# OpenStack: How can SR-IOV IB passthrough be set up for virtual machines with OpenStack Nova?

***How can SR-IOV IB passthrough be set up for virtual machines with OpenStack Nova?***

Allowing OpenStackNetworking and OpenStackCompute to create SR-IOV ports has been possible since the OpenStack Juno release.

This document describes how to:

1. Create a VF (a virtual function, as explained later) on the IB interface.
2. Configure Nova to use the VF for SR-IOV.

To do those steps in Bright, the following are required:

1. A Bright Cluster Manager (BCM) headnode version 7.3 or 8.0
2. At least one compute node for an all-in-one installation.
3. This compute node must have an IB card, and it must support SR-IOV.

What is a PF?:

A physical function. This is the physical ethernet controller that supports SR-IOV

What is a VF?:

A virtual function. This is the virtual PCIe device created from a physical function that allows SR-IOV

What is SR-IOV?:

Single Root - Input/Output virtualization. Lets one physical PCIe device appear like multiple virtual PCIe devices.

# OpenStack: How can SR-IOV IB passthrough be set up for virtual machines with OpenStack Nova?

So, the idea here is to make a physical card use SR-IOV to provide VFs to openstack instances from each PF on the hypervisor. The hypervisors are the nodes of the Bright Cluster that have the physical cards.

The steps that can be followed to carry this out are:

1. **Set up BCM, with OpenStack** (nothing special, just a regular BCM OpenStack installation. All the configuration is done in the next steps)
2. Set up IB Support on the cluster
3. Configure the IB interfaces for the hypervisors**.** This will normally be on the compute node hypervisors. If the head node has an IB card, then that should be configured as well.
4. Configure the OpenStack nova-api and nova-compute services with the proper PCI information so that the VFs work.

So, starting from step 2:

**Set up IB support on the cluster**

NOTE : If the cluster has multiple controller/network nodes, then the IB interfaces are needed only on the hypervisor nodes for all cases.

Example of configuring the network via the network and device modes of cmsh:

(in cmsh)

```
#network

# add ibnet
```

```
 ... configure the network ...

#device use master

#interfaces

#add ib0

#set network ibnet

...configure IP addresses...

#repeat the same for all of the compute nodes that have IB

#commit
```

Next, iommu is to be configured in order to enable SR-IOV. The administrator should be aware that there may also be some BIOS options that may need to be enabled, and should check for these.

(in cmsh)

```
#device use <compute node>

#append kernelparameters " intel_iommu=on"

#commit
```

Reboot the compute nodes. Also reboot the head node if it had an IB interface configured on it.

**Aside: OFED installation**

If the cluster had no OFED installed, then it needs to be installed. The scripts to install the OFED driver are run on the head node. They install OFED on the head node and on the software image. Examples:

*The following installs OFED on the head node:*

```
#yum install mlnx-ofed4\*
```

```
#/cm/local/apps/mlnx-ofed40/current/bin/mlnx-ofed40-install.sh -h
```

*The following installs OFED on the software image default-image:*

```
#/cm/local/apps/mlnx-ofed40/current/bin/mlnx-ofed40-install.sh -s
default-image
```

After OFED installation, the compute nodes and head node should be rebooted again.

After rebooting the compute node, a check to see if iommu is enabled should be done:

```
#ssh computenode
```

```
#dmesg | grep -i iommu
```

If all is well, then something like the following should be seen:

```
# [    0.000000] DMAR: IOMMU enabled
```

Right now SR-IOV for connectx-4 is to be enabled. Each manufacturer has its own special way of enabling SR-IOV. For example, Qlogic (now Intel True Scale) is not enabled in the same way as Mellanox, while connectx-3 is enabled the same way as connectx-4.

# OpenStack: How can SR-IOV IB passthrough be set up for virtual machines with OpenStack Nova?

This document continues with the procedure for connectx-4 for a cluster.

