MLNX_EN for Linux User Manual

Rev 2.1-1.0.0

www.mellanox.com

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCTO(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

Table of Contents

List of Tables

Document Revision History

Table 1 - Document Revision History

Release	Date	Description
2.1-1.0.0	January 2014	Added Section 5.5, “Ethernet Performance Counters”, on page 39
2.0-3.0.0	October 2013	• Added the following sections: • Section 5.4, “Single Root IO Virtualization (SR-IOV)”, on page 30 • Section 5.3, “Flow Steering”, on page 28 • Section 5.2, “Time-Stamping Service”, on page 25

About this Manual

This Preface provides general information concerning the scope and organization of this User's Manual.

Intended Audience

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards. It is also intended for application developers.

Common Abbreviations and Acronyms

Table 2 - Abbreviations and Acronyms (Sheet 1 of 2)

Abbreviation / Acronym	Whole Word / Description
B	(Capital) 'B' is used to indicate size in bytes or multiples of bytes (e.g., 1KB = 1024 bytes, and 1MB = 1048576 bytes)
b	(Small) 'b' is used to indicate size in bits or multiples of bits (e.g., 1Kb = 1024 bits)
FW	Firmware
HCA	Host Channel Adapter
HW	Hardware
IB	InfiniBand
iSER	iSCSI RDMA Protocol
LSB	Least significant *byte*
lsb	Least significant *bit*
MSB	Most significant *byte*
msb	Most significant *bit*
NIC	Network Interface Card
SW	Software
VPI	Virtual Protocol Interconnect
IPoIB	IP over InfiniBand
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet

Table 2 - Abbreviations and Acronyms (Sheet 2 of 2)

Abbreviation / Acronym	Whole Word / Description
SDP	Sockets Direct Protocol
SL	Service Level
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
EoIB	Ethernet over Infiniband
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane
vHBA	Virtual SCSI Host Bus adapter
uDAPL	User Direct Access Programming Library

Glossary

The following is a list of concepts and terms related to InfiniBand in general and to Subnet Managers in particular. It is included here for ease of reference, but the main reference remains the *InfiniBand Architecture Specification*.

Table 3 - Glossary (Sheet 1 of 2)

Channel Adapter (CA), Host Channel Adapter (HCA)	An IB device that terminates an IB link and executes transport functions. This may be an HCA (Host CA) or a TCA (Target CA).
HCA Card	A network adapter card based on an InfiniBand channel adapter device.
IB Devices	Integrated circuit implementing InfiniBand compliant communication.
IB Cluster/Fabric/ Subnet	A set of IB devices connected by IB cables.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
Local Identifier (ID)	An address assigned to a port (data sink or source point) by the Subnet Manager, unique within the subnet, used for directing packets within the subnet.
Local Device/Node/ System	The IB Host Channel Adapter (HCA) Card installed on the machine running IBDIAG tools.

Table 3 - Glossary (Sheet 2 of 2)

Local Port	The IB port of the HCA through which IBDIAG tools connect to the IB fabric.
Master Subnet Manager	The Subnet Manager that is authoritative, that has the reference configuration information for the subnet. See Subnet Manager.
Multicast Forwarding Tables	A table that exists in every switch providing the list of ports to forward received multicast packet. The table is organized by MLID.
Network Interface Card (NIC)	A network adapter card that plugs into the PCI Express slot and provides one or more ports to an Ethernet network.
Standby Subnet Manager	A Subnet Manager that is currently quiescent, and not in the role of a Master Subnet Manager, by agency of the master SM. See Subnet Manager.
Subnet Administrator (SA)	An application (normally part of the Subnet Manager) that implements the interface for querying and manipulating subnet management data.
Subnet Manager (SM)	One of several entities involved in the configuration and control of the an IB fabric.
Unicast Linear Forwarding Tables (LFT)	A table that exists in every switch providing the port through which packets should be sent to each LID.
Virtual Protocol Interconnet (VPI)	A Mellanox Technologies technology that allows Mellanox channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE subnet (each subnet connects to one of the adpater ports)

Related Documentation

Table 4 - Reference Documents

Document Name	Description
InfiniBand Architecture Specification, Vol. 1, Release 1.2.1	The InfiniBand Architecture Specification that is provided by IBTA
IEEE Std 802.3ae™-2002 (Amendment to IEEE Std 802.3-2002) Document # PDF: SS94996	Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation

Table 4 - Reference Documents

Document Name	Description
Firmware Release Notes for Mellanox adapter devices	See the Release Notes PDF file relevant to your adapter device under `docs/` folder of installed package.
MFT User's Manual	Mellanox Firmware Tools User's Manual. See under `docs/` folder of installed package.
MFT Release Notes	Release Notes for the Mellanox Firmware Tools. See under `docs/` folder of installed package.

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > InfiniBand/VPI Drivers > Linux SW/Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Overview

This document provides information on the MLNX_EN Linux driver and instructions for installing the driver on Mellanox ConnectX adapter cards supporting 10Gb/s and 40Gb/s Ethernet.

The MLNX_EN driver release exposes the following capabilities:

- Single/Dual port
- Up to 16 Rx queues per port
- 16 Tx queues per port
- Rx steering mode: Receive Core Affinity (RCA)
- MSI-X or INTx
- Adaptive interrupt moderation
- HW Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- Multi-core NAPI support
- VLAN Tx/Rx acceleration (HW VLAN stripping/insertion)
- Ethtool support
- Net device statistics
- SR-IOV support
- Flow steering
- Ethernet Time Stamping (at beta level)

1.1 Package Contents

This driver kit contains the following:

Table 5 - MLNX_EN Package Content

Components	Description
mlx4 driver	mlx4 is the low level driver implementation for the ConnectX adapters designed by Mellanox Technologies. The ConnectX can operate as an InfiniBand adapter and as an Ethernet NIC. To accommodate the two flavors, the driver is split into modules: mlx4_core, mlx4_en, and mlx4_ib. **Note:** mlx4_ib is not part of this package.
mlx4_core	Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the InfiniBand, Ethernet and FC functions can share a device without interfering with each other.
mlx4_en	Handles Ethernet specific functions and plugs into the netdev mid-layer.
mstflint	An application to burn a firmware binary image.
Software modules	Sources of all software modules (under conditions mentioned in the modules' LICENSE files)

Table 5 - MLNX_EN Package Content

Components	Description
Documentation	Release Notes, README

2 Driver Installation

2.1 Software Dependencies

- To install the driver software, kernel sources must be installed on the machine.
- MLNX_EN driver cannot coexist with OFED software on the same machine. Hence when installing MLNX_EN all OFED packages should be removed (done by the mlnx_en install script).

2.2 Installing the Driver

Step 1. Download Driver Package from the Mellanox site.

http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=27&menu_section=35

Step 2. Install Driver.

```
#> tar xzvf mlnx_en-2.0-3.0.0.tgz file
#> cd mlnx_en-2.0-3.0.0
#> ./install.sh
```

➢ ***To install mlnx-en-2.0-3.0.0 on XenServer6.1:***

```
# rpm -ihv RPMS/xenserver6u1/i386/`uname -r`/mlnx_en*rpm
```

The package consists of several source RPMs. The install script rebuilds the source RPMs and then installs the created binary RPMs. The created kernel module binaries are located at:

- For KMP RPMs installation:
  - On SLES (mellanox-mlnx-en-kmp RPM):

    ```
    /lib/modules/<kernel-ver>/updates/mellanox-mlnx-en
    ```

  - On RHEL (kmod-mellanox-mlnx-en RPM):

    ```
    /lib/modules/<kernel-ver>/extra/mellanox-mlnx-en
    ```

- For non-KMP RPMs (mlnx_en RPM):
  - On SLES:

    ```
    /lib/modules/<kernel-ver>/updates/mlnx_en
    ```

  - On RHE:

    ```
    /lib/modules/<kernel-ver>/extra/mlnx_en
    ```

mlnx_en installer supports 2 modes of installation. The install scripts selects the mode of driver installation depending of the running OS/kernel version.

- Kernel Module Packaging (KMP) mode, where the source rpm is rebuilt for each installed flavor of the kernel. This mode is used for RedHat and SUSE distributions.
- Non KMP installation mode, where the sources are rebuilt with the running kernel. This mode is used for vanilla kernels.

If the Vanilla kernel is installed as rpm, please use the `"--disable-kmp"` flag when installing the driver.

The kernel module sources are placed under `/usr/src/mellanox-mlnx-en-2.0/`.

