenStack Project > Compute > Access & Security > API Access page displays a different version.

   - **TENANT:** `demo`
   - **MANAGEMENT NETWORK:** `private`
   - Click **SAVE**.

3. Do the same to add the cloud account for grunt21. Use a different name, and replace the IP address with 192.168.12.121 (grunt21 IP address) in **AUTHENTICATION URL**.

5.2 Prepare the Image, Network Service Descriptor (NSD), and VNF Descriptor (VNFD)

5.2.1 Download the Image, NSD, and VNFD

From http://repo.riftio.com/releases/open.riftio.com/4.2.1/VNFS, download `rel_4.2.1_demos.zip` and unzip it. The package contains the images and VNF/NS descriptors needed for the three demos:

- `rift-root-latest-multivm-vnf.qcow2`
- `dkpi_1a_tg_ts_multivm_nsd.tar.gz`
- `dkpi_1a_trafgen_vnfd.tar.gz`
- `dkpi_1a_trafsink_vnfd.tar.gz`
- `dkpi_1b_tg_ts_multivm_nsd.tar.gz`
- `dkpi_1b_trafgen_vnfd.tar.gz`
- `dkpi_1b_trafsink_vnfd.tar.gz`
- `dkpi_2a_cag_pgw_multivm_nsd.tar.gz`
- `dkpi_2a_cag_vnfd.tar.gz`
- `dkpi_2a_pgw_vnfd.tar.gz`
- `dkpi_2b_cag_pgw_multivm_nsd.tar.gz`
- `dkpi_2b_cag_vnfd.tar.gz`
- `dkpi_2b_pgw_vnfd.tar.gz`
- `dkpi_3_slb_vnfd.tar.gz`
- `dkpi_3_tg_slb_ts_multivm_nsd.tar.gz`
- `dkpi_3_trafgen_vnfd.tar.gz`
- `dkpi_3_trafsink_vnfd.tar.gz`

5.2.2 Upload the Image to the OpenStack

With web browser, log in to OpenStack Dashboard* at http://192.168.12.106 and http://192.168.12.121 with `demo` user credentials, and upload the `rift-root-latest-multivm-vnf.qcow2` file to the demo project. Keep the image name as `rift-root-latest-multivm-vnf.qcow2`.

Note: Demo 1B uses DevStack* to install the OpenStack. It may not be able to upload the image through the local file. The mitigation is to upload the image through location.

For example, download the image to the Cobbler server, and under that folder, run `python -m SimpleHTTPServer 8000`.

Use a web browser to verify that http://192.168.12.111:8000 is working, and then, in the OpenStack Dashboard of grunt21, use `http://192.168.12.111:8000/rift-root-latest-multivm-vnf.qcow2` as the image location to upload the image.

5.2.3 Upload VNFD and NSD to RIFT.ware Launchpad

On the RIFT.ware Launchpad GUI, open the **CATALOG** page and onboard all the VNF descriptors (VNFD) and network service descriptors (NSD) for Demo 1, Demo 2, and Demo 3.

1. In the RIFT.ware Launchpad GUI, open the **CATALOG** page, and click the **VNFD** tab below the **DESCRIPTOR CATALOGS** label. Click the upload icon, and navigate to the `rel_4.2.1_demos` folder that was downloaded and unzipped in section 5.2.1. Select all the VNFDs (all the tar.gz files that contain `vnfd` string in the name).

2. After VNFDs are successfully uploaded, in the RIFT.ware Launchpad GUI, open the CATALOG page, and click the NSD tab below the **DESCRIPTOR CATALOGS** label. Click the upload icon, and navigate to the `rel_4.2.1_demos` folder that was downloaded and unzipped in section 5.2.1. Select all the NSDs (all the tar.gz files that contain `nsd` string in the name).