Mellanox describes how to enable SR-IOV on an IB connectx-4 card at:

https://community.mellanox.com/docs/DOC-2386#jive_content_id_I_Enable_SRIOV_on_the_Firmware

Steps :

```
#ssh <compute node>
#mst start
#mst status
```

Example output :

```
MST modules:
------------
    MST PCI module is not loaded
    MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt4115_pciconf0          - PCI configuration cycles access.
                                    domain:bus:dev.fn=0000:03:00.0 addr.reg=88
data.reg=92
                                    Chip revision is: 00
```

The “/dev/mst/mt4115_pciconf0” device is the PCI slot for the connectx-4 card that is to be used:

The interface can be queried with:

```
#mlxconfig -d /dev/mst/mt4115_pciconf0 q
```

Example output :

```
Device #1:
----------

Device type:    ConnectX4
PCI device:     /dev/mst/mt4115_pciconf0

Configurations:                              Next Boot
        NON_PREFETCHABLE_PF_BAR              False(0)
        NUM_OF_VFS                           4
        SRIOV_EN                             True(1)
        PF_LOG_BAR_SIZE                      5
        VF_LOG_BAR_SIZE                      1
        NUM_PF_MSIX                          63
```

```
        NUM_VF_MSIX                         11
        INT_LOG_MAX_PAYLOAD_SIZE            AUTOMATIC(0)
        CQE_COMPRESSION                     BALANCED(0)
        LRO_LOG_TIMEOUT0                    6
        LRO_LOG_TIMEOUT1                    7
        LRO_LOG_TIMEOUT2                    8
        LRO_LOG_TIMEOUT3                    12
        LOG_DCR_HASH_TABLE_SIZE             14
        DCR_LIFO_SIZE                       16384
        ROCE_NEXT_PROTOCOL                  254
        LLDP_NB_DCBX_P1                     False(0)
        LLDP_NB_RX_MODE_P1                  OFF(0)
        LLDP_NB_TX_MODE_P1                  OFF(0)
        CLAMP_TGT_RATE_AFTER_TIME_INC_P1    True(1)
        CLAMP_TGT_RATE_P1                   False(0)
        RPG_TIME_RESET_P1                   300
        RPG_BYTE_RESET_P1                   32767
        RPG_THRESHOLD_P1                    5
        RPG_MAX_RATE_P1                     0
        RPG_AI_RATE_P1                      5
        RPG_HAI_RATE_P1                     50
        RPG_GD_P1                           11
```

```
        RPG_MIN_DEC_FAC_P1                 50
        RPG_MIN_RATE_P1                    1
        RATE_TO_SET_ON_FIRST_CNP_P1        0
        DCE_TCP_G_P1                       4
        DCE_TCP_RTT_P1                     1
        RATE_REDUCE_MONITOR_PERIOD_P1      4
        INITIAL_ALPHA_VALUE_P1             1023
        MIN_TIME_BETWEEN_CNPS_P1           0
        CNP_802P_PRIO_P1                   7
        CNP_DSCP_P1                        46
        LINK_TYPE_P1                       IB(1)
        KEEP_ETH_LINK_UP_P1                True(1)
        KEEP_IB_LINK_UP_P1                 False(0)
        KEEP_LINK_UP_ON_BOOT_P1            True(1)
        KEEP_LINK_UP_ON_STANDBY_P1         False(0)
        ROCE_CC_PRIO_MASK_P1               0
        ROCE_CC_ALGORITHM_P1               ECN(0)
        DCBX_IEEE_P1                       True(1)
        DCBX_CEE_P1                        True(1)
        DCBX_WILLING_P1                    True(1)
        NUM_OF_VL_P1                       4_VLS(3)
        NUM_OF_TC_P1                       8_TCS(0)
```

```
        NUM_OF_PFC_P1                   8

        DUP_MAC_ACTION_P1               LAST_CFG(0)

        PORT_OWNER                      True(1)

        ALLOW_RD_COUNTERS               True(1)

        IP_VER                          IPv4(0)

        BOOT_VLAN                       0

        BOOT_VLAN_EN                    False(0)

        BOOT_OPTION_ROM_EN              True(1)

        BOOT_PKEY                       0

[root@node001 ~]#
```

The `NUM_OF_VFS` and `SRIOV_EN` lines show that SR-IOV is enabled, and that the number of VFs is 4.

