NETRONOME

CoreNIC Installation Verification Guide

Netronome Intelligent Server Adapters

Ubuntu 16.04

Revision: 0.3
September 2017

Note: All commands and scripts contained in this document must be executed with **root** privileges.

1. Removing Existing Software

When installing coreNIC 1.0 it is necessary to remove any previous installations of Agilio OVS and CoreNIC prior to starting the install procedure.

1.1. Uninstalling AOVS

The recommended method to remove AOVS is by running the uninstall script:

```
/opt/netronome/bin/agilio-ovs-uninstall.sh -y
```

1.2. Uninstalling Existing CoreNIC

Remove all BSP and Core NIC: apt-get -y remove ns-agilio-core-nic ; apt-get -y remove nfp-bsp.*

```
apt-get -y remove ns-agilio-core-nic
apt-get -y remove nfp-bsp.*
```

2. Installing CoreNIC 1.1 on Ubuntu 16.04

2.1. Install Dependencies

The following dependencies must be installed before installing coreNIC.

2.1.1. Install the prerequisite packages

```
apt-get install -y make autoconf automake libtool \
gcc g++ bison flex hwloc-nox libreadline-dev libpcap-dev dkms libftdi1 libjansson4 \
libjansson-dev guilt pkg-config libevent-dev ethtool libssl-dev \
libnl-3-200 libnl-3-dev libnl-genl-3-200 libnl-genl-3-dev psmisc gawk \
libzmq3-dev protobuf-c-compiler protobuf-compiler python-protobuf \
linux-headers-$(uname -r)
```

2.2. Install Netronome Packages

2.2.1. NFP BSP package

Install the NFP BSP rpm package provided by Netronome Support.

```
dpkg -i nfp-bsp-6000-*.deb
```

2.2.2. CoreNIC deb package

Next, install the Agilio Core NIC package with the following command

```
dpkg -i ns-agilio-core-nic_1.*.deb
```

The installation process does the following:

- Checks the running kernel has the fix for the PCIe quirk-(upstreamed from kernel versions 4.5+)
- Builds the ‘nfp’ driver modules for all kernels with quirk_nfp6000 fix on them
- Copies the right firmware into /lib/firmware/netronome
- Checks the flash version in the NFP and prints the commands to update NFP flash if required
- Creates nfp_pX netdev interfaces - Can be verified in dmesg

If you see the message below at the end of the installation your NFP device’s flash needs to be updated. Execute the given commands and reboot your machine.

```
======== NFP requires a new ARM flash image ========
        /opt/netronome/bin/nfp-flash --preserve-media-overrides -w
/opt/netronome/flash/flash-nic.bin
        /opt/netronome/bin/nfp-one
        reboot
```

*Note: If your attempt to update the flash fails with the following message, :

```
/opt/netronome/bin/nfp-flash: Failed to open NFP device 0 (No such device)
```

execute the following commands to enable the required card access and rerun the commands (if executing the commands below does not resolve the issue, a reboot is needed):

```
rmmod nfp
modprobe nfp nfp_pf_netdev=0 nfp_dev_cpp=1
```

3. Running Core NIC

3.1. View Interfaces

Once the package has been installed and the card has the correct flash version, new netdev interfaces named nfp_pX will be created e.g. :

```
ip link

18: nfp_p0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT
qlen 1000
    link/ether 00:15:4d:12:2c:ce brd ff:ff:ff:ff:ff:ff
19: nfp_p1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT
qlen 1000
    link/ether 00:15:4d:12:2c:cf brd ff:ff:ff:ff:ff:ff
```

3.2. Configure Interfaces

3.2.1. Port Speed

To configure the port speed of the NFP use the following commands. A reboot is required to take affect.

```
#view current status
nfp-media

#change media mode
nfp-media phy0=10G #set port 0 in 10G mode
nfp-media phy1=25G #set port 1 in 25G mode
nfp-media phy2=4x10G #set port 2 in 4x10G fanout mode
```