➢ ***To recompile the driver:***

```
#> cd /usr/src/mellanox-mlnx-en-2.0/
#> scripts/mlnx_en_patch.sh
#> make
#> make install
```

The uninstall and performance tuning scripts are installed.

If the driver was installed without kmp support, the sources would be located under `/usr/srs/mlnx_en-2.0/`.

2.3 Loading the Driver

Step 1. Make sure no previous driver version is currently loaded.

```
#> modprobe -r mlx4_en
```

Step 2. Load the new driver version.

```
#> modprobe mlx4_en
```

The result is a new net-device appearing in the 'ifconfig -a' output. For details on driver usage and configuration, please refer to Section 3, “Ethernet Driver Usage and Configuration”, on page 16.

On Ubuntu OS, the "mlnx-en" service is responsible for loading the mlx4_en driver upon boot.

2.4 Unloading the Driver

➢ ***To unload the Ethernet driver:***

```
#> modprobe -r mlx4_en
```

2.5 Uninstalling the Driver

➢ ***To uninstall the mlnx_en driver:***

```
#> /sbin/mlnx_en_uninstall.sh
```

3 Ethernet Driver Usage and Configuration

- ***To assign an IP address to the interface:***

```
#> ifconfig eth<x[a]> <ip>
```

a. 'x' is the OS assigned interface number

- ***To check driver and device information:***

```
#> ethtool -i eth<x>
```

Example:

```
#> ethtool -i eth2
driver: mlx4_en
version: 2.1.8 (Oct 06 2013)
firmware-version: 2.30.3110
bus-info: 0000:1a:00.0
```

- ***To query stateless offload status:***

```
#> ethtool -k eth<x>
```

- ***To set stateless offload status:***

```
#> ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off] [lro on|off]
```

- ***To query interrupt coalescing settings:***

```
#> ethtool -c eth<x>
```

- ***To enable/disable adaptive interrupt moderation:***

```
#>ethtool -C eth<x> adaptive-rx on|off
```

By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern.

- ***To set the values for packet rate limits and for moderation time high and low:***

```
#> ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high
N]
```

Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. Below a lower limit of packet rate, the moderation time will be set to its lowest value.

- ***To set interrupt coalescing settings when adaptive moderation is disabled:***

```
#> ethtool -C eth<x> [rx-usecs N] [rx-frames N]
```

usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.

- ***To query pause frame settings:***

```
#> ethtool -a eth<x>
```

- ***To set pause frame settings:***

```
#> ethtool -A eth<x> [rx on|off] [tx on|off]
```

- ***To query ring size values:***

```
#> ethtool -g eth<x>
```

- ***To modify rings size:***

```
#> ethtool -G eth<x> [rx <N>] [tx <N>]
```

- ***To obtain additional device statistics:***

```
#> ethtool -S eth<x>
```

- ***To perform a self diagnostics test:***

```
#> ethtool -t eth<x>
```

The driver defaults to the following parameters:

- Both ports are activated (i.e., a net device is created for each port)
- The number of Rx rings for each port is the nearest power of 2 of number of cpu cores, limited by 16.
- LRO is enabled with 32 concurrent sessions per Rx ring

Some of these values can be changed using module parameters, which can be displayed by running:

```
#> modinfo mlx4_en
```

To set non-default values to module parameters, add to the `/etc/modprobe.conf` file:

```
"options mlx4_en <param_name>=<value> <param_name>=<value> ..."
```

Values of all parameters can be observed in /sys/module/mlx4_en/parameters/.

4 Firmware Programming

The adapter card was shipped with the most current firmware available. This section is intended for future firmware upgrades, and provides instructions for (1) installing Mellanox firmware update tools (MFT), (2) downloading FW, and (3) updating adapter card firmware.

4.1 Installing Firmware Tools

The driver package compiles and installs the Mellanox 'mstflint' utility under /usr/local/bin/. You may also use this tool to burn a card-specific firmware binary image. See the file /tmp/mlnx_en/src/utils/mstflint/README file for details.

Alternatively, you can download the current Mellanox Firmware Tools package (MFT) from www.mellanox.com > Products > Adapter IB/VPI SW > Firmware Tools. The tools package to download is "MFT_SW for Linux" (tarball name is mft-X.X.X.tgz).

For help in identifying your adapter card, please visit

http://www.mellanox.com/content/pages.php?pg=firmware_HCA_FW_identification.

4.2 Updating Adapter Card Firmware

Using a card specific binary firmware image file, enter the following command:

```
#> mstflint -d <pci device> -i <image_name.bin> b
```

For burning firmware using the MFT package, please check the MFT user's manual under /www.mellanox.com > Products > Adapter IB/VPI SW > Firmware Tools.

After burning new firmware to an adapter card, reboot the machine so that the new firmware can take effect.

5 Driver Features

5.1 Quality of Service

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow (socket, rdma_cm connection) and manage its guarantees, limitations and its priority over other flows. This is accomplished by mapping the user's priority to a hardware TC (traffic class) through a 2/3 stages process. The TC is assigned with the QoS attributes and the different flows behave accordingly

5.1.1 Mapping Traffic to Traffic Classes

Mapping traffic to TCs consists of several actions which are user controllable, some controlled by the application itself and others by the system/network administrators.

The following is the general mapping traffic to Traffic Classes flow:

1. The application sets the required Type of Service (ToS).
2. The ToS is translated into a Socket Priority (`sk_prio`).
3. The `sk_prio` is mapped to a User Priority (UP) by the system administrator (some applications set `sk_prio` directly).
4. The UP is mapped to TC by the network/system administrator.
5. TCs hold the actual QoS parameters

QoS can be applied on the following types of traffic. However, the general QoS flow may vary among them:

- **Plain Ethernet** - Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver
- **RoCE** - Applications use the RDMA API to transmit using QPs
- **Raw Ethernet QP** - Application use VERBs API to transmit using a Raw Ethernet QP

5.1.2 Plain Ethernet Quality of Service Mapping

Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver.

The following is the Plain Ethernet QoS mapping flow:

1. The application sets the ToS of the socket using `setsockopt` (`IP_TOS`, value).
2. ToS is translated into the `sk_prio` using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the UP:
   - If the underlying device is a VLAN device, `egress_map` is used controlled by the `vconfig` command. This is per VLAN mapping.
   - If the underlying device is not a VLAN device, the `tc` command is used. In this case, even though `tc` manual states that the mapping is from the `sk_prio` to the TC number, the `mlx4_en` driver interprets this as a `sk_prio` to UP mapping.

Mapping the sk_prio to the UP is done by using `tc_wrap.py -i <dev name> -u 0,1,2,3,4,5,6,7`

4. The the UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used.

Socket applications can use `setsockopt` (`SK_PRIO`, value) to directly set the `sk_prio` of the socket. In this case the ToS to `sk_prio` fixed mapping is not needed. This allows the application and the administrator to utilize more than the 4 values possible via ToS.

In case of VLAN interface, the UP obtained according to the above mapping is also used in the VLAN tag of the traffic

5.1.3 Map Priorities with tc_wrap.py/mlnx_qos

Network flow that can be managed by QoS attributes is described by a User Priority (UP). A user's `sk_prio` is mapped to UP which in turn is mapped into TC.

- Indicating the UP
  - When the user uses `sk_prio`, it is mapped into a UP by the `'tc'` tool. This is done by the `tc_wrap.py` tool which gets a list of <= 16 comma separated UP and maps the `sk_prio` to the specified UP.
    For example, `tc_wrap.py -ieth0 -u 1,5` maps `sk_prio 0` of `eth0` device to UP 1 and `sk_prio 1` to UP 5.
  - Setting `set_egress_map` in VLAN, maps the `skb_priority` of the VLAN to a `vlan_qos`. The `vlan_qos` is represents a UP for the VLAN device.
  - In RoCE, `rdma_set_option` with `RDMA_OPTION_ID_TOS` could be used to set the UP
  - When creating QPs, the `sl` field in `ibv_modify_qp` command represents the UP
- Indicating the TC
  - After mapping the `skb_priority` to UP, one should map the UP into a TC. This assigns the user priority to a specific hardware traffic class. In order to do that, `mlnx_qos` should be used. `mlnx_qos` gets a list of a mapping between UPs to TCs. For example, `mlnx_qos -ieth0 -p 0,0,0,0,1,1,1,1` maps UPs 0-3 to `TC0`, and Ups 4-7 to `TC1`.