5.3 Run Demo 1A/1B

5.3.1 Instantiate Demo 1A Network Service Record (NSR)

1. Open the RIFT.ware Launchpad Dashboard, and select **INSTANTIATE** from the drop-down menu.
2. Select the `tg_ts_1a_nsd`.
3. Choose the cloud account for grunt106 created in section 5.1.
4. In the 3. **CONFIGURE NSD** panel, under **INSTANCE**, enter demo1a for **NAME**. Under **NS PLACEMENT GROUPS**, add a host aggregates filter using the following key/value pairs:
   - With the `Trafgen` group, use Key: `Trafgen` / Value: `True`
   - With the `Trafsink` group, use Key: `Trafsink` / Value: `True`
5. Click **LAUNCH**, and wait for the **NETWORK SERVICES STATUS** to become Running, and the **NETWORK SERVICE DETAILS** card to show `Active` status.
6. Navigate to the **VIEWPORT** page and click the **NSR CONFIG PRIMITIVES** tab.
7. Start traffic by entering `Start` in the **TRAFFIC TRIGGER** text window and clicking the **TRAFFIC** button.
8. Optionally, stop the traffic by entering `Stop` in the **TRAFFIC TRIGGER** text window and clicking the **TRAFFIC** button.

5.3.2 Instantiate Demo 1B NSR

1. Open the RIFT.ware Launchpad Dashboard, and select **INSTANTIATE** from the drop-down menu.
2. Select the `tg_ts_1b_nsd`.
3. Choose the cloud account for grunt21 created in section 5.1.
4. In the **3. CONFIGURE NSD** panel, under **INSTANCE**, enter demo1b for **NAME**. Under **NS PLACEMENT GROUPS**, add a host aggregates filter using the following key/value pairs:
   - With the `Trafgen` group, use Key: `Trafgen` / Value: `True`
   - With the `Trafsink` group, use Key: `Trafsink` / Value: `True`
5. Click **LAUNCH**, and wait for the **NETWORK SERVICES STATUS** to become Running, and the **NETWORK SERVICE DETAILS** card to show `Active` status.
6. Navigate to the **VIEWPORT** page and click the **NSR CONFIG PRIMITIVES** tab.
7. Start traffic by entering `Start` in the **TRAFFIC TRIGGER** text window and clicking the **TRAFFIC** button.
8. Optionally stop traffic by entering `Stop` in the **TRAFFIC TRIGGER** text window and clicking the **TRAFFIC** button.
9. Optionally, change the packet size and sending rate by entering values into **PACKET SIZE** and **TX RATE** text windows, then entering Start in the **TRAFFIC TRIGGER** text window, and clicking the **TRAFFIC** button.

5.4 Run Demo 2

5.4.1 Instantiate Demo 2A

1. Open the RIFT.ware Launchpad Dashboard, and select **INSTANTIATE** from the drop-down menu.
2. Select the `cag_pgw_2a_nsd`.
3. Choose the cloud account for grunt106 created in section 5.1.
4. In the 3. **CONFIGURE NSD** panel, under **INSTANCE**, enter `demo2a` for **NAME**. Under **NS PLACEMENT GROUPS**, add a host aggregates filter using the following key/value pair, Key: `NonQAT`, Value: `True`.

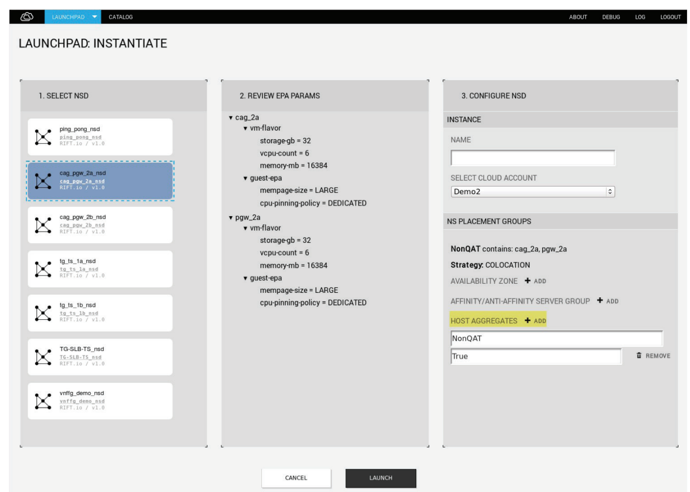


Figure 5. Instantiate Demo 2A.