3.3. Confirm Connectivity

3.3.1. Allocate IP addresses

Assign IP addresses to the respective interfaces on the NFP.

```
#host_0
ip address add 10.0.0.1/24 dev nfp_p0
ip link set nfp_p0 up

#host_1
ip address add 10.0.0.2/24 dev nfp_p0
ip link set nfp_p0 up
```

3.3.2. Ping interfaces

After you have successfully assigned IP addresses to the NFP interfaces perform a standard ping test to confirm L1-L3 connectivity.

```
#from host_0
ping 10.0.0.2

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.319 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.317 ms
```

4. Basic Performance Test

IPerf3 is a basic traffic generator and network performance measuring tool that allows one to quickly determine the throughput achievable by a device.

4.1. Installing IPerf

```
apt-get install -y iperf3
```

4.2. Running IPerf

4.2.1. Server

Run IPerf3 on the server.

```
iperf3 -s
```

4.2.2. Client

Execute the following on the client to connect to the server and start running the test.

```
iperf3 -c 10.0.0.2 -P 10
```

5. Installing & Configuring DPDK

5.1. Enable IOMMU

In order to bind the device to *VFIO* driver for DPDK testing, the machine has to have IOMMU enabled. Here http://dpdk-guide.gitlab.io/dpdk-guide/setup/binding.html some generic information about binding devices including the possibility of using UIO instead of VFIO, and also mentioning the VFIO no-iommu mode.

Although DPDK is about avoiding interrupts, there is an option of a NAPI-like approach using RX interrupts. This is supported by PMD NFP and with VFIO it is possible to have a RX interrupt per queue (with UIO just one interrupt per device). Because of this VFIO is the preferred option.

5.1.1. Edit grub configuration file

This change is required for working with VFIO, but with kernels > 4.5, it is possible to work with VFIO and no-IOMMU mode.

If your system comes with a kernel > 4.5, you can work with VFIO and no-IOMMU enabling this mode:

echo 1 > /sys/module/vfio/parameters/enable_unsafe_noiommu_mode

For kernels older than 4.5, working with VFIO requires to enable IOMMU in the kernel at boot time. Add the necessary kernel parameters to **`/etc/default/grub`**

```
GRUB_CMDLINE_LINUX_DEFAULT="... intel_iommu=on iommu=pt intremap=on"
```

It is worth to note *iommu=pt* is not required for DPDK if VFIO is used, but it helps for avoiding a performance impact in host drivers, like the NFP netdev driver, when *intel_iommu=on* is enabled.

5.1.2. Implement changes

Apply changes and reboot.

```
update-grub
reboot
```

5.2. Build DPDK-ns

Download the dpdk-ns archive and perform the following steps to build dpdk.

```
#build DPDK
cd /root
tar zxvf dpdk-ns.tar.gz
cd dpdk-ns

export RTE_SDK=/root/dpdk-ns
export RTE_TARGET=x86_64-native-linuxapp-gcc
make T=$RTE_TARGET install
```

5.3. Configure hugepages

Allocate some hugepages for the DPDK apps to use.

```
#configure hugepages
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

5.4. Bind DPDK Driver

5.4.1. Enable NFP interfaces

```
nfp -m mac set port ifup 0 0-8
```

5.4.2. Attach vfio-pci driver

```
#bind vfio-pci driver
modprobe vfio-pci

#Physical Function
rmmod nfp
echo 19ee 4000 > /sys/bus/pci/drivers/vfio-pci/new_id

#Virtual Function
echo 19ee 6003 > /sys/bus/pci/drivers/vfio-pci/new_id
```

5.4.3. Attach igb-uio driver

```
#bind igb-uio driver
modprobe uio
DRKO=$(find ~/dpdk-ns -iname 'igb_uio.ko' | head -1 )
insmod $DRKO

#Physical Function
rmmod nfp
echo 19ee 4000 > /sys/bus/pci/drivers/igb_uio/new_id

#Virtual Function
echo 19ee 6003 > /sys/bus/pci/drivers/igb_uio/new_id
```

5.4.4. Confirm attached driver

```
#confirm that the driver has been attached
lspci -d 19ee: -k
```

5.4.5. Unbind driver