5.1.4 Quality of Service Properties

The different QoS properties that can be assigned to a TC are:

- Strict Priority (see "Strict Priority")
- Minimal Bandwidth Guarantee (ETS) (see "Minimal Bandwidth Guarantee (ETS)")
- Rate Limit (see "Rate Limit")

5.1.4.1 Strict Priority

When setting a TC's transmission algorithm to be 'strict', then this TC has absolute (strict) priority over other TC strict priorities coming before it (as determined by the TC number: TC 7 is highest priority, TC 0 is lowest). It also has an absolute priority over non strict TCs (ETS).

This property needs to be used with care, as it may easily cause starvation of other TCs.

A higher strict priority TC is always given the first chance to transmit. Only if the highest strict priority TC has nothing more to transmit, will the next highest TC be considered.

Non strict priority TCs will be considered last to transmit.

This property is extremely useful for low latency low bandwidth traffic. Traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

5.1.4.2 Minimal Bandwidth Guarantee (ETS)

After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.

If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio.

Since this is a minimal guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.

5.1.4.3 Rate Limit

Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

5.1.5 Quality of Service Tools

5.1.5.1 mlnx_qos

`mlnx_qos` is a centralized tool used to configure QoS features of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The `mlnx_qos` tool enables the administrator of the system to:

- Inspect the current QoS mappings and configuration

  The tool will also display maps configured by TC and `vconfig set_egress_map` tools, in order to give a centralized view of all QoS mappings.
- Set UP to TC mapping
- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
- Set rate limit to TCs

For unlimited ratelimit set the ratelimit to 0.

Usage:

```
mlnx_qos -i <interface> [options]
```

Options:

```
--version             show program's version number and exit
-h, --help            show this help message and exit
-p LIST, --prio_tc=LIST
                      maps UPs to TCs. LIST is 8 comma seperated TC numbers.
                      Example: 0,0,0,0,1,1,1,1 maps UPs 0-3 to TC0, and UPs
                      4-7 to TC1
-s LIST, --tsa=LIST   Transmission algorithm for each TC. LIST is comma
                      seperated algorithm names for each TC. Possible
                      algorithms: strict, etc. Example: ets,strict,ets sets
                      TC0,TC2 to ETS and TC1 to strict. The rest are
                      unchanged.
-t LIST, --tcbw=LIST  Set minimal guaranteed %BW for ETS TCs. LIST is comma
                      seperated percents for each TC. Values set to TCs that
                      are not configured to ETS algorithm are ignored, but
                      must be present. Example: if TC0,TC2 are set to ETS,
                      then 10,0,90 will set TC0 to 10% and TC2 to 90%.
                      Percents must sum to 100.
-r LIST, --ratelimit=LIST
                      Rate limit for TCs (in Gbps). LIST is a comma
                      seperated Gbps limit for each TC. Example: 1,8,8 will
                      limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each.
-i INTF, --interface=INTF
                      Interface name
-a                    Show all interface's TCs
```

Get Current Configuration:

```
tc: 0 ratelimit: unlimited, tsa: strict
         up:  0
                  skprio: 0
                  skprio: 1
                  skprio: 2 (tos: 8)
                  skprio: 3
                  skprio: 4 (tos: 24)
                  skprio: 5
                  skprio: 6 (tos: 16)
                  skprio: 7
                  skprio: 8
                  skprio: 9
                  skprio: 10
                  skprio: 11
                  skprio: 12
                  skprio: 13
                  skprio: 14
                  skprio: 15
         up:  1
         up:  2
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
```

Set ratelimit. 3Gbps for tc0 4Gbps for tc1 and 2Gbps for tc2:

```
tc: 0 ratelimit: 3 Gbps, tsa: strict
         up:  0
                  skprio: 0
                  skprio: 1
                  skprio: 2 (tos: 8)
                  skprio: 3
                  skprio: 4 (tos: 24)
                  skprio: 5
                  skprio: 6 (tos: 16)
                  skprio: 7
                  skprio: 8
                  skprio: 9
                  skprio: 10
                  skprio: 11
                  skprio: 12
                  skprio: 13
                  skprio: 14
                  skprio: 15
         up:  1
         up:  2
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
```

Configure QoS. map UP 0,7 to tc0, 1,2,3 to tc1 and 4,5,6 to tc 2. set tc0,tc1 as ets and tc2 as strict. divide ets 30% for tc0 and 70% for tc1:

```
mlnx_qos -i eth3 -s ets,ets,strict -p 0,1,1,1,2,2,2 -t 30,70
tc: 0 ratelimit: 3 Gbps, tsa: ets, bw: 30%
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
         up:  7
tc: 1 ratelimit: 4 Gbps, tsa: ets, bw: 70%
         up:  1
         up:  2
         up:  3
tc: 2 ratelimit: 2 Gbps, tsa: strict
         up:  4
         up:  5
         up:  6
```

5.1.5.2 tc and tc_wrap.py

The 'tc' tool is used to setup `sk_prio` to UP mapping, using the `mqprio` queue discipline.

In kernels that do not support `mqprio` (such as 2.6.34), an alternate mapping is created in `sysfs`. The 'tc_wrap.py' tool will use either the `sysfs` or the 'tc' tool to configure the `sk_prio` to UP mapping.

Usage:

```
tc_wrap.py -i <interface> [options]
```

Options:

```
--version                               show program's version number and exit
-h, --help                              show this help message and exit
-u SKPRIO_UP, --skprio_up=SKPRIO_UP  maps sk_prio to UP. LIST is <=16 comma separated
                                     UP. index of element is sk_prio.
-i INTF, --interface=INTF            Interface name
```

Example: set skprio 0-2 to UP0, and skprio 3-7 to UP1 on eth4

```
UP  0
        skprio: 0
        skprio: 1
        skprio: 2 (tos: 8)
        skprio: 7
        skprio: 8
        skprio: 9
        skprio: 10
        skprio: 11
        skprio: 12
        skprio: 13
        skprio: 14
        skprio: 15
UP  1
        skprio: 3
        skprio: 4 (tos: 24)
        skprio: 5
        skprio: 6 (tos: 16)
UP  2
UP  3
UP  4
UP  5
UP  6
UP  7
```

5.1.5.3 Additional Tools

tc tool compiled with the `sch_mqprio` module is required to support kernel v2.6.32 or higher. This is a part of `iproute2` package v2.6.32-19 or higher. Otherwise, an alternative custom sysfs interface is available.

- `mlnx_qos tool` (package: ofed-scripts) requires python >= 2.5
- `tc_wrap.py` (package: ofed-scripts) requires python >= 2.5

5.2 Time-Stamping Service

Time Stamping is currently at beta level.
Please be aware that everything listed here is subject to change.

Time Stamping is currently supported in ConnectX®-3/ConnectX®-3 Pro adapter cards only.

Time stamping is the process of keeping track of the creation of a packet/ A time-stamping service supports assertions of proof that a datum existed before a particular time. Incoming packets are time-stamped before they are distributed on the PCI depending on the congestion in the PCI buffers. Outgoing packets are time-stamped very close to placing them on the wire.

5.2.1 Enabling Time Stamping

Time-stamping is off by default and should be enabled before use.

➢ ***To enable time stamping for a socket:***

- Call setsockopt() with SO_TIMESTAMPING and with the following flags:

```
SOF_TIMESTAMPING_TX_HARDWARE:  try to obtain send time stamp in hardware
SOF_TIMESTAMPING_TX_SOFTWARE:  if SOF_TIMESTAMPING_TX_HARDWARE is off or
                               fails, then do it in software
SOF_TIMESTAMPING_RX_HARDWARE:  return the original, unmodified time stamp
                               as generated by the hardware
SOF_TIMESTAMPING_RX_SOFTWARE:  if SOF_TIMESTAMPING_RX_HARDWARE is off or
                               fails, then do it in software
SOF_TIMESTAMPING_RAW_HARDWARE: return original raw hardware time stamp
SOF_TIMESTAMPING_SYS_HARDWARE: return hardware time stamp transformed to
                               the system time base
SOF_TIMESTAMPING_SOFTWARE:     return system time stamp generated in
                               software
SOF_TIMESTAMPING_TX/RX determine how time stamps are generated.
SOF_TIMESTAMPING_RAW/SYS determine how they are reported
```

➢ ***To enable time stamping for a net device:***

Admin privileged user can enable/disable time stamping through calling ioctl(sock, SIOCSHWT-STAMP, &ifreq) with following values:

Send side time sampling:

