WHITE PAPER

Service Provider
Testing the NVMe P3700 card and the Intel® QuickAssist Adapter

Using OpenStack & Enhanced Platform Awareness for the NVMe P3700 card and the Intel® QuickAssist Adapter

A guide to testing and validation of the NVMe P3700 card and the Intel® QuickAssist Adapter

Author

Luc Provoost
NFV Solution Architect

Table of Contents

Introduction

Service Providers (SPs) face severe challenges to both their business models and their margins. The reasons for this are widely understood and discussed: the breadth and aggressiveness of competition, the extraordinary increase in data traffic, IoT, device diversification and the decline in voice revenue.

Facing these challenges brings a number of imperatives for SPs, among them:

- Reducing the cost of creating and launching innovative services
- Improving the customer experience
- Delivering right-sized capacity in the network
- Bringing services to market faster through increased agility

Intel, along with a substantial set of ecosystem partners, is helping SPs with a range of solutions to foster agility, accelerate network efficiency and improve customer service – from providing the infrastructure and architecture to deliver a 360° customer profile, to various solutions which support aspects of Network Functions Virtualization (NFV). Together these solutions provide SPs with the tools and architectures they need to manage what Gyanee Dewnarain, Research VP, Gartner calls “an unparalleled transformational force that is completely reshaping the telecommunications industry.”

This document offers very specific advice about one of these solutions – testing the use of PCI cards to accelerate access to storage and speed encryption in virtualized environments. By keeping the storage and processing close to each other in the OpenStack virtualized environment, the solution decreases latency and increases response for certain applications where the databases are relatively small, but the speed of response is critical either to customer experience or to network performance. These may include, for the customer, information about parental controls, service and content subscriptions, security credentials, or other indices where any latency in the response directly affects customer experience. In the network typical use cases might include various kinds of sophisticated load balancing (where the criteria are more complex than held in memory) and the acceleration of other kinds of stateful Virtualised Network Functions (VNFs).

For both the NVMe P3700 card and an Intel® QuickAssist Adapter, a method is defined to test basic performance to check that native performance is achieved in the VMs launched by OpenStack using the fio and Data Plane Development Kit (DPDK) test application.

1 Summary

Virtual Infrastructure Managers (VIMs) like OpenStack are abstracting the underlying HW through virtualization.

The Limits of Architectural Abstraction in Network Function Virtualization describes in detail the effects on Border Network Gateway (BNG) performance when tuning and optimizing the platforms by hand and states that an Orchestrator that is aware of these factors can make better decisions during VNF deployment.

In order to reach certain data plane performance levels, it is better to have the VIM know more details about the underlying HW so it can make better decisions on work placement with high performance in mind. Enhanced Platform Awareness (EPA) is the terminology Intel is using to describe the features added in OpenStack and other VIMs.

A Path to Line-Rate-Capable NFV Deployments with Intel® Architecture and the OpenStack* Kilo Release details how the different EPA technologies can be enabled in OpenStack for the same BNG example as referenced before.

While the use cases referenced above are focused on the networking applications using EPA, this paper is giving an example of how to use EPA and PCI pass-through with two different PCI cards in particular: a NVMe P3700 card and an Intel QuickAssist Adapter. The paper also shows an easy way to check that native performance is actually achieved in the VMs.

2 Scope

This document is listing the steps that have to be taken in order to use the PCI cards in PCI pass-through in an OpenStack environment. It will guide you through the configuration and SW installation that needs to be done in Fuel, the controller and the compute platforms.

For both the NVMe P3700 card and an Intel QuickAssist Adapter, a method is defined to test basic performance to check that native performance is achieved in the VMs launched by OpenStack using the fio and a DPDK test application.

Engineers looking into using these two cards in a PCI pass-through OpenStack environment, can use this document to speed-up and verify their HW and SW installation.

3 Setup

The testing has been done on both the OPNFV Brahmaputra & Colorado releases which are based on the Liberty & Mitaka OpenStack releases. Fuel is used to deploy the OpenStack environment.

Only one controller and one compute node are deployed. The compute node has the two PCI cards used in the tests and they are connected to two different NUMA nodes.

