Single Root IO Virtualization (SR-IOV)

Table of contents

List of Figures

Figure 11. Procedure Heading Icon Version 1 Modificationdate 1717697191472 Api V2

Single Root IO Virtualization (SR-IOV) is a technology that allows a physical PCIe device to present itself multiple times through the PCIe bus. This technology enables multiple virtual instances of the device with separate resources. NVIDIA adapters are capable of exposing up to 127 virtual instances (Virtual Functions (VFs) for each port in the NVIDIA ConnectX® family cards. These virtual functions can then be provisioned separately. Each VF can be seen as an additional device connected to the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of the Physical Function.
SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines direct hardware access to network resources hence increasing its performance.
In this chapter we will demonstrate setup and configuration of SR-IOV in a Red Hat Linux environment using ConnectX® VPI adapter cards.

System Requirements

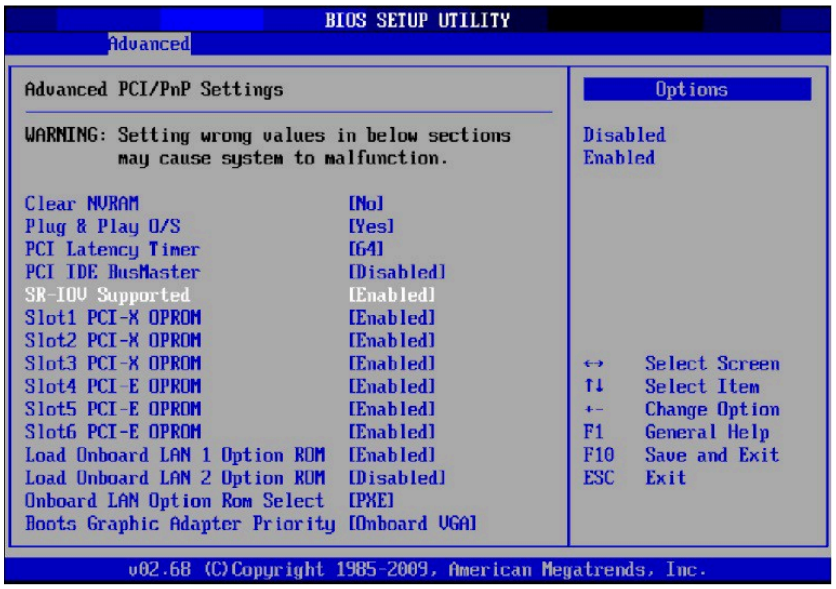
To set up an SR-IOV environment, the following is required:

- MLNX_EN Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6
- NVIDIA ConnectX® VPI Adapter Card family with SR-IOV capability

Setting Up SR-IOV

Depending on your system, perform the steps below to set up your BIOS. The figures used in this section are for illustration purposes only. For further information, please refer to the appropriate BIOS User Manual:

1. Enable "SR-IOV" in the system BIOS.



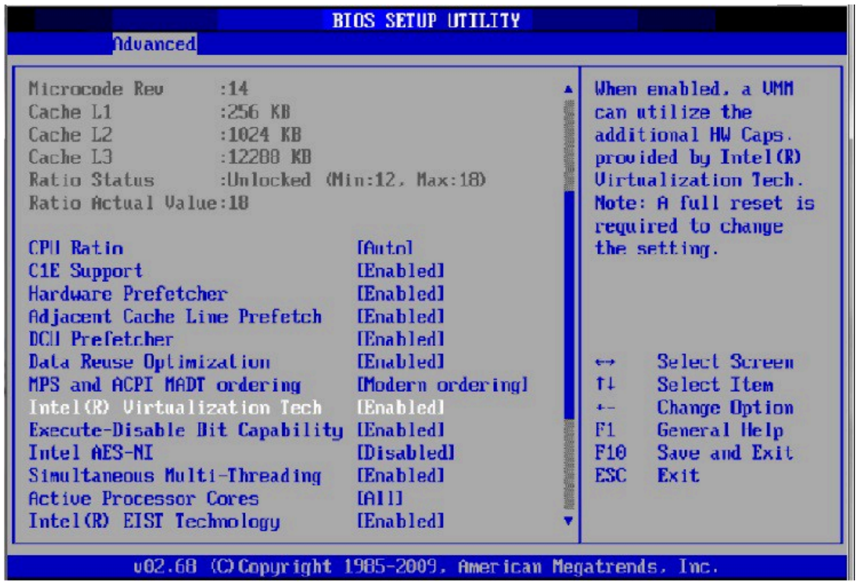
```
                          BIOS SETUP UTILITY
        Advanced
 Advanced PCI/PnP Settings                          Options

 WARNING: Setting wrong values in below sections    Disabled
          may cause system to malfunction.          Enabled

 Clear NVRAM                    [No]
 Plug & Play O/S                [Yes]
 PCI Latency Timer              [64]
 PCI IDE BusMaster              [Disabled]
 SR-IOV Supported               [Enabled]
 Slot1 PCI-X OPROM              [Enabled]
 Slot2 PCI-X OPROM              [Enabled]
 Slot3 PCI-X OPROM              [Enabled]           ↔    Select Screen
 Slot4 PCI-E OPROM              [Enabled]           ↑↓   Select Item
 Slot5 PCI-E OPROM              [Enabled]           +-   Change Option
 Slot6 PCI-E OPROM              [Enabled]           F1   General Help
 Load Onboard LAN 1 Option ROM  [Enabled]           F10  Save and Exit
 Load Onboard LAN 2 Option ROM  [Disabled]          ESC  Exit
 Onboard LAN Option Rom Select  [PXE]
 Boots Graphic Adapter Priority [Onboard VGA]

        v02.68 (C)Copyright 1985-2009, American Megatrends, Inc.
```

2. Enable "Intel Virtualization Technology".



```
                          BIOS SETUP UTILITY
        Advanced
 Microcode Rev       :14                              When enabled, a VMM
 Cache L1            :256 KB                          can utilize the
 Cache L2            :1024 KB                         additional HW Caps.
 Cache L3            :12288 KB                        provided by Intel(R)
 Ratio Status        :Unlocked (Min:12, Max:18)       Virtualization Tech.
 Ratio Actual Value:18                                Note: A full reset is
                                                      required to change
 CPU Ratio                        [Auto]              the setting.
 C1E Support                      [Enabled]
 Hardware Prefetcher              [Enabled]
 Adjacent Cache Line Prefetch     [Enabled]
 DCU Prefetcher                   [Enabled]
 Data Reuse Optimization          [Enabled]           ↔    Select Screen
 MPS and ACPI MADT ordering       [Modern ordering]   ↑↓   Select Item
 Intel(R) Virtualization Tech     [Enabled]           +-   Change Option
 Execute-Disable Bit Capability   [Enabled]           F1   General Help
 Intel AES-NI                     [Disabled]          F10  Save and Exit
 Simultaneous Multi-Threading     [Enabled]           ESC  Exit
 Active Processor Cores           [All]
 Intel(R) EIST Technology         [Enabled]

        v02.68 (C)Copyright 1985-2009, American Megatrends, Inc.
```

3. Install a hypervisor that supports SR-IOV.

4. Depending on your system, update the /boot/grub/grub.conf file to include a similar command line load parameter for the Linux kernel.
   For example, to Intel systems, add:

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
```

```
hiddenmenu
title Red Hat Enterprise Linux Server (4.x.x)
root (hd0,0)
kernel /vmlinuz-4.x.x ro root=/dev/VolGroup00/LogVol00 rhgb quiet
intel_iommu=on initrd /initrd-4.x.x.img
```

Note: Please make sure the parameter "intel_iommu=on" exists when updating the /boot/grub/grub.conf file, otherwise SR-IOV cannot be loaded.

Some OSs use /boot/grub2/grub.cfg file. If your server uses such file, please edit this file instead (add "intel_iommu=on" for the relevant menu entry at the end of the line that starts with "linux16").

Configuring SR-IOV (Ethernet)

To set SR-IOV in Ethernet mode, refer to HowTo Configure SR-IOV for ConnectX-4/ConnectX- 5/ConnectX-6 with KVM (Ethernet) Community Post.

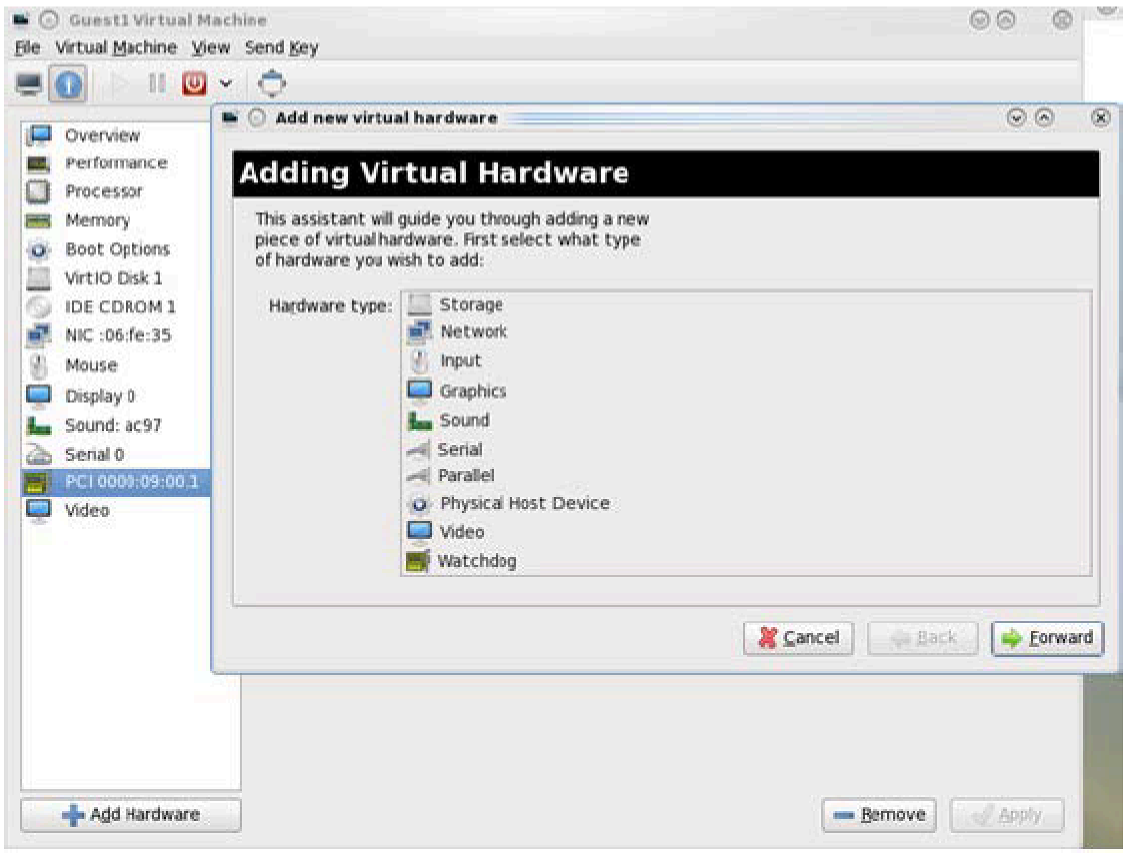
Additional SR-IOV Configurations

Assigning a Virtual Function to a Virtual Machine

This section describes a mechanism for adding a SR-IOV VF to a Virtual Machine.

Assigning the SR-IOV Virtual Function to the Red Hat KVM VM Server

1. Run the virt-manager.
2. Double click on the virtual machine and open its Properties.
3. Go to Details Add hardware PCI host device.



This assistant will guide you through adding a new piece of virtual hardware. First select what type of hardware you wish to add:

4. Choose a NVIDIA virtual function according to its PCI device (e.g., 00:03.1)

5. If the Virtual Machine is up reboot it, otherwise start it.

6. Log into the virtual machine and verify that it recognizes the NVIDIA card. Run:

```
lspci | grep Mellanox
```

Example:

```
lspci | grep Mellanox
01:00.0 Infiniband controller: Mellanox Technologies MT28800 Family
[ConnectX-5 Ex]
```

7. Add the device to the /etc/sysconfig/network-scripts/ifcfg-ethX configuration file. The MAC address for every virtual function is configured randomly, therefore it is not necessary to add it.

Ethernet Virtual Function Configuration when Running SR-IOV

SR-IOV Virtual function configuration can be done through Hypervisor iprout2/netlink tool, if present. Otherwise, it can be done via sysfs.