5. Click **LAUNCH**, and wait for the **NETWORK SERVICES STATUS** to become Running, and the **NETWORK SERVICE DETAILS** card to show `Active` status. This step creates the RIFT.ware CAG and RIFT.ware PGW, each with four IPsec services.
6. Navigate to the **VIEWPORT** page and click the **NSR CONFIG PRIMITIVES** tab.
7. Start the traffic by entering `Start` in the **TRIGGER** text window and 35 in the **TUNNELS** text window. Click the **IKE TRAFFIC** button. This setting results in 35 × 4 = 140 tunnels initiated per PGW.

5.4.2 Instantiate Demo 2B

1. Open the RIFT.ware Launchpad Dashboard, and select **INSTANTIATE** from the drop-down menu.
2. Select the `cag_pgw_2b_nsd`.
3. Choose the cloud account for grunt106 created in section 5.1.
4. In the **3. CONFIGURE NSD** panel, under **INSTANCE**, enter `demo2b` for **NAME**.
5. Click **LAUNCH**, and wait for the **NETWORK SERVICES STATUS** to show Running, and the **NETWORK SERVICE DETAILS** card to show `Active` status. This step creates the RIFT.ware CAG and RIFT.ware PGW, each with four IPsec services.
6. Navigate to the **VIEWPORT** page and click the **NSR CONFIG PRIMITIVES** tab.
7. Start the traffic by entering `Start` in the **TRIGGER** text window and `700` in the **TUNNELS** text window. Click the **IKE TRAFFIC** button. This setting results in 700 × 4 = 2800 tunnels initiated per RIFT.ware PGW, which is 20× more than in Demo 2A.

5.4.3 Delete Tunnels

Optionally, you can delete tunnels started in sections 5.4.1 and 5.4.2. To delete the tunnel,

1. Navigate to the **VIEWPORT** page and click the **NSR CONFIG PRIMITIVES** tab.
2. Stop the traffic by entering `Stop` in the **TRIGGER** text window and the number of tunnels need to be stopped in the **TUNNELS** text window. Click the **IKE TRAFFIC** button.

5.5 Run Demo 3

1. Open the RIFT.ware Launchpad Dashboard, and select **INSTANTIATE** from the drop-down menu.
2. Select the `TG-SLB-TS_nsd`.
3. Choose the cloud account for grunt106 created in section 5.1.
4. In the 3. **CONFIGURE NSD** panel, under **INSTANCE**, enter `demo3` for **NAME**. Under **NS PLACEMENT GROUPS**, add two host aggregates filters using the following key/value pairs:
   - With the `SLB` group, Key: `CAT` / Value: `True`
   - With the `Trafgen`, `Trafsink` group, Key: `NonCAT` / Value: `True`
5. Click **LAUNCH**, and wait for the **NETWORK SERVICES STATUS** to become `Running`, and the **NETWORK SERVICE DETAILS** card to show `Active` status.

   This step creates and configures three VNFs: RIFT.ware Trafgen, RIFT.ware SLB, and RIFT.ware Trafsink.
6. Navigate to the **VIEWPORT** page and click the **NSR CONFIG PRIMITIVES** tab.
7. Perform the following actions:
   - Start or stop traffic by entering `Start` or `Stop` in the **TRAFFIC TRIGGER** text window and clicking the **TRAFFIC** button.
   - Start or stop noisy neighbor by entering `Start` or `Stop` in the **NOISY NEIGHBOR** text window, and the host address (the physical server), which you can find on the OpenStack Dashboard. In our example, it is 192.168.12.104.
   - Start or stop CAT by entering `Start` or `Stop` in the **CAT** text window, and the host address, which you can find on the OpenStack Dashboard. In our example, it is 192.168.12.104.