In the following sections, we will describe the setup of Fuel, BIOS and OpenStack of both the controller and compute node. The test setup will then demonstrate that “native performance” is indeed achieved.

4 NVMe P3700

In an OpenStack environment, storage is one of the key resources that are offered to the user like compute and networking. With the availability of Intel® SSD Data Center Family for NVMe*, these fast storage devices can now also be offered directly to a VNF if fast local storage is required. Functions that need to update or consult customer-related data in “real time” like HSS and Session Control functions could benefit from this way of storing their data.

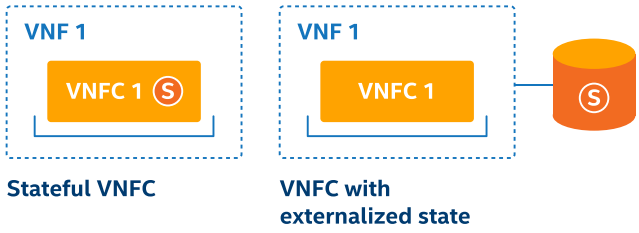

Figure 2. VNFC stateful patterns

More generically, a stateful Virtual Network Function Component (VNFC) or the external storage for a VNFC with externalized state as described in chapter 5.1.3 of the Virtual Network Functions Architecture, can benefit from NVMe devices in PCI pass-through.

OpenStack allows to deploy such a VNF on a compute host that has the required HW in PCI pass-through to place this function. We will now describe in detail what needs to be done and show that this results in native performance, similar to the performance without virtualization.