If the query output is as shown in the preceding, then there is nothing much to do here, and the node does not need to be rebooted or set. If the query output differs --- for example `NUM_OF_VFS` is 0, and `SRIOV_EN` is False, then the following needs to be run:

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 set SRIOV_EN=1 NUM_OF_VFS=4
```

To determine the maximum number of VFs supported, the release notes for the driver, or user manual, should be consulted.

The following command can also be run to determine this:

```
#cat /sys/class/net/ib0/device/sriov_totalvfs
```

A systemd unit must now be created to start before cmd, to enable the VFs:

A file called `mlx.service` should be created with the following content:

```
[Unit]

Description=Enable VFs on SR-IOV enabled interface

Before=cmd.service


[Service]

type=oneshot

ExecStart= /usr/bin/bash /usr/local/mlx_sriov_configure.sh


[Install]

WantedBy=multi-user.target
```

The file should go under `/lib/systemd/system` in the software image :

The `mlx_sriov_configure.sh` script should be something like this:

```
#!/bin/bash
echo 8 > /sys/class/infiniband/mlx5_0/device/mlx5_num_vfs
sleep 3
for i in $(ls /sys/class/infiniband/mlx5_0/device/sriov);do

RAN="${RANDOM:(-2)}"
RAN2="${RANDOM:(-2)}"
RAN3="${RANDOM:(-2)}"
RAN4="${RANDOM:(-2)}"
PORT=$(echo "$RAN+1" | bc)
echo Follow > /sys/class/infiniband/mlx5_0/device/sriov/$i/policy
echo "11:22:33:44:$RAN4:$RAN3:$RAN2:$RAN" > /sys/class/infiniband/mlx5
_0/device/sriov/$i/node
echo "11:22:33:44:$RAN4:$RAN3:$RAN2:$PORT" > /sys/class/infiniband/mlx
5_0/device/sriov/$i/port
done

sleep 5
for i in $(lspci -nn | grep -i mellan | grep -i virtual | awk '{print
$1}'); do echo "0000:$i" > /sys/bus/pci/drivers/mlx5_core/unbind ; sle
ep 1 ; echo "0000:$i" > /sys/bus/pci/drivers/mlx5_core/bind ; echo "$i
 configured" ; done
```

Please note that due to changes between different drivers this can also be Up.

```
echo Follow > /sys/class/infiniband/mlx5_0/device/sriov/$i/policy
```

This service is supposed to start before cmd, so it should be enabled in the software image:

```
#cd /cm/images/image-name

#chroot .
```

```
#systemctl enable mlx
```

#### Configure the IB interfaces for the hypervisors

Add the following content to opensm.conf to /cm/images/<image-name>/etc/opensm/ :

virt_enabled 2

Also, assign the subnet manager role to each hypervisor that will be used for SR-IOV :

(in cmsh:)

device use <hypervisor>

roles

assign subnetmanager

commit

You will need to reboot your nodes after the previous steps.

#### Configure the OpenStack nova-api and nova-compute services

Nova-api must now be configured with the vendor_id and product_id of the SR-IOV subinterfaces :

The node which has the interfaces in it can be entered via ssh:

```
#ssh node001
```

and the following can be executed:

```
#lspci -nn | grep -i mellan | grep -i virtual
```

Something like this should be seen:

```
05:00.1 Infiniband controller [0207]: Mellanox Technologies MT27700 Family
[ConnectX-4 Virtual Function] [15b3:1014]

05:00.2 Infiniband controller [0207]: Mellanox Technologies MT27700 Family
[ConnectX-4 Virtual Function] [15b3:1014]

05:00.3 Infiniband controller [0207]: Mellanox Technologies MT27700 Family
[ConnectX-4 Virtual Function] [15b3:1014]