6.0 Validation

6.1 Demo 1

1. Follow section 5.3.1 to run the Demo 1A. Start the traffic and measure the performance.

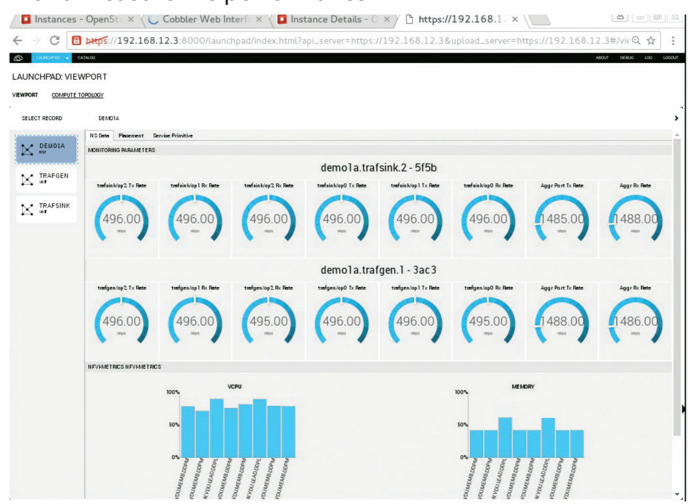


Figure 6. Demo 1A result.

In Demo1A, without EPA and with native Open vSwitch, the aggregated receiving rate for RIFT.ware Trafgen is 1,486 kbps.

2. Follow section 5.3.2 to run Demo 1B. Start the traffic and measure the performance.

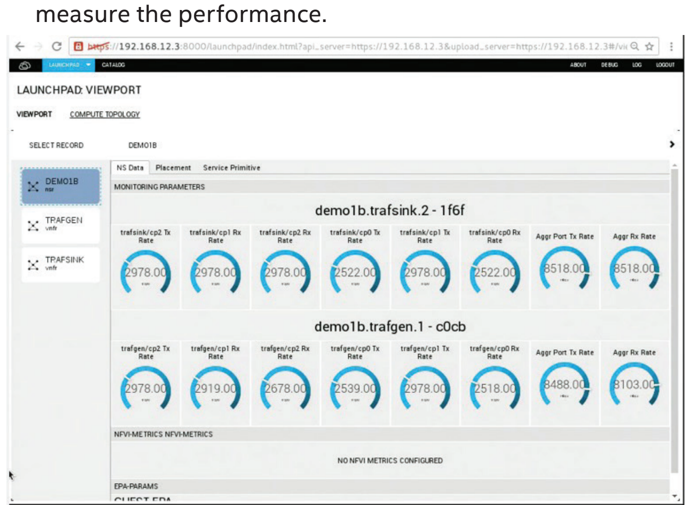


Figure 7. Demo 1B test result.

In Demo1B, with EPA and OVS-DPDK, the aggregated receiving rate for RIFT.ware Trafgen is 8,103 kbps, which is 5.75× better than in Demo1A.

6.2 Demo 2

1. Follow section 5.4.1 to run Demo 2A. Start from 35 tunnels, and test the best achievable performance of Demo 2A (without Intel QuickAssist Technology).

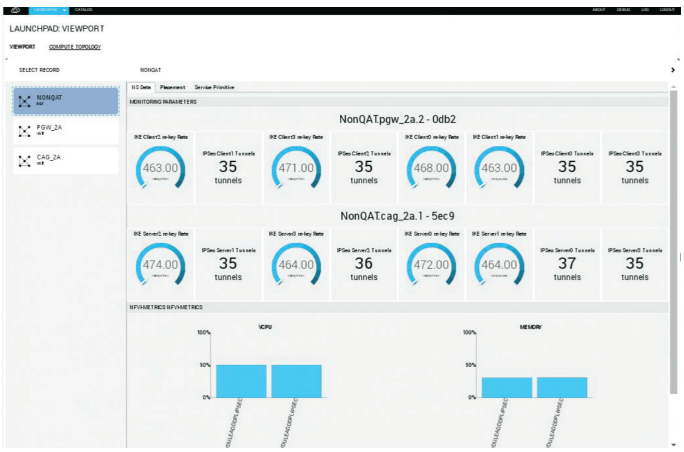


Figure 8. Demo 2A test result.