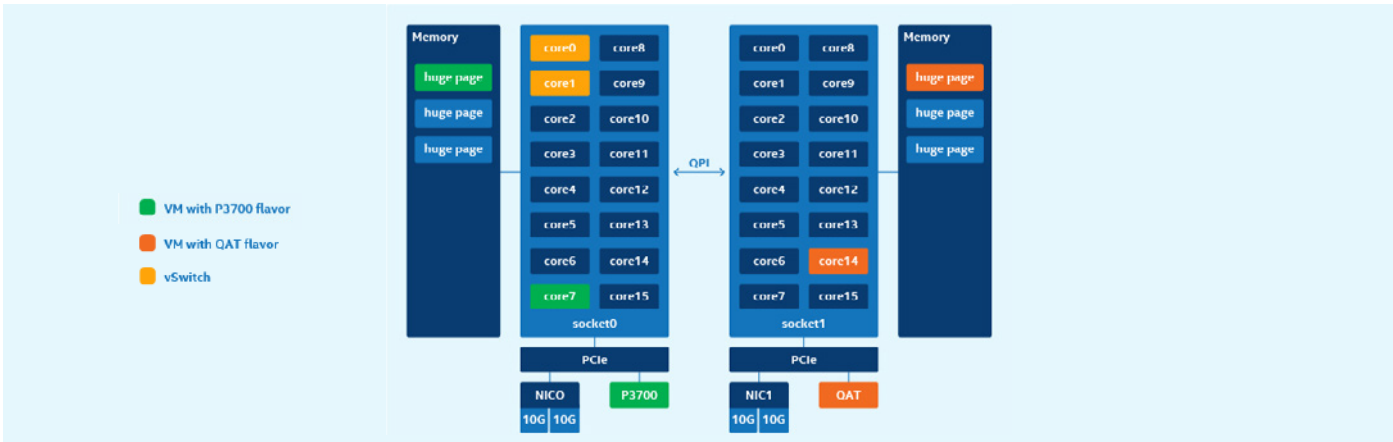

Figure 1. Compute node

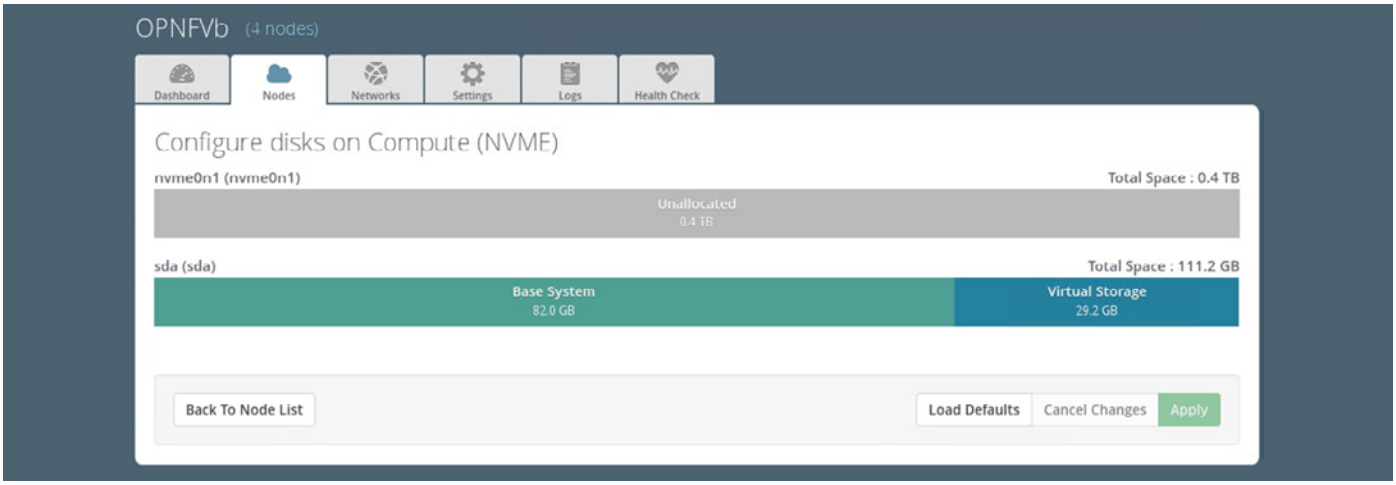

Figure 3. Fuel disk configuration screenshot

4.1 Fuel installation

Standard Fuel Installation will use the available NVMe devices as storage to be used as Base System or Virtual Storage by OpenStack. The NVMe devices should first be marked as "unallocated" by Fuel. This can be achieved using the Fuel GUI or using the Fuel command line.

In the following example, Fuel command lines are used to unallocate the NVMe drive on compute node 4:

```
[root@fuel ~]# fuel node --node-id 4 --disk --download
```

Node attributes for disks were written to:

```
/root/node_4/disks.yaml
```

Then edit the disks.yaml file as shown in the example (0 size for OS) and upload the file:

```
[root@fuel ~]# fuel node --node-id 4 --disk --upload
```

Node attributes for disks were uploaded from.

```
- extra: []
  id: nvme0n1
  name: nvme0n1
  size: 380926
  volumes:
  - keep_data: false
    name: os
    size: 0
  - keep_data: false
    name: vm
    size: 380926
- extra:
  - disk/by-id/wwn-0x50015178f35cd63a
  - disk/by-id/ata-INTEL_SSDSC2BB120G4_BTWL306400DT120LGN
  id: sda
  name: sda
  size: 113845
  volumes:
  - keep_data: false
    name: os
    size: 83968
  - keep_data: false
    name: vm
    size: 29877
```

4.2 Preparation of the compute platform

Ensure the card is actually inserted in a PCI gen 3 slot. If not, you will measure the limits of PCI gen 2 and not the capability of the NVMe device.

The BIOS on the compute platform needs to be enabled for Intel® Virtualization Technology and for Intel® Virtualization Technology for Directed I/O.

Figure 4. Screenshots for BIOS setup

Note that the default on some systems is <Disabled>. The system needs to be powered off before this setting takes effect.