- Enabled by ifreq.hwtstamp_config.tx_type when

```
/* possible values for hwtstamp_config->tx_type */
enum hwtstamp_tx_types {
        /*
         * No outgoing packet will need hardware time stamping;
         * should a packet arrive which asks for it, no hardware
         * time stamping will be done.
        */
        HWTSTAMP_TX_OFF,

        /*
         * Enables hardware time stamping for outgoing packets;
         * the sender of the packet decides which are to be
         * time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
         * before sending the packet.
         */
        HWTSTAMP_TX_ON,
  /*
         * Enables time stamping for outgoing packets just as
         * HWTSTAMP_TX_ON does, but also enables time stamp insertion
         * directly into Sync packets. In this case, transmitted Sync
         * packets will not received a time stamp via the socket error
         * queue.
         */
        HWTSTAMP_TX_ONESTEP_SYNC,
};
Note: for send side time stamping currently only HWTSTAMP_TX_OFF and
HWTSTAMP_TX_ON are supported.
```

Receive side time sampling:

- Enabled by ifreq.hwtstamp_config.rx_filter when

```
/* possible values for hwtstamp_config->rx_filter */
enum hwtstamp_rx_filters {
        /* time stamp no incoming packet at all */
        HWTSTAMP_FILTER_NONE,

        /* time stamp any incoming packet */
        HWTSTAMP_FILTER_ALL,
      /* return value: time stamp all packets requested plus some others */
        HWTSTAMP_FILTER_SOME,

        /* PTP v1, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
        /* PTP v1, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
        /* PTP v1, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
        /* PTP v2, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
        /* PTP v2, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
        /* PTP v2, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,

        /* 802.AS1, Ethernet, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
        /* 802.AS1, Ethernet, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
        /* 802.AS1, Ethernet, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,

        /* PTP v2/802.AS1, any layer, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_EVENT,
        /* PTP v2/802.AS1, any layer, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_SYNC,
        /* PTP v2/802.AS1, any layer, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
};
Note: for receive side time stamping currently only HWTSTAMP_FILTER_NONE and
HWTSTAMP_FILTER_ALL are supported.
```

5.2.2 Getting Time Stamping

Once time stamping is enabled time stamp is placed in the socket Ancillary data. recvmsg() can be used to get this control message for regular incoming packets. For send time stamps the outgoing packet is looped back to the socket's error queue with the send time stamp(s) attached. It can be received with recvmsg(flags=MSG_ERRQUEUE). The call returns the original outgoing packet data including all headers preprended down to and including the link layer, the scm_timestamping control message and a sock_extended_err control message with ee_errno==ENOMSG and ee_origin==SO_EE_ORIGIN_TIMESTAMPING. A socket with such a pending bounced packet is ready for reading as far as select() is concerned. If the outgoing

packet has to be fragmented, then only the first fragment is time stamped and returned to the sending socket.

When time-stamping is enabled, VLAN stripping is disabled. For more info please refer to Documentation/networking/timestamping.txt in kernel.org

5.3 Flow Steering

Flow Steering is applicable to the mlx4 driver only.

Flow steering is a new model which steers network flows based on flow specifications to specific QPs. Those flows can be either unicast or multicast network flows. In order to maintain flexibility, domains and priorities are used. Flow steering uses a methodology of flow attribute, which is a combination of L2-L4 flow specifications, a destination QP and a priority. Flow steering rules could be inserted either by using ethtool or by using InfiniBand verbs. The verbs abstraction uses an opposed terminology of a flow attribute (ibv_flow_attr), defined by a combination of specifications (struct ibv_flow_spec_*).

5.3.1 Enable/Disable Flow Steering

Flow Steering is disabled by default and regular L2 steering is performed instead (B0 Steering). When using SR-IOV, flow steering is enabled if there is adequate amount of space to store the flow steering table for the guest/master.

- ***To enable Flow Steering:***

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.

Step 2. Set the parameter `log_num_mgm_entry_size` to `-1` by writing the option `mlx4_core log_num_mgm_entry_size=-1`.

Step 3. Restart the driver

- ***To disable Flow Steering:***

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.

Step 2. Remove the `options mlx4_core log_num_mgm_entry_size= -1`.

Step 3. Restart the driver

5.3.2 Flow Domains and Priorities

Flow steering defines the concept of domain and priority. Each domain represents a user agent that can attach a flow. The domains are prioritized. A higher priority domain will always supersede a lower priority domain when their flow specifications overlap. Setting a lower priority value will result in higher priority.

In addition to the domain, there is priority within each of the domains. Each domain can have at most 2^12 priorities in accordance to its needs.

The following are the domains at a descending order of priority:

- **Ethtool**

  Ethtool domain is used to attach an RX ring, specifically its QP to a specified flow.

  Please refer to the most recent ethtool manpage for all the ways to specify a flow.

  Examples:

  - ethtool –U eth5 flow-type ether dst 00:11:22:33:44:55 loc 5 action 2

    All packets that contain the above destination MAC address are to be steered into rx-ring 2 (its underlying QP), with priority 5 (within the ethtool domain)

  - ethtool –U eth5 flow-type tcp4 src-ip 1.2.3.4 dst-port 8888 loc 5 action 2

    All packets that contain the above destination IP address and source port are to be steered into rx-ring 2. When destination MAC is not given, the user's destination MAC is filled automatically.

  - ethtool –u eth5

    Shows all of ethtool's steering rule

  When configuring two rules with the same priority, the second rule will overwrite the first one, so this ethtool interface is effectively a table. Inserting Flow Steering rules in the kernel requires support from both the ethtool in the user space and in kernel (v2.6.28).

  MLX4 Driver Support

  The mlx4 driver supports only a subset of the flow specification the ethtool API defines. Asking for an unsupported flow specification will result with an "invalid value" failure.

  The following are the flow specific parameters:

Table 6 - Flow Specific Parameters

	ether	tcp4/udp4	ip4
Mandatory	dst		src-ip/dst-ip
Optional	vlan	src-ip, dst-ip, src-port, dst-port, vlan	src-ip, dst-ip, vlan

- **RFS**

  RFS is an in-kernel-logic responsible for load balancing between CPUs by attaching flows to CPUs that are used by flow's owner applications. This domain allows the RFS mechanism to use the flow steering infrastructure to support the RFS logic by implementing the `ndo_rx_flow_steer`, which, in turn, calls the underlying flow steering mechanism with the RFS domain.

  Enabling the RFS requires enabling the `'ntuple'` flag via the ethtool,

  For example, to enable ntuple for eth0, run:

```
ethtool -K eth0 ntuple on
```

  RFS requires the kernel to be compiled with the `CONFIG_RFS_ACCEL` option. This options is available in kernels 2.6.39 and above. Furthermore, RFS requires Device Managed Flow Steering support.

RFS cannot function if LRO is enabled. LRO can be disabled via ethtool.

- **All of the rest**

  The lowest priority domain serves the following users:

  - **The mlx4 Ethernet driver** attaches its unicast and multicast MACs addresses to its QP using L2 flow specifications

Fragmented UDP traffic cannot be steered. It is treated as 'other' protocol by hardware (from the first packet) and not considered as UDP traffic.

5.4 Single Root IO Virtualization (SR-IOV)

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. Mellanox adapters are capable of exposing in ConnectX®-3 adapter cards 63 virtual instances called Virtual Functions (VFs). These virtual functions can then be provisioned separately. Each VF can be seen as an addition 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 Mellanox ConnectX® VPI adapter cards family.

5.4.1 System Requirements

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

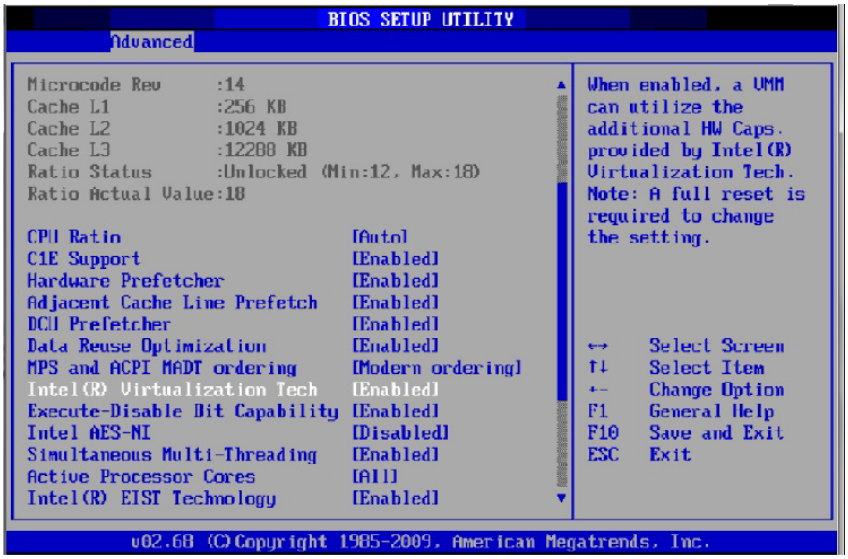
- MLNX_EN Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6.*
- Mellanox ConnectX® VPI Adapter Card family with SR-IOV capability

5.4.2 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:

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

Step 2. Enable "Intel Virtualization Technology".