05:00.4 Infiniband controller [0207]: Mellanox Technologies MT27700 Family
[ConnectX-4 Virtual Function] [15b3:1014]
```

Here, 15b3 is the vendor_id, and 1014 is the product_id. The “Virtual Function” text means that the SR-IOV VF is being used, and not the PF.

Back at the headnode, the configuration of the nova-api can start:

(in cmsh:)

```
#configurationoverlay

#user openstackcontrollers

#customization

#add /etc/nova/nova.conf

#entries

#add DEFAULT pci_alias ""

#set value "{\"vendor_id\": \"15b3\", \"product_id\":\"1014\",
\"device_type\":\"type-VF\", \"name\":\"ib\"}"
```

Now the hypervisor can be configured:

(in cmsh:)

```
#configurationoverylay

#use openstackhypervisors

#customization

#add /etc/nova/nova.conf

#entries

#add DEFAULT pci_passthrough_whitelist ""

#set value "[{\"vendor_id\": \"15b3\", \"product_id\":\"1014\"}]"
```

A flavor should be configured:

```
#openstack flavor create --ram 2048 --disk 10 --property
"pci_passthrough:alias"="ib:1" ib.large
```

If you are considering using cluster on demand OpenStack you should create your flavor like :

openstack flavor create --ram 4096 --vcpus 4 --property "pci_passthrough:alias"="ib:1" cod.ib

nova-compute can be restarted on the hypervisor :

In cmsh:

```
#device ; use node001 ; services ; restart openstack-nova-*
```

A Centos image can be downloaded and a keypair created:

```
#cd /opt
```

```
#wget https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud-1
705.raw.tar.gz
```

```
#tar xzvf Centps-7*.tar.gz
```

```
#openstack image create --disk-format raw --container-format bare --file
CentOS-7-x86_64-GenericCloud-1705.raw Centos_Image
```

Now the keypair is created:

```
#openstack keypair create bright > bright.pem ; chmod 0600 bright.pem
```

A machine can now be started with ib.large:

```
#openstack server create --nic net-id=f0e9b3ba-c005-4eca-9eaf-a17a5583d717
--flavor eb994ae3-bdfe-4543-b96a-5cfa5c81dc72 --image
601a033f-a683-4284-a044-b3f4bb2a68e7 --key-name bright ib-test-3
```

The server can be accessed via its IP addres to run the following:

```
#cd /opt

#ssh centos@IP -i bright.pem

#sudo -i

#ip a
```

The result would be something like (note the ib0 result):

```
[centos@ib-test-3 ~]$ sudo -i

[root@ib-test-3 ~]# ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1

    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
    inet 127.0.0.1/8 scope host lo

       valid_lft forever preferred_lft forever

    inet6 ::1/128 scope host

       valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP
qlen 1000

    link/ether fa:16:3e:84:7c:05 brd ff:ff:ff:ff:ff:ff

    inet 10.141.152.1/16 brd 10.141.255.255 scope global dynamic eth0

       valid_lft 86222sec preferred_lft 86222sec

    inet6 fe80::f816:3eff:fe84:7c05/64 scope link

       valid_lft forever preferred_lft forever

3: ib0: <BROADCAST,MULTICAST> mtu 4092 qdisc noop state DOWN qlen 256

    link/infiniband
80:00:00:67:fe:80:00:00:00:00:00:00:00:00:00:00:00:00:00:00 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff

[root@ib-test-3 ~]#
```

Now you need to install your OFED stack inside your virtual machine:

yum install mlnx-ofed34 -y

If the machine is a head node then run:

/cm/local/apps/mlnx-ofed34/current/bin/mlnx-ofed34-install.sh -h

# OpenStack: How can SR-IOV IB passthrough be set up for virtual machines with OpenStack Nova?

If the machine is one of your compute nodes and you want to install the OFED stack on your software image then run:

/cm/local/apps/mlnx-ofed34/current/bin/mlnx-ofed34-install.sh -s default-image

Unique solution ID: #1361
Author: Frank Furter
Last update: 2018-07-13 09:12