In Demo2A, without Intel QuickAssist Technology, IPSec NS can support 35 × 4 = 140 tunnels, and the rekey rate is around 1,880 per second.

2. Follow section 5.4.2 to run the Demo 2B. Based on the maximum number of tunnels achieved in step 1, verify whether it is possible to achieve 20× of that number in Demo 2B (with Intel QuickAssist Technology). On grunt116, execute also the following commands to check whether Intel QuickAssist Technology keeps on working.

```
# watch –d –n 0 "cat /proc/icp_dh895xcc_
dev0/qat"
# watch –d –n 0 "cat /proc/icp_dh895xcc_
dev1/qat"
```

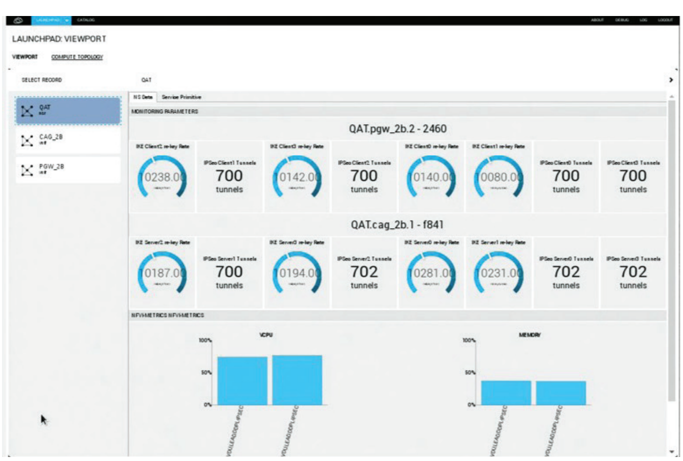


Figure 9. Demo 2B test result.

In Demo2B, with Intel QuickAssist Technology, IPSec NS now can support 700 × 4 = 2,800 tunnels, and the rekey rate is around 40,400 per second. Both the tunnel number and rekey rate are 20× higher than in Demo2A.

6.3 Demo 3

1. Follow section 5.5 to run Demo 3.
2. Without starting noisy neighbor traffic and without CAT, start the traffic, and observe the traffic is being sent and received on the trafgen/cp0 port. Note the throughput on the RX of that port.

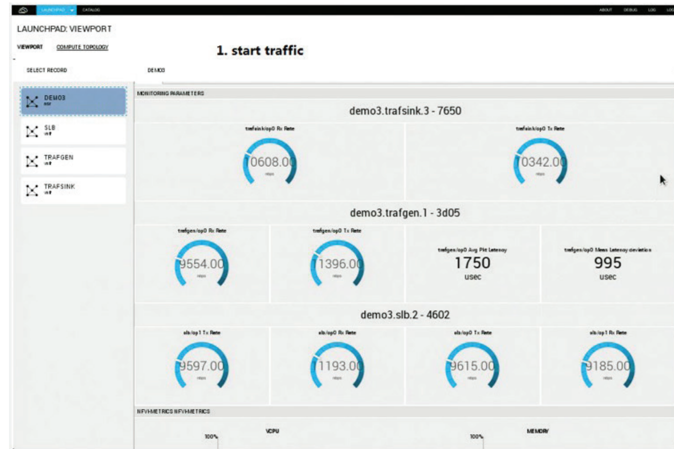


Figure 10. Demo 3 test result—start traffic.

The performance is approximately 9,500 kbps.

3. Start the noisy neighbor on grunt104, and observe the traffic being received on the trafgen/cp0 port is less than the value from Step 2.