Step 3. Install the hypervisor that supports SR-IOV.

Step 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 (2.6.32-36.x86-645)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-36.x86-64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        intel_iommu=on[a]
        initrd /initrd-2.6.32-36.x86-64.img
```

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

Step 5. Install the MLNX_EN driver for Linux that supports SR-IOV.

Step 6. Verify the HCA is configured to support SR-IOV.

```
[root@selene ~]# mstflint -dev <PCI Device> dc
```

- Verify in the [HCA] section the following field appears[1],[2]:

```
[HCA]
num_pfs = 1
total_vfs = 5
sriov_en = true
```

HCA parameters can be configured during firmware update using the `mlnxofedinstall` script and running the `'--enable-sriov'` and `'--total-vfs <0-63>'` installation parameters.

If the current firmware version is the same as one provided with MLNX_EN, run it in combination with the `'--force-fw-update'` parameter.

This configuration option is supported only in HCAs that their configuration file (INI) is included in MLNX_EN.

Parameter	Recommended Value
num_pfs	1 **Note:** This field is optional and might not always appear.
total_vfs	63
sriov_en	true

- If the HCA does not support SR-IOV, please contact Mellanox Support: support@mellanox.com

Step 7. Create the text file /etc/modprobe.d/mlx4_core.conf if it does not exist, otherwise delete its contents.

1. If the fields in the example above do not appear in the [HCA] section, meaning SR-IOV is not supported in the used INI.
2. If SR-IOV is supported, to enable if it is not, it is sufficient to set “sriov_en = true” in the INI.

Step 8. Insert an "option" line in the /etc/modprobe.d/mlx4_core.conf file to set the number of VFs. the protocol type per port, and the allowed number of virtual functions to be used by the physical function driver (probe_vf).

```
options mlx4_core num_vfs=5 probe_vf=1
```

Parameter	Recommended Value

num_vfs	Absent, or zero:
	• The SRI-OV mode is not enabled in the driver, hence no VFs will be available.
	• Its value is a single number in the range of 0-63. The driver will enable the `num_vfs` VFs on the HCA and this will be applied to all ConnectX® HCAs on the host.
	• Its format is a string which allows the user to specify the `num_vfs` parameter separately per installed HCA.
	• Its format is: "bb:dd.f-v,bb:dd.f-v,…"
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to enable for that HCA
	This parameter can be set in one of the following ways. For example:
	• `num_vfs=5` - The driver will enable 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
	• `num_vfs=00:04.0-5,00:07.0-8` - The driver will enable 5 VFs on the HCA positioned in BDF 00:04.0 and 8 on the one in 00:07.0)
	Note: PFs not included in the above list will not have SR-IOV enabled.

probe_vf	Absent, or zero:
	• No VFs will be used by the PF driver
	• Its value is a single number in the range of 0-63. Physical Function driver will use `probe_vf` VFs and this will be applied to all ConnectX® HCAs on the host.
	• Its format is a string which allows the user to specify the `probe_vf` parameter separately per installed HCA.
	• Its format is: "bb:dd.f-v,bb:dd.f-v,…
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to use in the PF driver for that HCA
	This parameter can be set in one of the following ways. For example:
	• `probe_vfs=5` - The PF driver will probe 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
	• `probe_vfs=00:04.0-5,00:07.0-8` - The PF driver will probe 5 VFs on the HCA positioned in BDF 00:04.0 and 8 for the one in 00:07.0)
	Note: PFs not included in the above list will not use any of their VFs in the PF driver.

The example above loads the driver with 5 VFs (num_vfs). The standard use of a VF is a single VF per a single VM. However, the number of VFs varies upon the working mode requirements.

Step 9. Reboot the server.

If the SR-IOV is not supported by the server, the machine might not come out of boot/load.

Step 10. Load the driver and verify the SR-IOV is supported. Run:

```
lspci | grep Mellanox
03:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX VPI PCIe 2.0 5GT/s - IB QDR / 10GigE]
(rev b0)
03:00.1 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.2 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.3 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.4 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.5 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
```

Where:

- “03:00" represents the Physical Function
- “03:00.**X**" represents the Virtual Function connected to the Physical Function

5.4.3 Enabling SR-IOV and Para Virtualization on the Same Setup

➢ ***To enable SR-IOV and Para Virtualization on the same setup:***

Step 1. Create a bridge.

```
vim /etc/sysconfig/network-scripts/ifcfg-bridge0
DEVICE=bridge0
TYPE=Bridge
IPADDR=12.195.15.1
NETMASK=255.255.0.0
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
DELAY=0
```

Step 2. Change the related interface (in the example below bridge0 is created over eth5).

```
DEVICE=eth5
BOOTPROTO=none
STARTMODE=on
HWADDR=00:02:c9:2e:66:52
TYPE=Ethernet
NM_CONTROLLED=no
ONBOOT=yes
BRIDGE=bridge0
```

Step 3. Restart the service network.

Step 4. Attach a virtual NIC to VM.

```
ifconfig -a
…
eth6      Link encap:Ethernet  HWaddr 52:54:00:E7:77:99
          inet addr:13.195.15.5  Bcast:13.195.255.255  Mask:255.255.0.0
          inet6 addr: fe80::5054:ff:fee7:7799/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:481 errors:0 dropped:0 overruns:0 frame:0
          TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22440 (21.9 KiB)  TX bytes:19232 (18.7 KiB)
          Interrupt:10 Base address:0xa000
…
```

Step 5. Add the MAC `52:54:00:E7:77:99` to the `/sys/class/net/eth5/fdb` table on HV.

```
Before:

cat /sys/class/net/eth5/fdb
33:33:00:00:02:02
33:33:ff:2e:66:52
01:00:5e:00:00:01
33:33:00:00:00:01

Do:
echo "+52:54:00:E7:77:99" > /sys/class/net/eth5/fdb

After:
cat /sys/class/net/eth5/fdb
52:54:00:e7:77:99
33:33:00:00:02:02
33:33:ff:2e:66:52
01:00:5e:00:00:01
33:33:00:00:00:01…
```

5.4.4 Assigning a Virtual Function to a Virtual Machine

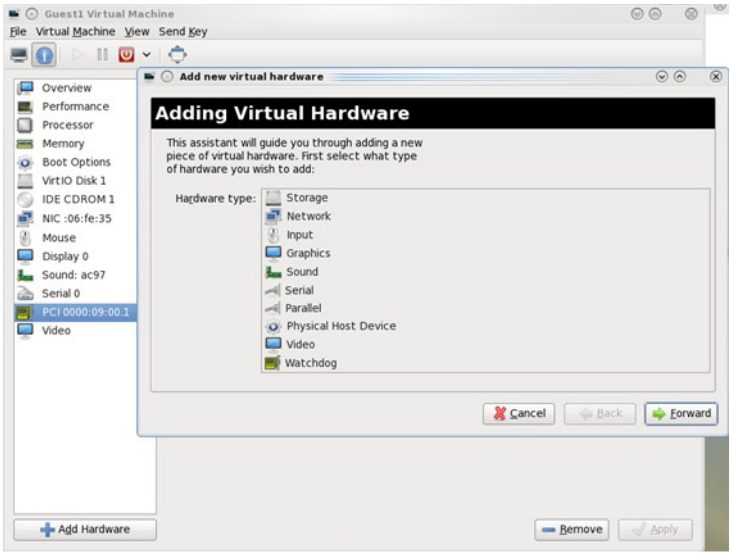
This section will describe a mechanism for adding a SR-IOV VF to a Virtual Machine.

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

Step 1. Run the virt-manager.

Step 2. Double click on the virtual machine and open its Properties.

Step 3. Go to Details->Add hardware ->PCI host device.



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

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

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

```
lspci | grep Mellanox

00:03.0 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
```

Step 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.

5.4.5 Uninstalling SR-IOV Driver

➢ ***To uninstall SR-IOV driver, perform the following:***

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

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

```
# /sbin/mlnx_en_uninstall.sh
MLNX_EN uninstall done
```

Step 3. Restart the server.

5.4.6 Burning Firmware with SR-IOV

The following procedure explains how to create a binary image with SR-IOV enabled that has 63 VFs. However, the number of VFs varies according to the working mode requirements.

➢ ***To burn the firmware:***

Step 1. Verify you have MFT installed in your machine.

Step 2. Enter the firmware directory, according to the HCA type (e.g. ConnectX®-3).

The path is: /mlnx_en/firmware/<device>/<FW version>

Step 3. Find the ini file that contains the HCA's PSID. Run:

```
# mstflint -d 03:00.0 q | grep PSID
PSID:            MT_1090110019
```

If such ini file cannot be found in the firmware directory, you may want to dump the configuration file using mstflint. Run:

```
#  mstflint -dev <PCI device> dc > <ini device file>
```

Step 4. Edit the ini file that you found in the previous step, and add the following lines to the [HCA] section in order to support 63 VFs.