```
#determine card address
PCIA=$(lspci -d 19ee: | awk '{print $1}')

#unbind vfio-pci driver
echo 0000:$PCIA > /sys/bus/pci/drivers/vfio-pci/unbind

#unbind igb_uio driver
echo 0000:$PCIA > /sys/bus/pci/drivers/igb_uio/unbind
```

5.5. DPDK sources with PF PMD support

Current upstreamed DPDK sources do not offer NFP PMD support(Only VFs are supported). The provided DPDK-ns branch should be used instead. The CoreNIC package includes these sources.

5.6. PF PMD Multiport support

The PMD can work with up to 8 ports on the same PF device. The number of available ports is firmware and hardware dependent, and the driver looks for a firmware symbol during initialization to know how many can be used.

DPDK apps work with ports, and a port is usually a PF or a VF PCI device. However, with the NFP PF multiport there is just one PF PCI device. Supporting this particular configuration requires the PMD to create ports in a special way, although once they are created, DPDK apps should be able to use them as normal PCI ports.

NFP ports belonging to same PF can be seen inside PMD initialization with a suffix added to the PCI ID: **wwww:xx:yy.z_port_n**. For example, a PF with PCI ID 0000:03:00.0 and four ports is seen by the PMD code as:

0000:03:00.0_port_0
0000:03:00.0_port_1
0000:03:00.0_port_2
0000:03:00.0_port_3

*Note - There is some limitations with multiport support: RX interrupts and device hotplug are not available. This is only the case when multiple PF ports are configured, and RX interrupts and device hotplug are fully available when using a single PF port.

6. Virtual Functions

6.1. Create Virtual Functions

Four Virtual Functions(VF) will be created using the commands below. The created VFs are allocated across the available physical ports e.g. If you have two ports, every odd VF will be statically linked to the first physical port.

```
#determine card's address
PCIA=0000:$(lspci -d 19ee: | awk '{print $1}' | head -1)

#remove drivers
rmmod vfio-pci
rmmod igb_uio

echo 4 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
setpci -d 19ee:6003 0xc8.L=0xa810 #Send FLR
echo 0 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
echo 4 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
```

6.2. Remove Virtual Functions

```
#determine card's address <-- this assumes that there's only ever going to be one card in the
system.
PCIA=0000:$(lspci -d 19ee: | awk '{print $1}' | head -1)

echo 0 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
```

7. XVIO(Virtiorelay)

7.1. Prerequisites

- Follow the initial CoreNIC install steps as outlined earlier in this document up to chapter 6 “Virtual Functions”.
- Ensure igb_uio driver successfully installed

7.2. Configure hugepages

```
#mount hugepages
mkdir -p /mnt/huge && mount nodev -t hugetlbfs -o rw,pagesize=2M /mnt/huge/
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages

service libvirt-bin restart
mkdir -p /mnt/huge/libvirt
chown libvirt-qemu:kvm -R /mnt/huge/libvirt
```

7.3. Create VFs

```
PCIA=$(lspci -d 19ee: | awk '{print $1}')
echo 4 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
setpci -d 19ee:6003 0xc8.L=0xa810 #Send FLR
echo 0 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
echo 4 > /sys/bus/pci/devices/$PCIA/sriov_numvfs
```

7.4. Attach igb-uio driver

```
#bind igb-uio driver

modprobe uio
DRKO=$(find ~/dpdk-ns -iname 'igb_uio.ko' | head -1 )
insmod $DRKO

#Physical Function
rmmod nfp
echo 19ee 4000 > /sys/bus/pci/drivers/igb_uio/new_id

#Virtual Function
echo 19ee 6003 > /sys/bus/pci/drivers/igb_uio/new_id
```

7.5. Launch XVIO

```
mkdir -p /run/virtiorelayd

chmod 777 /tmp/virtiorelay0.sock

./virtiorelayd -C2,4 -p /run --zmq-port-control-ep=ipc:///run/virtiorelayd/port_control
-zipc:///run/virtiorelayd/stats --loglevel=7  -P0000:05:08.0=0
```

7.6. Add Interface to Guest XML