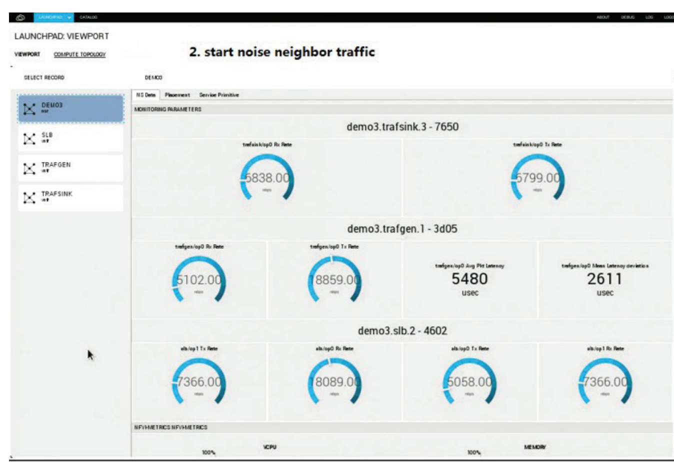


Figure 11. Demo 3 test result—enable noise traffic.

The performance dropped to around 5,100 kbps.

4. Start CAT on grunt104, and observe that the traffic being received on the trafgen/cp0 port is more than the value observed in step 3, which shows that due to the CATfeature, the throughput is higher.

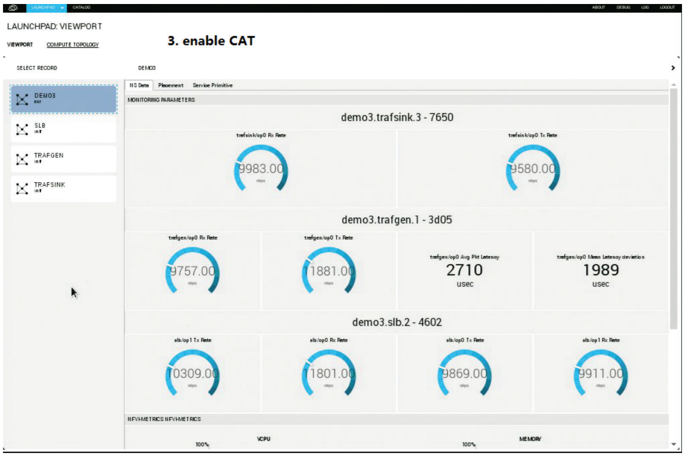


Figure 12. Demo 3 test result— enable CAT.

The performance increases to around 9,700 kbps.

5. Stop CAT on grunt104, and observe that the traffic being received on the trafgen/cp0 port is now almost the same as in the step 3, and less than in the step 4, which shows that due to disabling the CAT feature, the throughput is again affected by noisy neighbor. Meanwhile, use the pqos command to observe cache utilization.

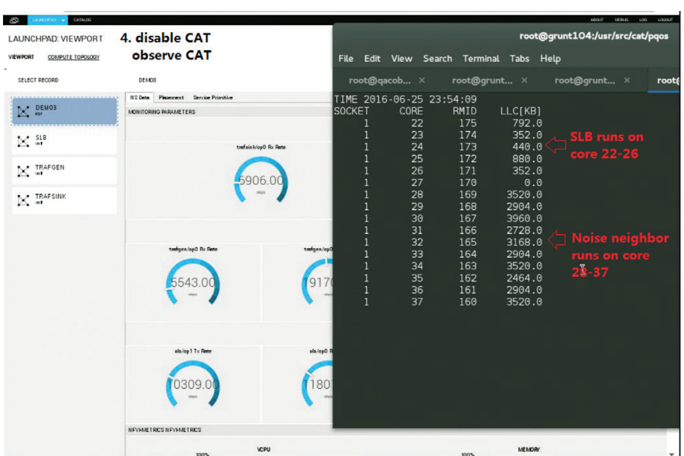


Figure 13. Demo 3 test result—disable CAT, observe cache utilization through pqos.

The performance dropped again to around 5,500 kbps.

6. Start CAT on grunt104 again, and observe that the traffic being received on the `trafgen/cp0` port is increasing again. Meanwhile, use the `pqos` command to observe cache utilization.