```
;; SRIOV enable
total_vfs = 63[a]
num_pfs = 1
sriov_en = true
```

a. Some servers might have issues accepting 63 Virtual Functions or more. In such case, please set the number of "total_vfs" to any required value.

Step 5. Create a binary image using the modified ini file.

Step a. Download the Mellanox Firmware Tools (www.mellanox.com > Products > Adapter IB/VPI SW > Firmware Tools) and install the package.

Step b. Run:

```
#  mlxburn -fw ./<fw name>mlx -conf <modified ini file> -wrimage <file name>.bin
```

The file <file name>.bin is a firmware binary file with SR-IOV enabled that has 63 VFs. It can be spread across all machines and can be burnt using mstflint, which is part of the bundle, using the following command:

```
#  mstflint -dev <PCI device> -image <file name>.bin  b
```

After burning the firmware, the machine **must be rebooted**. If the driver is only restarted, the machine may hang and a reboot using power OFF/ON might be required.

5.4.7 Ethernet Virtual Function Configuration when Running SR-IOV

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

When running ETH ports on VFs, 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. Any vlan-tagged packets sent by the VF are silently dropped. The default behavior is VGT.

The feature may be controlled on the Hypervisor from userspace via iprout2 / netlink:

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

use:

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

- where `NUM = 0..max-vf-num`
- vlan_id = `0..4095` (4095 means `"set VGT"`)
- qos = `0..7`

For example:

- `ip link set dev eth2 vf 2 qos 3` - sets VST mode for VF #2 belonging to PF eth2, with qos = 3
- `ip link set dev eth2 vf 4095` - sets mode for VF 2 back to VGT

5.4.7.2 Additional Ethernet VF Configuration Options

- Guest MAC configuration

  By default, guest MAC addresses are configured to be all zeroes. In the MLNX_EN guest driver, if a guest sees a zero MAC, it generates a random MAC address for itself. 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 guests, which do not generate random MACs, the adminstrator 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]
```

5.5 Ethernet Performance Counters

Counters are used to provide information about how well an operating system, an application, a service, or a driver is performing. The counter data helps determine system bottlenecks and fine-tune the system and application performance. The operating system, network, and devices provide counter data that an application can consume to provide users with a graphical view of how well the system is performing.

The counter index is a QP attribute given in the QP context. Multiple QPs may be associated with the same counter set, If multiple QPs share the same counter its value represents the cumulative total.

- ConnectX®-3 support 127 different counters which allocated:
  - 4 counters reserved for PF - 2 counters for each port
  - 2 counters reserved for VF - 1 counter for each port
  - All other counters if exist are allocated by demand
- RoCE counters are available only through sysfs located under:
  - # /sys/class/infiniband/mlx4_*/ports/*/counters/
  - # /sys/class/infiniband/mlx4_*/ports/*/counters_ext/
- Physical Function can also read Virtual Functions' port counters through sysfs located under:
  - # /sys/class/net/eth*/vf*_statistics/

To display the network device Ethernet statistics, you can run:

```
Ethtool -S <devname>
```

Table 7 - Port IN Counters

Counter	Description
rx_packets	Total packets successfully received.
rx_bytes	Total bytes in successfully received packets.
rx_multicast_packets	Total multicast packets successfully received.
rx_broadcast_packets	Total broadcast packets successfully received.
rx_errors	Number of receive packets that contained errors preventing them from being deliverable to a higher-layer protocol.
rx_dropped	Number of receive packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.

Table 7 - Port IN Counters

Counter	Description
rx_length_errors	Number of received frames that were dropped due to an error in frame length
rx_over_errors	Number of received frames that were dropped due to overflow
rx_crc_errors	Number of received frames with a bad CRC that are not runts, jabbers, or alignment errors
rx_jabbers	Number of received frames with a length greater than MTU octets and a bad CRC
rx_in_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1500:46] (42 is also counted for VLANtagged frames)
rx_out_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1535:1501]
rx_lt_64_bytes_packets	Number of received 64-or-less-octet frames
rx_127_bytes_packets	Number of received 65-to-127-octet frames
rx_255_bytes_packets	Number of received 128-to-255-octet frames
rx_511_bytes_packets	Number of received 256-to-511-octet frames
rx_1023_bytes_packets	Number of received 512-to-1023-octet frames
rx_1518_bytes_packets	Number of received 1024-to-1518-octet frames
rx_1522_bytes_packets	Number of received 1519-to-1522-octet frames
rx_1548_bytes_packets	Number of received 1523-to-1548-octet frames
rx_gt_1548_bytes_packets	Number of received 1549-or-greater-octet frames

Table 8 - Port OUT Counters

Counter	Description
tx_packets	Total packets successfully transmitted.
tx_bytes	Total bytes in successfully transmitted packets.
tx_multicast_packets	Total multicast packets successfully transmitted.
tx_broadcast_packets	Total broadcast packets successfully transmitted.
tx_errors	Number of frames that failed to transmit
tx_dropped	Number of transmitted frames that were dropped

Table 8 - Port OUT Counters

Counter	Description
tx_lt_64_bytes_packets	Number of transmitted 64-or-less-octet frames
tx_127_bytes_packets	Number of transmitted 65-to-127-octet frames
tx_255_bytes_packets	Number of transmitted 128-to-255-octet frames
tx_511_bytes_packets	Number of transmitted 256-to-511-octet frames
tx_1023_bytes_packets	Number of transmitted 512-to-1023-octet frames
tx_1518_bytes_packets	Number of transmitted 1024-to-1518-octet frames
tx_1522_bytes_packets	Number of transmitted 1519-to-1522-octet frames
tx_1548_bytes_packets	Number of transmitted 1523-to-1548-octet frames
tx_gt_1548_bytes_packets	Number of transmitted 1549-or-greater-octet frames

Table 9 - Port VLAN Priority Tagging (where <i> is in the range 0…7)

Counter	Description
rx_prio_<i>_packets	Total packets successfully received with priority i.
rx_prio_<i>_bytes	Total bytes in successfully received packets with priority i.
rx_novlan_packets	Total packets successfully received with no VLAN priority.
rx_novlan_bytes	Total bytes in successfully received packets with no VLAN priority.
tx_prio_<i>_packets	Total packets successfully transmitted with priority i.
tx_prio_<i>_bytes	Total bytes in successfully transmitted packets with priority i.
tx_novlan_packets	Total packets successfully transmitted with no VLAN priority.
tx_novlan_bytes	Total bytes in successfully transmitted packets with no VLAN priority.

Table 10 - Port Pause (where <i> is in the range 0…7)

Counter	Description
rx_pause_prio_<i>	The total number of PAUSE frames received from the far-end port
rx_pause_duration_prio_<i>	The total time in microseconds that far-end port was requested to pause transmission of packets.