You should also update the GRUB_CMDLINE_LINUX_ DEFAULT to enable iommu.

```
vim /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on"
update-grub
```

Now edit the nova.conf file on the compute node with the NVMe device and add the following line:

```
pci_passthrough_whitelist=[{ "vendor_id":"8086","product_
id":"0953"}]
```

Then restart nova compute

```
service nova-compute restart
```

It is preferable to also install the Intel® SSD Data Center Tool (isdct) on the host: in this way you can also test performance on the host allowing to compare performance between running the NVMe application on the host and in a virtual machine (VM). This is described in section 4.4 with results on the host in the window below.

```
root@node-4:/mnt/nvme# fio test.ini
job1: (g=0): rw=read, bs=64K-64K/64K-64K/64K-64K, ioengine=libaio,
iodepth=128
fio-2.1.3
Starting 1 process
Jobs: 1 (f=1): [R] [100.0% done] [3214MB/0KB/0KB /s] [51.5K/0/0 iops] [eta
00m:00s]
job1: (groupid=0, jobs=1): err= 0: pid=57256: Fri Mar 18 12:36:08 2016
  read : io=20480MB, bw=3212.5MB/s, iops=51392, runt=  6376msec
    slat (usec): min=2, max=597, avg= 4.99, stdev= 2.43
    clat (usec): min=908, max=5433, avg=2484.00, stdev=148.86
     lat (usec): min=914, max=5436, avg=2489.05, stdev=148.82
    clat percentiles (usec):
     |  1.00th=[ 2128],  5.00th=[ 2192], 10.00th=[ 2352], 20.00th=[ 2416],
     | 30.00th=[ 2448], 40.00th=[ 2448], 50.00th=[ 2480], 60.00th=[ 2512],
     | 70.00th=[ 2512], 80.00th=[ 2544], 90.00th=[ 2608], 95.00th=[ 2832],
     | 99.00th=[ 2928], 99.50th=[ 3056], 99.90th=[ 3280], 99.95th=[ 3408],
     | 99.99th=[ 4448]
    bw (MB  /s): min= 3191, max= 3226, per=99.99%, avg=3211.58, stdev=
8.99
    lat (usec) : 1000=0.01%
    lat (msec) : 2=0.05%, 4=99.92%, 10=0.02%
  cpu          : usr=4.60%, sys=32.41%, ctx=241211, majf=0, minf=1636
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%,
>=64=100.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%,
>=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%,
>=64=0.1%
     issued    : total=r=327680/w=0/d=0, short=r=0/w=0/d=0

Run status group 0 (all jobs):
   READ: io=20480MB, aggrb=3212.5MB/s, minb=3212.5MB/s,
maxb=3212.5MB/s, mint=6376msec, maxt=6376msec

Disk stats (read/write):
  nvme0n1: ios=315955/9, merge=0/0, ticks=783280/0, in_queue=783916,
util=98.51%
```

4.3 Preparation of the controller platform

Edit nova.conf on the controller by adding a pci_alias and giving it a name, in this case “nvme”. This is the name that will be used when defining the flavor that will be used to create the VM.

```
pci_alias={"vendor_id":"8086", "product_id":"0953",
"name":"nvme"}
```

Also, make sure that the PciPassthroughFilter is added to the list of filters that will be used by the nova scheduler to select a suitable compute host.

```
scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
scheduler_available_filters=nova.scheduler.filters.all_filters
scheduler_available_filters=nova.scheduler.filters.pci_passthrough_filter.
PciPassthroughFilter
scheduler_default_filters=RetryFilter,Availability
ZoneFilter,RamFilter,CoreFilter,DiskFilter,ComputeFilter,
ComputeCapabilitiesFilter,ImagePropertiesFilter,Server
GroupAntiAffinityFilter,ServerGroupAffinityFilter,
PciPassthroughFilter
```

Then restart the scheduler:

```
service nova-scheduler restart
```

Now one can add some extra specifications to a flavor in the GUI or through the command line:

```
nova flavor-key  nfv set  "pci_passthrough:alias"="nvme:1"
```

This flavor will instruct the scheduler to select the proper host on which it can create a VM that gets 1 nvme PCI device in pass-through.