```
ip link set { dev DEVICE | group DEVGROUP } [ { up | down } ]
...
[ vf NUM [ mac LLADDR ] [ vlan VLANID [ qos VLAN-QOS ] ]
...
[ spoofchk { on | off} ] ]
...

sysfs configuration (ConnectX-4):

/sys/class/net/enp8s0f0/device/sriov/[VF]

+-- [VF]
| +-- config
| +-- link_state
| +-- mac
| +-- mac_list
| +-- max_tx_rate
| +-- min_tx_rate
| +-- spoofcheck
| +-- stats
| +-- trunk
| +-- trust
| +-- vlan
```

VLAN Guest Tagging (VGT) and VLAN Switch Tagging (VST)

When running ETH ports on VGT, the ports may be configured to simply pass through packets as is from VFs (VLAN Guest Tagging), or the administrator may configure the Hypervisor to silently force packets to be associated with a VLAN/Qos (VLAN Switch Tagging).
In the latter case, untagged or priority-tagged outgoing packets from the guest will have the VLAN tag inserted, and incoming packets will have the VLAN tag removed.
The default behavior is VGT.

To configure VF VST mode, run:

```
ip link set dev <PF device> vf <NUM> vlan <vlan_id> [qos <qos>]
```

where:

- NUM = 0..max-vf-num
- vlan_id = 0..4095
- qos = 0..7

For example:

- ip link set dev eth2 vf 2 vlan 10 qos 3 - sets VST mode for VF #2 belonging to PF eth2, with vlan_id = 10 and qos = 3
- ip link set dev eth2 vf 2 vlan 0 - sets mode for VF 2 back to VGT

Additional Ethernet VF Configuration Options

- **Guest MAC configuration** - by default, guest MAC addresses are configured to be all zeroes. If the administrator wishes the guest to always start up with the same MAC, he/she should configure guest MACs before the guest driver comes up. The guest MAC may be configured by using:

  ```
  ip link set dev <PF device> vf <NUM> mac <LLADDR>
  ```

For legacy and ConnectX-4 guests, which do not generate random MACs, the administrator should always configure their MAC addresses via IP link, as above.

- **Spoof checking** - Spoof checking is currently available only on upstream kernels newer than 3.1.

```
ip link set dev <PF device> vf <NUM> spoofchk [on | off]
```

- **Guest Link State**

```
ip link set dev <PF device> vf <UM> state [enable| disable| auto]
```

Virtual Function Statistics

Virtual function statistics can be queried via sysfs:

```
cat /sys/class/infiniband/mlx5_2/device/sriov/2/stats tx_packets : 5011
tx_bytes : 4450870
tx_dropped : 0
rx_packets : 5003
rx_bytes : 4450222
rx_broadcast : 0
rx_multicast : 0
tx_broadcast : 0
tx_multicast : 8
rx_dropped : 0
```

Mapping VFs to Ports

➤ ***To view the VFs mapping to ports:***

Use the ip link tool v2.6.34~3 and above.

```
ip link
```

Output:

```
61: p1p1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode
DEFAULT group default qlen 1000
link/ether 00:02:c9:f1:72:e0 brd ff:ff:ff:ff:ff:ff
vf 0 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
vf 37 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
vf 38 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
vf 39 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
```

When a MAC is ff:ff:ff:ff:ff:ff, the VF is not assigned to the port of the net device it is listed under. In the example above, **vf38** is not assigned to the same port as **p1p1**, in contrast to **vf0**.

However, even VFs that are not assigned to the net device, could be used to set and change its settings. For example, the following is a valid command to change the spoof check:

```
ip link set dev p1p1 vf 38 spoofchk on
```

This command will affect only the **vf38**. The changes can be seen in ip link on the net device that this device is assigned to.

RoCE Support

RoCE is supported on Virtual Functions and VLANs may be used with it. For RoCE, the hypervisor GID table size is of 16 entries while the VFs share the remaining 112 entries. When the number of VFs is larger than 56 entries, some of them will have GID table with only a single entry which is inadequate if VF's Ethernet device is assigned with an IP address.

Virtual Guest Tagging (VGT+)

VGT+ is an advanced mode of Virtual Guest Tagging (VGT), in which a VF is allowed to tag its own packets as in VGT, but is still subject to an administrative VLAN trunk policy. The policy determines which VLAN IDs are allowed to be transmitted or received. The policy does not determine the user priority, which is left unchanged.
Packets can be sent in one of the following modes: when the VF is allowed to send/receive untagged and priority tagged traffic and when it is not. No default VLAN is defined for VGT+ port. The send packets are passed to the eSwitch only if they match the set, and the received packets are forwarded to the VF only if they match the set.

Configuration

Note

When working in SR-IOV, the default operating mode is VGT.

To enable VGT+ mode:

Set the corresponding port/VF (in the example below port eth5, VF0) range of allowed VLANs.