Table 10 - Port Pause (where <i> is in the range 0…7)

Counter	Description
rx_pause_transition_prio_<i>	The number of receiver transitions from XON state (paused) to XOFF state (non-paused)
tx_pause_prio_<i>	The total number of PAUSE frames sent to the far-end port
tx_pause_duration_prio_<i>	The total time in microseconds that transmission of packets has been paused
tx_pause_transition_prio_<i>	The number of transmitter transitions from XON state (paused) to XOFF state (non-paused)

Table 11 - VPort Statistics (where <i>=<empty_string> is the PF, and ranges 1…NumOfVf per VF)

Counter	Description
vport<i>_rx_unicast_packets	Unicast packets received successfully
vport<i>_rx_unicast_bytes	Unicast packet bytes received successfully
vport<i>_rx_multicast_packets	Multicast packets received successfully
vport<i>_rx_multicast_bytes	Multicast packet bytes received successfully
vport<i>_rx_broadcast_packets	Broadcast packets received successfully
vport<i>_rx_broadcast_bytes	Broadcast packet bytes received successfully
vport<i>_rx_dropped	Received packets discarded due to out-of-buffer condition
vport<i>_rx_errors	Received packets discarded due to receive error condition
vport<i>_tx_unicast_packets	Unicast packets sent successfully
vport<i>_tx_unicast_bytes	Unicast packet bytes sent successfully
vport<i>_tx_multicast_packets	Multicast packets sent successfully
vport<i>_tx_multicast_bytes	Multicast packet bytes sent successfully
vport<i>_tx_broadcast_packets	Broadcast packets sent successfully

Table 11 - VPort Statistics (where <i>=<empty_string> is the PF, and ranges 1…NumOfVf per VF)

Counter	Description
vport<i>_tx_broadcast_bytes	Broadcast packet bytes sent successfully
vport<i>_tx_errors	Packets dropped due to transmit errors

Table 12 - SW Statistics

Counter	Description
rx_lro_aggregated	Number of packets aggregated
rx_lro_flushed	Number of LRO flush to the stack
rx_lro_no_desc	Number of times LRO description was not found
rx_alloc_failed	Number of times failed preparing receive descriptor
rx_csum_good	Number of packets received with good checksum
rx_csum_none	Number of packets received with no checksum indication
tx_chksum_offload	Number of packets transmitted with checksum offload
tx_queue_stopped	Number of times transmit queue suspended
tx_wake_queue	Number of times transmit queue resumed
tx_timeout	Number of times transmitter timeout
tx_tso_packets	Number of packet that were aggregated

Table 13 - Per Ring (SW) Statistics (where <i> is the ring I – per configuration)

Counter	Description
rx<i>_packets	Total packets successfully received on ring i
rx<i>_bytes	Total bytes in successfully received packets on ring i.
tx<i>_packets	Total packets successfully transmitted on ring i.
tx<i>_bytes	Total bytes in successfully transmitted packets on ring i.

6 Performance Tuning

6.1 Increasing Packet Rate

To increase packet rate (especially for small packets), set the value of `"high_rate_steer"` module parameter in mlx4_module to 1 (default 0).

Enabling this mode will cause the following chassis management features to stop working:

- NC-SI
- RoL

6.2 General System Configurations

The following sections describe recommended configurations for system components and/or interfaces. Different systems may have different features, thus some recommendations below may not be applicable.

6.2.1 PCI Express (PCIe) Capabilities

Table 14 - Recommended PCIe Configuration

PCIe Generation	3.0
Speed	8GT/s
Width	x8 or x16
Max Payload size	256
Max Read Request	4096

For ConnectX3® based network adapters, 40GbE Ethernet adapters it is recommended to use an x16 PCIe slot to benefit from the additional buffers allocated by the CPU.

6.2.2 Memory Configuration

For high performance it is recommended to use the highest memory speed with fewest DIMMs and populate all memory channels for every CPU installed.

For further information, please refer to your vendor's memory configuration instructions or memory configuration tool available Online.

6.2.3 Recommended BIOS Settings

These performance optimizations may result in higher power consumption.

6.2.3.1 General

Set BIOS power management to Maximum Performance.

6.2.3.2 Intel® Sandy Bridge Processors

The following table displays the recommended BIOS settings in machines with Intel code name Sandy Bridge based processors.

Table 15 - Recommended BIOS Settings for Intel Sandy Bridge Processors

	BIOS Option	Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Enabled
	Hyper-Threading[a]	HPC: disabled Data Centers: enabled
	CPU frequency select	Max performance
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

a. Hyper-Threading can increase message rate for multi process applications by having more logical cores. It might increase the latency of a single process, due to lower frequency of a single logical core when hyper-threading is enabled.

6.2.3.3 Intel® Nehalem/Westmere Processors

The following table displays the recommended BIOS settings in machines with Intel Nehalem-based processors.Configuring the Completion Queue Stall Delay.

Table 16 - Recommended BIOS Settings for Intel® Nehalem/Westmere Processors

	BIOS Option	Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Disabled
	Hyper-Threading[a]	Disabled Recommended for latency and message rate sensitive applications.
	CPU frequency select	Max performance
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

a. Hyper-Threading can increase message rate for multi process applications by having more logical cores. It might increase the latency of a single process, due to lower frequency of a single logical core when hyper-threading is enabled.

6.2.3.4 AMD Processors

The following table displays the recommended BIOS settings in machines with AMD based processors.

Table 17 - Recommended BIOS Settings for AMD Processors

	BIOS Option	Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Disabled
	HPC Optimizations	Enabled
	CPU frequency select	Max performance

Table 17 - Recommended BIOS Settings for AMD Processors

	BIOS Option	Values
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

6.3 Performance Tuning for Linux

You can use the Linux *sysctl* command to modify default system network parameters that are set by the operating system in order to improve IPv4 and IPv6 traffic performance. Note, however, that changing the network parameters may yield different results on different systems. The results are significantly dependent on the CPU and chipset efficiency.

6.3.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance

The following changes are recommended for improving IPv4 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

- Enable the TCP selective acks option for better throughput:

```
sysctl -w net.ipv4.tcp_sack=1
```

- Increase the maximum length of processor input queues:

```
sysctl -w net.core.netdev_max_backlog=250000
```

- Increase the TCP maximum and default buffer sizes using setsockopt():

```
sysctl  -w net.core.rmem_max=4194304
sysctl  -w net.core.wmem_max=4194304
sysctl  -w net.core.rmem_default=4194304
sysctl  -w net.core.wmem_default=4194304
sysctl  -w net.core.optmem_max=4194304
```

- Increase memory thresholds to prevent packet dropping:

```
sysctl -w net.ipv4.tcp_rmem="4096 87380 4194304"
sysctl -w net.ipv4.tcp_wmem="4096 65536 4194304"
```

- "Enable low latency mode for TCP:

```
sysctl -w net.ipv4.tcp_low_latency=1
```

6.3.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance

The following changes are recommended for improving IPv6 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

- Enable the TCP selective acks option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_sack=1
```

6.3.3 Preserving Your Performance Settings after a Reboot

To preserve your performance settings after a reboot, you need to add them to the file `/etc/sysctl.conf` as follows:

```
<sysctl name1>=<value1>
<sysctl name2>=<value2>
<sysctl name3>=<value3>
<sysctl name4>=<value4>
```

For example, “Tuning the Network Adapter for Improved IPv4 Traffic Performance” on page 48 lists the following setting to disable the TCP timestamps option:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

In order to keep the TCP timestamps option disabled after a reboot, add the following line to `/etc/sysctl.conf`:

```
net.ipv4.tcp_timestamps=0
```

6.3.4 Tuning Power Management

Check that the output CPU frequency for each core is equal to the maximum supported and that all core frequencies are consistent.

- Check the maximum supported CPU frequency:

```
#cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_max_freq
```

- Check that core frequencies are consistent:

```
#cat /proc/cpuinfo | grep "cpu MHz"
```

- Check that the output frequencies are the same as the maximum supported.

  If the CPU frequency is not at the maximum, check the BIOS settings according to tables in is section “Recommended BIOS Settings” on page 46 to verify that power state is disabled.

- Check the current CPU frequency to check whether it is configured to max available frequency:

```
#cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq
```

6.3.4.1 Setting the Scaling Governor

If the following modules are loaded, CPU scaling is supported, and you can improve performance by setting the scaling mode to performance:

- freq_table
- acpi_cpufreq: this module is architecture dependent.

It is also recommended to disable the module cpuspeed; this module is also architecture dependent.

➢ ***To set the scaling mode to performance, use:***

```
# echo performance > /sys/devices/system/cpu/cpu7/cpufreq/scaling_governor
```

➢ ***To disable cpuspeed, use:***

```
# service cpuspeed stop
```

6.3.4.2 Kernel Idle Loop Tuning

The mlx4_en kernel module has an optional parameter that can tune the kernel idle loop for better latency. This will improve the CPU wake up time but may result in higher power consumption.

To tune the kernel idle loop, set the following options in the `/etc/modprobe.d/mlx4.conf` file:

- For MLNX_EN 2.0.x

```
options mlx4_core enable_sys_tune=1
```

- For MLNX_EN 1.5.10

```
options mlx4_en enable_sys_tune=1
```

6.3.4.3 OS Controlled Power Management

Some operating systems can override BIOS power management configuration and enable c-states by default, which results in a higher latency.

➢ ***To resolve the high latency issue, please follow the instructions below:***

1. Edit the /boot/grub/grub.conf file or any other bootloader configuration file.
2. Add the following kernel parameters to the bootloader command.