4.4 Installing Intel® SSD Data Center Tool and running the test

It is preferable to also install the isdct on the host: in this way you can also test performance on the host allowing to compare performance between running the NVMe application on the host and in a virtual machine.

```
wget https://downloadmirror.intel.com/23931/eng/
DataCenterTool_3_0_0_Linux.zip
unzip DataCenterTool_3_0_0_Linux.zip
```

(If not installed, apt-get install alien)

```
alien -i isdct-3.0.0.400-15.x86_64.rpm
isdct show -all -intelssd
isdct delete -intelssd 0
mkfs.ext4 /dev/nvme0n1
mount /dev/nvme0n1 /mnt/nvme/
yum install epel-release
yum install fio or apt-get install fio
```

Then run fio test.ini

With following test.ini

```
[global]
filename=/dev/nvme0n1
ioengine=libaio
direct=1
bs=64k
rw=read
iodepth=128
numjobs=1
buffered=0
size=20G
[job1]
```

Results for this particular test show results above 3Gb/s in the window below (Figure 5) which are very similar to the result on the host.

4.5 Conclusion

Using an NVMe device in an OpenStack environment or any other VIM that enables PCI pass-through results in native performance: We could not determine a performance degradation using the fio test. Hence functions like subscriber databases that need low latency and high

```
[root@node-4 mnt]# fio test.ini
job1: (g=0): rw=read, bs=64K-64K/64K-64K/64K-64K, ioengine=libaio, iodepth=128
fio-2.2.8
Starting 1 process
Jobs: 1 (f=1): [R(1)] [100.0% done] [3216MB/0KB/0KB /s] [51.5K/0/0 iops] [eta 00m:00s]
job1: (groupid=0, jobs=1): err= 0: pid=2883: Fri Mar 25 15:32:47 2016
  read : io=20480MB, bw=3211.6MB/s, iops=51384, runt=  6377msec
    slat (usec): min=4, max=204, avg= 6.28, stdev= 2.90
    clat (usec): min=1261, max=10227, avg=2479.92, stdev=150.30
     lat (usec): min=1270, max=10438, avg=2486.51, stdev=150.88
    clat percentiles (usec):
     |  1.00th=[ 2320],  5.00th=[ 2320], 10.00th=[ 2352], 20.00th=[ 2352],
     | 30.00th=[ 2352], 40.00th=[ 2384], 50.00th=[ 2480], 60.00th=[ 2544],
     | 70.00th=[ 2576], 80.00th=[ 2576], 90.00th=[ 2608], 95.00th=[ 2640],
     | 99.00th=[ 2896], 99.50th=[ 3088], 99.90th=[ 3280], 99.95th=[ 3472],
     | 99.99th=[ 6816]
    bw (MB  /s): min= 3159, max= 3227, per=99.98%, avg=3210.95, stdev=17.09
    lat (msec) : 2=0.05%, 4=99.91%, 10=0.04%, 20=0.01%
  cpu          : usr=13.43%, sys=49.22%, ctx=156341, majf=0, minf=2057
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
     issued    : total=r=327680/w=0/d=0, short=r=0/w=0/d=0, drop=r=0/w=0/d=0
     latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
   READ: io=20480MB, aggrb=3211.6MB/s, minb=3211.6MB/s, maxb=3211.6MB/s, mint=6377msec, maxt=6377msec

Disk stats (read/write):
  nvme0n1: ios=316030/0, merge=0/0, ticks=778808/0, in_queue=779786, util=98.46%
[root@node-4 mnt]#
```

Figure 5. Screenshot showing NVMe performance

throughput could benefit from these NVMe devices and still be deployed using standard OpenStack or other VIMs. More generically, any VNFC that needs to maintain state are good candidates to store their state on this type of device.

Other EPA features than the PCI-pass-through feature, like CPU pinning, could also be used to increase performance of a VNF but were not applied in this particular test setup.

The way the NVMe device is exposed through the flavor in this example setup is good enough for the demo. However, the usage of the "nvme" device in the flavor should be standardized across all VNFs and the NVFi.

5 Intel QuickAssist Adapter

How to use an Intel QuickAssist Adapter is described in Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology. This chapter shows that it is quite easy to realize the same in an OpenStack environment which will deploy a VNF using Intel® QuickAssist technology on a compute host that has the required HW to place this function. A simple test will also demonstrate that the performance of this OpenStack deployment is equivalent to the performance of an adaptor running in the host.

In this setup, we will be using the Virtual Functions of the Intel QuickAssist Adapter.