```
echo "<add> <start_vid> <end_vid>" > /sys/class/net/eth5/device/sriov/0/trunk
```

Examples:

- Adding VLAN ID range (4-15) to trunk:

  ```
  echo add 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
  ```

- Adding a single VLAN ID to trunk:

  ```
  echo add 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
  ```

Note: When VLAN ID = 0, it indicates that untagged and priority-tagged traffics are allowed

➢ ***To disable VGT+ mode, make sure to remove all VLANs.***

```
echo rem 0 4095 > /sys/class/net/eth5/device/sriov/0/trunk
```

➢ ***To remove selected VLANs.***

- Remove VLAN ID range (4-15) from trunk:

  ```
  echo rem 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
  ```

- Remove a single VLAN ID from trunk:

  ```
  echo rem 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
  ```

SR-IOV Advanced Security Features

SR-IOV MAC Anti-Spoofing

Normally, MAC addresses are unique identifiers assigned to network interfaces, and they are fixed addresses that cannot be changed. MAC address spoofing is a technique for altering the MAC address to serve different purposes. Some of the cases in which a MAC address is altered can be legal, while others can be illegal and abuse security mechanisms or disguises a possible attacker.
The SR-IOV MAC address anti-spoofing feature, also known as MAC Spoof Check provides protection against malicious VM MAC address forging. If the network administrator assigns a MAC address to a VF (through the hypervisor) and enables spoof check on it, this will limit the end user to send traffic only from the assigned MAC address of that VF.

MAC Anti-Spoofing Configuration

Note

MAC anti-spoofing is disabled by default.

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.
There are two ways to enable or disable MAC anti-spoofing:

1. Use the standard IP link commands - available from Kernel 3.10 and above.
    1. To enable MAC anti-spoofing, run:

       ```
       ip link set ens785f1 vf 0 spoofchk on
       ```

    2. To disable MAC anti-spoofing, run:

       ```
       ip link set ens785f1 vf 0 spoofchk off
       ```

2. Specify echo "ON" or "OFF" to the file located under /sys/class/net/<ifname / device/sriov/<VF index>/spoofcheck.
    1. To enable MAC anti-spoofing, run:

       ```
       echo "ON" > /sys/class/net/ens785f1/vf/0/spoofchk
       ```

    2. To disable MAC anti-spoofing, run:

       ```
       echo "OFF" > /sys/class/net/ens785f1/vf/0/spoofchk
       ```

Note

This configuration is non-persistent and does not survive driver restart.

Limit and Bandwidth Share Per VF

This feature enables rate limiting traffic per VF in SR-IOV mode. For details on how to configure rate limit per VF for ConnectX-4 and above adapter cards, please refer to HowTo Configure Rate Limit per VF for ConnectX-4/ConnectX-5/ConnectX-6 Community post.

Limit Bandwidth per Group of VFs

VFs Rate Limit for vSwitch (OVS) feature allows users to join available VFs into groups and set a rate limitation on each group. Rate limitation on a VF group ensures that the total Tx bandwidth that the VFs in this group get (altogether combined) will not exceed the given value.
With this feature, a VF can still be configured with an individual rate limit as in the past (under /sys/class/net/<ifname>/device/sriov/<vf_num>/max_tx_rate). However, the actual bandwidth limit on the VF will eventually be determined considering the VF group limitation and how many VFs are in the same group.
For example: 2 VFs (0 and 1) are attached to group 3.