```
   <interface type='vhostuser'>
     <mac address='aa:2c:a6:27:e2:88'/>
     <source type='unix' path='/tmp/virtiorelay0.sock' mode='client'/>
     <model type='virtio'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x10' function='0x0'/>
   </interface>
```

7.7. Enable shared hugepages in Guest XML

```
 <memoryBacking>
   <hugepages>
     <page size='2048' unit='KiB' nodeset='0'/>
   </hugepages>
 </memoryBacking>

<cpu mode='host-model'>
   <model fallback='allow'/>
   <numa>
     <cell id='0' cpus='0-1' memory='3670016' unit='KiB' memAccess='shared'/>
   </numa>
 </cpu>
```

7.8. Configure MAC Address

Note: The MAC address assigned to the virtio device in the VM must match the MAC address assigned to the VF on the host. You can either configure the virtio device or the host VF.

7.8.1. Set virtio MAC(VM)

```
ip a flush dev ens16
ip l set dev ens16 address ba:2c:a6:27:e2:88 #host VF MAC address
```

OR

7.8.2. Set VF MAC(Host)

Firstly ensure that the dpdk-port is detached from any applications.

```
#unbind the driver from igb_uio (if port is not detached, kernel crashes happen)
  echo "$PCI_ADDR" > /sys/bus/pci/devices/${PCI_ADDR}/driver/unbind

#set the MAC with ip link set
  ip link set dev $PF_NETDEV vf $VF_NO mac $MACADR

#trigger FLR (if PCI device is not unbound, the kernel will crash)
  setpci -s ${PCI_ADDR}  0xc8.L=0xa810

#rebind the driver to igb_uio
  echo "$PCI_ADDR" > /sys/bus/pci/drivers/igb_uio/bind
```

8. Configure NFP Features

Interface parameters may be changed using the ethtool command.

8.1. View interface Parameters

To view the current feature configuration of an interface use the *-k* flag e.g.

```
# ethtool -k nfp_p0

Features for nfp_p0:
rx-checksumming: on
```

```
tx-checksumming: on
        tx-checksum-ipv4: on
        tx-checksum-ip-generic: off [fixed]
        tx-checksum-ipv6: on
        tx-checksum-fcoe-crc: off [fixed]
        tx-checksum-sctp: off [fixed]
scatter-gather: on
        tx-scatter-gather: on
        tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: off
        tx-tcp-segmentation: off
        tx-tcp-ecn-segmentation: off [fixed]
        tx-tcp-mangleid-segmentation: off
        tx-tcp6-segmentation: off
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: on
tx-vlan-offload: on
ntuple-filters: off [fixed]
receive-hashing: on
highdma: on
rx-vlan-filter: on
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: on
tx-gre-csum-segmentation: off [fixed]
tx-ipxip4-segmentation: off [fixed]
tx-ipxip6-segmentation: off [fixed]
tx-udp_tnl-segmentation: on
tx-udp_tnl-csum-segmentation: off [fixed]
tx-gso-partial: off [fixed]
tx-sctp-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
rx-fcs: off [fixed]
rx-all: off [fixed]
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off [fixed]
hw-tc-offload: off [fixed]
```

8.2. Configure Interface Parameters

Features can be enabled or disabled by using the *-K* flag e.g.

```
# Enable tcp-segmentation-offload
ethtool -K nfp_p0 tso on

# Disable tcp-segmentation-offload
ethtool -K nfp_p0 tso off
```

Appendix A: Example VM XML

cat /proc/meminfo:

HugePages_Total: 5100
HugePages_Free: 2924
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB

xmldump VM1:

```
<domain type='kvm' id='14'>
  <name>VM1</name>
  <uuid>dbabbdc3-8669-4ba6-9854-819d1ee4dac4</uuid>
  <memory unit='KiB'>3670016</memory>
  <currentMemory unit='KiB'>3670016</currentMemory>
  <memoryBacking>
    <hugepages>
      <page size='2048' unit='KiB' nodeset='0'/>
    </hugepages>
  </memoryBacking>
  <vcpu placement='static'>4</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='0-31'/>
    <vcpupin vcpu='1' cpuset='0-31'/>
    <vcpupin vcpu='2' cpuset='0-31'/>
    <vcpupin vcpu='3' cpuset='0-31'/>
  </cputune>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-xenial'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
  </features>
  <cpu mode='host-model'>                    - see chapter 5.3
    <model fallback='allow'/>
    <numa>
      <cell id='0' cpus='0-1' memory='3670016' unit='KiB' memAccess='shared'/>
    </numa>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <pm>
```

```
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk enabled='no'/>
</pm>
<devices>
  <emulator>/usr/bin/kvm-spice</emulator>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2'/>
    <source file='/var/lib/libvirt/images/VM1.qcow2'/>
    <backingStore/>
    <target dev='hda' bus='ide'/>
    <alias name='ide0-0-0'/>
    <address type='drive' controller='0' bus='0' target='0' unit='0'/>
  </disk>
  <controller type='usb' index='0' model='ich9-ehci1'>
    <alias name='usb'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x7'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci1'>
    <alias name='usb'/>
    <master startport='0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' multifunction='on'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci2'>
    <alias name='usb'/>
    <master startport='2'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x1'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci3'>
    <alias name='usb'/>
    <master startport='4'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x2'/>
  </controller>
  <controller type='pci' index='0' model='pci-root'>
    <alias name='pci.0'/>
  </controller>
  <controller type='ide' index='0'>
    <alias name='ide'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
  </controller>
  <interface type='network'>
    <mac address='52:54:00:fd:6d:50'/>
    <source network='default' bridge='virbr0'/>
    <target dev='vnet0'/>
    <model type='virtio'/>
    <alias name='net0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
  </interface>
  <interface type='vhostuser'>                    - see chapter 7.6
    <mac address='0e:41:bb:fe:06:a0'/>
    <source type='unix' path='/tmp/virtiorelay0.sock' mode='client'/> - see  Chapter 7.5 and 7.6
    <model type='virtio'/>
    <alias name='net1'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
  </interface>
  <serial type='pty'>
    <source path='/dev/pts/9'/>
```

```
    <target port='0'/>
    <alias name='serial0'/>
   </serial>
   <console type='pty' tty='/dev/pts/9'>
    <source path='/dev/pts/9'/>
    <target type='serial' port='0'/>
    <alias name='serial0'/>
   </console>
   <input type='mouse' bus='ps2'/>
   <input type='keyboard' bus='ps2'/>
   <graphics type='vnc' port='5900' autoport='yes' listen='0.0.0.0'>
    <listen type='address' address='0.0.0.0'/>
   </graphics>
   <video>
    <model type='cirrus' vram='16384' heads='1'/>
    <alias name='video0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
   </video>
   <memballoon model='virtio'>
    <alias name='balloon0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
   </memballoon>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
   <label>libvirt-dbabbdc3-8669-4ba6-9854-819d1ee4dac4</label>
   <imagelabel>libvirt-dbabbdc3-8669-4ba6-9854-819d1ee4dac4</imagelabel>
  </seclabel>
</domain>
```

Appendix B: Fault find Checklist

Please check the following:

1. Installed packages (ns-agilio-core-nic version is **1.1**)

```
dpkg -l | grep core-nic
```

2. Check deb package install log if **device needs to be reflashed**
3. Check that interfaces *nfp_p0* and *nfp_p1* exists before any VF configs
4. Check if **DKMS modules** for the needed kernels built successfully modinfo src/nfp.ko | grep src_ver
5. Check if correct driver assigned to VFs (***lspci -kd 19ee:***)
6. Check if **HugePages** is correctly enabled
7. Some config changes requires a **libvirt service** restart
8. Ensure the **MAC** in the "***vhost user***" tag in the VM XML is the same as the connected VF MAC

Contact us

Netronome Systems, Inc.
2903 Bunker Hill Lane, Suite 150 Santa Clara, CA 95054
Tel: 408.496.0022 | Fax: 408.586.0002
www.netronome.com