5.1 Preparation of the compute platform

The same preparation as in section 4.2 needs to be done with regards to enabling Intel Virtualization Technology and for Intel Virtualization Technology for Directed I/O in the BIOS and with regards to the intel_iommu parameter.

Now edit the nova.conf file on the compute node with the NVMe device and add the following line:

```
pci_passthrough_whitelist=[{ "vendor_id":"8086","product_
id":"0443"}]
```

Then restart nova compute

```
service nova-compute restart
```

We need now to install the Virtual Functions for the Intel QuickAssist Adapter so that we can use the card in SR-IOV pass-through.

```
yum install zlib-devel openssl-devel yasm tmux
```

Install now QAT drivers:

```
mkdir /QAT
cd /QAT
# copy qatmux.l.2.3.0-34.tgz to this location
tar zxof qatmux.l.2.3.0-34.tgz
export ICP_WITHOUT_IOMMU=1
./installer.sh install QAT1.6 host
```

You can check the installation by taking a look at:

```
cat /proc/icp_dh895xcc_dev0/version
```

5.2 Preparation of the controller platform

Edit nova.conf on the controller by adding a pci_alias and giving it a name, in this case "qat". This is the name that will be used when specifying the flavor that will be used to create the VM.

```
pci_alias={"vendor_id":"8086", "product_id":"0443",
"name":"qat"}
```

Also, make sure that the PciPassthroughFilter is added to the list of filters that will be used by the nova scheduler to select a suitable compute host. This is also described in section 4.3.

Then restart the scheduler:

```
service nova-scheduler restart
```

Now one can create a flavor in the GUI or through the command line:

```
nova flavor-key  nfv set  "pci_passthrough:alias"="qat:1"
```

This flavor will instruct the scheduler to select the proper host on which it can create a VM that gets 1 QuickAssist PCI device in pass-through.

5.3 Installing the cryptodev test SW

We will be using the DPDK SW in the VM to test the performance of the Intel QuickAssist Adapter. To build DPDK with the AESNI_MB_PMD the user is required to download the multi-buffer library from here and compile it before building DPDK. When building the multi-buffer library it is necessary to have YASM package installed and also requires the overriding of YASM path when building, as a path is hard coded in the Makefile of the release package.

```
yum install yasm
unzip ipsec_043.zip
cd code
make YASM=/usr/bin/yasm
export  AESNI_MULTI_BUFFER_LIB_PATH=/root/ipsec/code
Set CONFIG_RTE_LIBRTE_PMD_AESNI_MB=y in config/common_base
Set CONFIG_RTE_LIBRTE_PMD_QAT=y in confg/common_base
export RTE_SDK=/root/dpdk
export RTE_TARGET=x86_64-native-linuxapp-gcc
cd /sys/bus/pci/drivers/qat_1_6_adfvf
echo -n 0000:00:05.0 > unbind (PCI address is just an example)
echo "8086 0443" > /sys/bus/pci/drivers/igb_uio/new_id
```

Now you can compile the test application

```
make test
```

Make sure to blacklist any NIC device while running the test since they are not used and not properly configured for tunning this test.

```
./test –b 00:03.0
```

5.4 Results of the cryptodev test SW

On the RTE command line, 2 tests can now be run: cryptodev_qat_perftest and cryptodev_aesni_mb_perftest. The former is using the accelerator, the latter is doing the same crypto workload running on the host CPU.

The encryption performance using the Intel QuickAssist Adapter results in 37.89Gbps for large packet sizes, reaching native performance as expected.

5.5 Conclusion

Using an Intel® QuickAssist Adapter in an OpenStack environment or any other VIM that enables PCI pass-through results in native performance. The procedure described in this document gives an easy step-by-step description to verify the results on any OpenStack deployment with an Intel QuickAssist Adapter.

SR-IOV has been used, but this device can also be used only in PCI pass-through without SR-IOV. This will have advantages when VNFs are requiring a predefined level of crypto acceleration, which cannot be guaranteed when using Virtual functions: the crypto accelerator resources will be shared amongst all VNFs using a VF and hence the performance allocated to a specific VNF will depend on what the other VNFs are requesting at that moment.