Case 1: The rate limitation on the group is set to 20G. Rate limit of each VF is 15G

Result: Each VF will have a rate limit of 10G

Case 2: Group’s max rate limitation is still set to 20G. VF 0 is configured to 30G limit, while VF 1 is configured to 5G rate limit

Result: VF 0 will have 15G de-facto. VF 1 will have 5G

The rule of thumb is that the group's bandwidth is distributed evenly between the number of VFs in the group. If there are leftovers, they will be assigned to VFs whose individual rate limit has not been met yet.

VFs Rate Limit Feature Configuration

1. When VF rate group is supported by FW, the driver will create a new hierarchy in the SRI-OV sysfs named "groups" (/sys/class/net/<ifname>/device/sriov/groups/). It will contain all the info and the configurations allowed for VF groups.

2. All VFs are placed in group 0 by default since it is the only existing group following the initial driver start. It would be the only group available under /sys/class/net/<ifname>/device/sriov/groups/

3. The VF can be moved to a different group by writing to the group file -> echo $GROUP_ID > /sys/class/net/<ifname>/device/sriov/<vf_id>/group

4. The group IDs allowed are 0-255

5. Only when there is at least 1 VF in a group, there will be a group configuration available under /sys/class/net/<ifname>/device/sriov/groups/ (Except for group 0, which is always available even when it's empty).

6. Once the group is created (by moving at least 1 VF to that group), users can configure the group's rate limit. For example:

    1. echo 10000 > /sys/class/net/<ifname>/device/sriov/5/max_tx_rate – setting individual rate limitation of VF 5 to 10G (Optional)

    2. echo 7 > /sys/class/net/<ifname>/device/sriov/5/group – moving VF 5 to group 7

    3. echo 5000 > /sys/class/net/<ifname>/device/sriov/groups/7/max_tx_rate – setting group 7 with rate limitation of 5G

    4. When running traffic via VF 5 now, it will be limited to 5G because of the group rate limit even though the VF itself is limited to 10G

    5. echo 3 > /sys/class/net/<ifname>/device/sriov/5/group – moving VF 5 to group 3

6. Group 7 will now disappear from /sys/class/net/<ifname>/device/sriov/groups since there are 0 VFs in it. Group 3 will now appear. Since there’s no rate limit on group 3, VF 5 can transmit at 10G (thanks to its individual configuration)

Notes:

- You can see to which group the VF belongs to in the ‘stats’ sysfs (cat /sys/class/net/<ifname>/device/sriov/<vf_num>/stats)

- You can see the current rate limit and number of attached VFs to a group in the group’s ‘config’ sysfs (cat /sys/class/net/<ifname>/device/sriov/groups/<group_id>/config)

Bandwidth Guarantee per Group of VFs

Bandwidth guarantee (minimum BW) can be set on a group of VFs to ensure this group is able to transmit at least the amount of bandwidth specified on the wire.

Note the following:

- The minimum BW settings on VF groups determine how the groups share the total BW between themselves. It does not impact an individual VF’s rate settings.

- The total minimum BW that is set on the VF groups should not exceed the total line rate. Otherwise, results are unexpected.

- It is still possible to set minimum BW on the individual VFs inside the group. This will determine how the VFs share the group’s minimum BW between themselves. The total minimum BW of the VF member should not exceed the minimum BW of the group.

For instruction on how to create groups of VFs, see Limit Bandwidth per Group of VFs above.

Example

With a 40Gb link speed, assuming 4 groups and default group 0 have been created:

```
echo 20000 > /sys/class/net/<ifname>/device/sriov/group/1/min_tx_rate
echo 5000 > /sys/class/net/<ifname>/device/sriov/group/2/min_tx_rate
echo 15000 > /sys/class/net/<ifname>/device/sriov/group/3/min_tx_rate
```

```
Group 0(default) : 0 - No BW guarantee is configured.
Group 1 : 20000 - This is the maximum min rate among groups
Group 2 : 5000 which is 25% of the maximum min rate
Group 3 : 15000 which is 75% of the maximum min rate
Group 4 : 0 - No BW guarantee is configured.
```

Assuming there are VFs attempting to transmit in full line rate in all groups, the results would look like: In which case, the minimum BW allocation would be:

```
Group0 – Will have no BW to use since no BW guarantee was set on it while other groups do have such settings.
Group1 – Will transmit at 20Gb/s
Group2 – Will transmit at 5Gb/s
Group3 – Will transmit at 15Gb/s
Group4 - Will have no BW to use since no BW guarantee was set on it while other groups do have such settings.
```

Privileged VFs

In case a malicious driver is running over one of the VFs, and in case that VF's permissions are not restricted, this may open security holes. However, VFs can be marked as trusted and can thus receive an exclusive subset of physical function privileges or permissions. For example, in case of allowing all VFs, rather than specific VFs, to enter a promiscuous mode as a privilege, this will enable malicious users to sniff and monitor the entire physical port for incoming traffic, including traffic targeting other VFs, which is considered a severe security hole.

Privileged VFs Configuration

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.
There are two ways to enable or disable trust:

1. Use the standard IP link commands - available from Kernel 4.5 and above.
    1. To enable trust for a specific VF, run:

        ```
        ip link set ens785f1 vf 0 trust on
        ```

    2. To disable trust for a specific VF, run:

        ```
        ip link set ens785f1 vf 0 trust off
        ```

2. Specify echo "ON" or "OFF" to the file located under /sys/class/net/<ETH_IF_NAME> / device/sriov/<VF index>/trust.

1. To enable trust for a specific VF, run:

    ```
    echo "ON" > /sys/class/net/ens785f1/device/sriov/0/trust
    ```

2. To disable trust for a specific VF, run:

    ```
    echo "OFF" > /sys/class/net/ens785f1/device/sriov/0/trust
    ```

Probed VFs

Probing Virtual Functions (VFs) after SR-IOV is enabled might consume the adapter cards' resources. Therefore, it is recommended not to enable probing of VFs when no monitoring of the VM is needed.
VF probing can be disabled in two ways, depending on the kernel version installed on your server:

1. If the kernel version installed is v4.12 or above, it is recommended to use the PCI sysfs interface `sriov_drivers_autoprobe`. For more information, see linux-next branch .

2. If the kernel version installed is older than v4.12, it is recommended to use the mlx5_core module parameter probe_vf with driver version 4.1 or above.

Example:

```
echo 0 > /sys/module/mlx5_core/parameters/probe_vf
```

For more information on how to probe VFs, see HowTo Configure and Probe VFs on mlx5 DriversCommunity post.

VF Promiscuous Rx Modes

VF Promiscuous Mode

VFs can enter a promiscuous mode that enables receiving the unmatched traffic and all the multicast traffic that reaches the physical port in addition to the traffic originally targeted to the VF. The unmatched traffic is any traffic's DMAC that does not match any of the VFs' or PFs' MAC addresses.

Note: Only privileged/trusted VFs can enter the VF promiscuous mode.

➢ ***To set the promiscuous mode on for a VF, run:***

```
ifconfig eth2 promisc
```

To exit the promiscuous mode, run:

➢

```
ifconfig eth2 –promisc
```

VF All-Multi Mode

VFs can enter an all-multi mode that enables receiving all the multicast traffic sent from/to the other functions on the same physical port in addition to the traffic originally targeted to the VF.

Note: Only privileged/trusted VFs can enter the all-multi RX mode.

To set the all-multi mode on for a VF, run:

➢

```
ifconfig eth2 allmulti
```

To exit the all-multi mode, run:

➢

```
#ifconfig eth2 –allmulti
```

Uninstalling the SR-IOV Driver

To uninstall SR-IOV driver, perform the following:

➢

1. For Hypervisors, detach all the Virtual Functions (VF) from all the Virtual Machines (VM) or stop the Virtual Machines that use the Virtual Functions.

   Please be aware that stopping the driver when there are VMs that use the VFs, will cause machine to hang.

2. Run the script below. Please be aware, uninstalling the driver deletes the entire driver's file, but does not unload the driver.

   ```
   [root@swl022 ~]# /usr/sbin/ofed_uninstall.sh
   This program will uninstall all OFED packages on your machine.
   Do you want to continue?[y/N]:y
   ```

```
Running /usr/sbin/vendor_pre_uninstall.sh
Removing OFED Software installations
Running /bin/rpm -e --allmatches kernel-ib kernel-ib-devel libibverbs
libibverbs-devel libibverbs-devel-static libibverbs-utils libmlx4 libmlx4-devel
libibcm libibcm-devel libibumad libibumad-devel libibumad-static libibmad
libibmad-devel libibmad-static librdmacm librdmacm-utils librdmacm-devel
ibacm opensm-libs opensm-devel perftest compat-dapl compat-dapl-devel
dapl dapl-devel dapl-devel-static dapl-utils srptools infiniband-diags-guest ofed-
scripts opensm-devel
warning: /etc/infiniband/openib.conf saved as
/etc/infiniband/openib.conf.rpmsave
Running /tmp/2818-ofed_vendor_post_uninstall.sh
```

3. Restart the server.