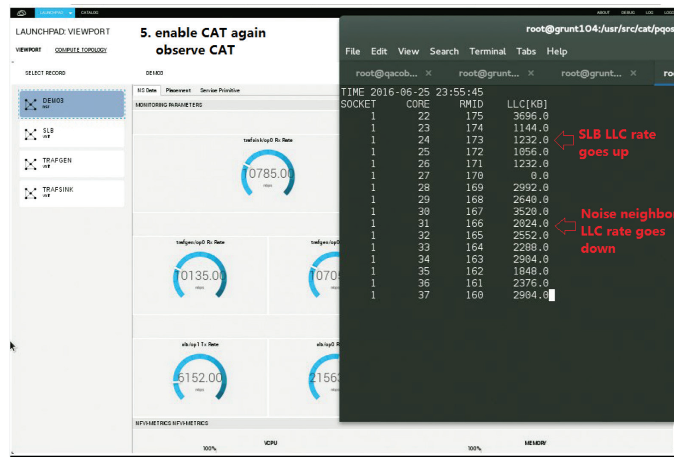


Figure 14. Demo 3 test result—enable CAT, observe cache utilization through pqos.

The performance increases again to around 10,100 kbps.

All above steps demonstrate that CAT can effectively help avoid cache conflicts and improve performance.

Appendix A: References

REFERENCE	SOURCE
Cobbler	http://cobbler.github.io/
DevStack	https://git.openstack.org/cgit/openstack-dev/devstack
HP ProLiant DL380 Gen9	http://www8.hp.com/us/en/products/proliant-servers/product-detail.html?oid=7271241
Intel Ethernet Controller I350: Datasheet	http://www.intel.com/content/www/us/en/embedded/products/networking/ethernet-controller-i350-datasheet.html
Intel Ethernet Converged Network Adapter XL710-QDA2	http://ark.intel.com/products/83967/Intel-Ethernet-Converged-Network-Adapter-XL710-QDA2
Intel Ethernet Server Adapter X520-DA2	http://ark.intel.com/products/55353/Intel-Ethernet-Server-Adapter-X520-DA2
Intel QuickAssist Adapter 8950	http://ark.intel.com/products/79483/Intel-QuickAssist-Adapter-8950
Intel Server Board S2600GZ	http://ark.intel.com/products/56253/Intel-Server-Board-S2600GZ
Intel Server Board S2600WT2	http://ark.intel.com/products/82155/Intel-Server-Board-S2600WT2
Packstack	https://wiki.openstack.org/wiki/Packstack
Quanta Computer Inc. QuantaGrid D51B-1U	http://www.qct.io/account/download/download?order_download_id=759&dtype=Datasheet
RIFT.ware 4.2 installation and operation guide	https://open.riftio.com/webdocs/RIFTware-4.2
RIFT.ware 4.2 Release	http://repo.riftio.com/releases/open.riftio.com/4.2.1/
RIFT.ware installation video	https://open.riftio.com/videos/

Appendix B: Abbreviations

ABBREVIATION	DESCRIPTION
BIOS	Basic I/O System
RIFT.ware CAG	RIFT.ware Converged Access Gateway
CAT	Cache Allocation Technology
CPU	Central Processing Unit
Intel® DDIO	Intel® Data Direct I/O Technology
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
DPDK	Data Plane Development Kit
EPA	Enhanced Platform Awareness
GRUB	GNU GRand Unified Bootloader
GUI	Graphical UI
I/O	Input/Output
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	IP Security
LAN	Local Area Network
LOM	LAN-on-Motherboard
MAC	Media Access Control
MANO	Management and Orchestration
NFV	Network Functions Virtualization
NS	Network Service
NSD	Network Service Descriptor

ABBREVIATION	DESCRIPTION
NSR	Network Service Record
NUMA	Non-Uniform Memory Access
OVS-DPDK	DPDK-accelerated Open vSwitch
RIFT.ware PGW	RIFT.ware Premises Gateway
PXE	Preboot eXecution Environment
RPM	RPM Package Manager
RX	Receive
SATA	Serial Advanced Technology Attachment
SELinux	Security-Enhanced Linux
RIFT.ware SLB	RIFT.ware Scriptable Load Balancer
SR-IOV	Single Root I/O Virtualization
SSH	Secure Shell
TCP	Transmission Control Protocol
TX	Transmit
UI	User Interface
URL	Uniform Resource Locator
vCPU	Virtual CPU
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtual Network Functions
VNFD	VNF Descriptor