```
intel_idle.max_cstate=0 processor.max_cstate=1
```

3. Reboot the system.

   Example:

```
title RH6.2x64
root (hd0,0)
kernel /vmlinuz-RH6.2x64-2.6.32-220.el6.x86_64
root=UUID=817c207b-c0e8-4ed9-9c33-c589c0bb566f console=tty0
console=ttyS0,115200n8 rhgb intel_idle.max_cstate=0 processor.max_cstate=1
```

6.3.5 Interrupt Moderation

Interrupt moderation is used to decrease the frequency of network adapter interrupts to the CPU. Mellanox network adapters use an adaptive interrupt moderation algorithm by default. The algorithm checks the transmission (Tx) and receive (Rx) packet rates and modifies the Rx interrupt moderation settings accordingly.

To manually set Tx and/or Rx interrupt moderation, use the ethtool utility. For example, the following commands first show the current (default) setting of interrupt moderation on the interface eth1, then turns off Rx interrupt moderation, and last shows the new setting.

```
> ethtool -c eth1
Coalesce parameters for eth1:
Adaptive RX: on  TX: off
...
pkt-rate-low: 400000
pkt-rate-high: 450000

rx-usecs: 16
rx-frames: 88
rx-usecs-irq: 0
rx-frames-irq: 0
...

> ethtool -C eth1 adaptive-rx off rx-usecs 0 rx-frames 0

> ethtool -c eth1
Coalesce parameters for eth1:
Adaptive RX: off TX: off
...
pkt-rate-low: 400000
pkt-rate-high: 450000

rx-usecs: 0
rx-frames: 0
rx-usecs-irq: 0
rx-frames-irq: 0
...
```

6.3.6 Tuning for NUMA Architecture

6.3.6.1 Tuning for Intel® Sandy Bridge Platform

The Intel Sandy Bridge processor has an integrated PCI express controller. Thus every PCIe adapter OS is connected directly to a NUMA node.

On a system with more than one NUMA node, performance will be better when using the local NUMA node to which the PCIe adapter is connected.

In order to identify which NUMA node is the adapter's node the system BIOS should support ACPI SLIT.

- ***To see if your system supports PCIe adapter's NUMA node detection:***

```
# cat /sys/class/net/[interface]/device/numa_node
# cat /sys/devices/[PCI root]/[PCIe function]/numa_node
```

Example for supported system:

```
# cat /sys/class/net/eth3/device//numa_node
0
```

Example for unsupported system:

```
# cat /sys/class/net/ib0/device/numa_node
-1
```

6.3.6.1.1 Improving Application Performance on Remote NUMA Node

Verbs API applications that mostly use polling, will have an impact when using the remote NUMA node.

libmlx4 has a build-in enhancement that recognizes an application that is pinned to a remote NUMA node and activates a flow that improves the out-of-the-box latency and throughput.

However, the NUMA node recognition must be enabled as described in section "Tuning for Intel® Sandy Bridge Platform" on page 51.

In systems which do not support SLIT, the following environment variable should be applied:

```
MLX4_LOCAL_CPUS=0x[bit mask of local NUMA node]
```

Example for local NUMA node which its cores are 0-7:

```
MLX4_LOCAL_CPUS=0xff
```

Additional modification can apply to impact this feature by changing the following environment variable:

```
MLX4_STALL_NUM_LOOP=[integer] (default: 400)
```

The default value is optimized for most applications. However, several applications might benefit from increasing/decreasing this value.

6.3.6.2 Tuning for AMD® Architecture

On AMD architecture there is a difference between a 2 socket system and a 4 socket system.

- With a 2 socket system the PCIe adapter will be connected to socket 0 (nodes 0,1).
- With a 4 socket system the PCIe adapter will be connected either to socket 0 (nodes 0,1) or to socket 3 (nodes 6,7).

6.3.6.3 Recognizing NUMA Node Cores

➢ ***To recognize NUMA node cores, run the following command:***

```
# cat /sys/devices/system/node/node[X]/cpulist | cpumap
```

Example:

```
# cat /sys/devices/system/node/node1/cpulist
1,3,5,7,9,11,13,15
# cat /sys/devices/system/node/node1/cpumap
0000aaaa
```

6.3.6.3.1 Running an Application on a Certain NUMA Node

In order to run an application on a certain NUMA node, the process affinity should be set in either in the command line or an external tool.

For example, if the adapter's NUMA node is 1 and NUMA 1 cores are 8-15 then an application should run with process affinity that uses 8-15 cores only.

➢ ***To run an application, run the following commands:***

```
taskset -c 8-15 ib_write_bw -a
```

or:

```
taskset 0xff00 ib_write_bw -a
```

6.3.7 IRQ Affinity

The affinity of an interrupt is defined as the set of processor cores that service that interrupt. To improve application scalability and latency, it is recommended to distribute interrupt requests (IRQs) between the available processor cores. To prevent the Linux IRQ balancer application from interfering with the interrupt affinity scheme, the IRQ balancer must be turned off.

The following command turns off the IRQ balancer:

```
> /etc/init.d/irqbalance stop
```

The following command assigns the affinity of a single interrupt vector:

```
> echo <hexadecimal bit mask> > /proc/irq/<irq vector>/smp_affinity
```

Bit i in <hexadecimal bit mask> indicates whether processor core i is in <irq vector>'s affinity or not.

6.3.7.1 IRQ Affinity Configuration

It is recommended to set each IRQ to a different core.

For Sandy Bridge or AMD systems set the irq affinity to the adapter's NUMA node:

- For optimizing single-port traffic, run:

```
set_irq_affinity_bynode.sh <numa node> <interface>
```

- For optimizing dual-port traffic, run:

```
set_irq_affinity_bynode.sh <numa node> <interface1> <interface2>
```

- To show the current irq affinity settings, run:

```
show_irq_affinity.sh <interface>
```

6.3.7.2 Auto Tuning Utility

MLNX_EN 2.0.x introduces a new affinity tool called mlnx_affinity. This tool can automatically adjust your affinity settings for each network interface according to the system architecture.

Usage:

- Start

```
# mlnx_affinity start
```

- Stop

```
# mlnx_affinity stop
```

- Restart

```
# mlnx_affinity restart
```

mlnx_affinity can also be started by driver load/unload

➢ ***To enable mlnx_affinity by default:***

- Add the line below to the /etc/infiniband/openib.conf file.

```
RUN_AFFINITY_TUNER=yes
```

6.3.7.3 Tuning for Multiple Adapters

When optimizing the system performance for using more than one adapter. It is recommended to separate the adapter's core utilization so there will be no interleaving between interfaces.

The following script can be used to separate each adapter's IRQs to different set of cores.

```
# set_irq_affinity_cpulist.sh <cpu list>
<interface>
<cpu list> can be either a comma separated list of single core numbers (0,1,2,3)
or core groups (0-3)
```

Example:

If the system has 2 adapters on the same NUMA node (0-7) each with 2 interfaces run the following:

```
# /etc/init.d/irqbalancer stop
# set_irq_affinity_cpulist.sh 0-1 eth2
# set_irq_affinity_cpulist.sh 2-3 eth3
# set_irq_affinity_cpulist.sh 4-5 eth4
# set_irq_affinity_cpulist.sh 6-7 eth5
```

6.3.8 Tuning Multi-Threaded IP Forwarding

➢ ***To optimize NIC usage as IP forwarding:***

1. Set the following options in /etc/modprobe.d/mlx4.conf:
   - For MLNX_EN-2.0.x:

```
options mlx4_en  inline_thold=0
options mlx4_core high_rate_steer=1
```

   - For MLNX_EN-1.5.10:

```
options mlx4_en num_lro=0 inline_thold=0
options mlx4_core high_rate_steer=1
```

2. Apply interrupt affinity tuning.
3. Forwarding on the same interface:

```
# set_irq_affinity_bynode.sh <numa node> <interface>
```

4. Forwarding from one interface to another:

```
# set_irq_affinity_bynode.sh <numa node> <interface1> <interface2>
```

5. Disable adaptive interrupt moderation and set status values, using:

```
# ethtool -C adaptive-rx off
```