Using the PCI-pass-through feature, the cores for the VF are allocated automatically on the appropriate NUMA node.

Other EPA features than the PCI-pass-through feature, like CPU pinning, could also be used to increase performance of a VNF but were not applied in this particular test setup.

The way the device is exposed through the flavor in this example setup is good enough for the demo. However, the usage of the “qat” device in the flavor should be agreed upon by all VNFs requiring this feature and the NVF Infrastructure.

```
RTE>>cryptodev_qat_perftest
 + ------------------------------------------------------- +
 + Test Suite : Crypto Device Unit Test Suite
 + ------------------------------------------------------- +

Throughput test which will continually attempt to send AES128_CBC_SHA256_HMAC requests with a constant burst size of 64 while varying payloa
d sizes
Dev No  QP No   Req Size(B)     Num Sent        Num Received    Mrps    Thoughput(Gbps) Retries (Attempted a burst, but the device was busy)
0       0       64              10000000        10000000        6.56    3.36            16652262
0       0       128             10000000        10000000        6.09    6.23            19551655
0       0       256             10000000        10000000        5.31    10.88           23421997
0       0       512             10000000        10000000        5.13    21.01           15728522
0       0       768             10000000        10000000        4.82    29.62           12917580
0       0       1024            10000000        10000000        3.63    29.71           35431774
0       0       1280            10000000        10000000        2.98    30.48           56954292
0       0       1536            10000000        10000000        2.58    31.65           74949242
0       0       1792            10000000        10000000        2.32    33.30           89944385
0       0       2048            10000000        10000000        2.31    37.89           82546237
USER1: CRYPTO_PERF_MBUFPOOL count 8191
 + TestCase [ 0] : test_perf_encrypt_digest_vary_req_size succeeded

Test to measure the IA cycle cost using AES128_CBC_SHA256_HMAC algorithm with a constant request size of 64.
This test will keep retries at 0 and only measure IA cycle cost for each request.
Dev No  QP No   Num Sent        Num Received    Tx/Rx burst     Retries (Device Busy)   Average IA cycle cost (assuming 0 retries)
0       0       4096            4096            2               0                       736
0       0       4096            4096            4               0                       443
0       0       4096            4096            8               0                       336
0       0       4096            4096            16              0                       240
0       0       4096            4096            32              0                       204
0       0       4096            4096            64              0                       186
0       0       4096            4096            128             0                       184
USER1: CRYPTO_PERF_MBUFPOOL count 8191
 + TestCase [ 1] : test_perf_vary_burst_size succeeded
USER1: CRYPTO_PERF_MBUFPOOL count 8191
 + ------------------------------------------------------- +
 + Test Suite Summary
 + Tests Total :        2
 + Tests Skipped :      0
 + Tests Executed :     2
 + Tests Passed :       2
 + Tests Failed :       0
 + ------------------------------------------------------- +
Test OK
RTE>>
```

Figure 6. Screenshot showing encryption performance

6 Glossary

BNG	Border Network Gateway
EPA	Enhanced Platform Awareness
ISDCT	Intel® SSD Data Center Tool
NFV	Network Function Virtualization
NVMe	Non Volatile Memory
QAT	Quick Assist Technology
VF	Virtual Functions
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function

7 Machine description

Item	Description	Notes
Platform	Intel® Server Board S2600WT Family	
Form factor	2U Rack Mountable	
Processor(s)	2x Intel® Xeon® Processor E5-2690 v3	46080KB L3 Cache per CPU, 18 cores per CPU (36 logical cores due to Hyper-threading)
Memory	32 GB RAM (4x 8 GB) per socket	Quad channel 2134 DDR4
BIOS	SE5C610.86B.01.01.0009.060120151350	Hyper-threading enabled Hardware prefetching enabled COD disabled
Openstack	OPNFV Brahmaputra	

8 References

Virtual Network Functions Architecture

The Limits of Architectural Abstraction in Network Function Virtualization

A Path to Line-Rate-Capable NFV Deployments with Intel® Architecture and the OpenStack* Kilo Release