Connect. Accelerate. Outperform.™

Mellanox OFED for FreeBSD User Manual

Rev 2.1.5

www.mellanox.com

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

2.1-2.1.5	January 15, 2015	• Added the following sections:
		  • Section 4, “Driver Features”, on page 16
		• Updated the following sections:
		  • Section 1, “Overview”, on page 9
		  • Section 2, “Installation”, on page 10
		  • Section 5, “Performance Tuning”, on page 18

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

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
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
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet
SL	Service Level
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane

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

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.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
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.
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

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > Software > Ethernet Drivers > FreeBSD Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Overview

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

The driver rele ase exposes the following capabilities:

- Single/Dual port
- Up to 16 Rx queues per port
- Up to 32 Tx queues per port (according to number of CPUs)
- MSI-X or INTx
- Adaptive interrupt moderation
- HW Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- VLAN Tx/Rx acceleration (HW VLAN stripping/insertion)
- Net device statistics

1.1 Mellanox OFED for FreeBSD Package Contents

1.1.1 Tarball Package

Mellanox OFED for FreeBSD package includes the following directories:

- modules - contains the relevant Makefiles
- ofed - source code

1.1.2 Software Components

Mellanox OFED for FreeBSD contains the following software components:

Table 5 - Mellanox OFED for FreeBSD Software Components

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, mlx4_en.
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.
mlx4_ib	Handles InfiniBand specific functions supplied by ib_core in order to interact with verbs and ULPs.
Documentation	Release Notes, User Manual

2 Installation

This chapter describes how to install and test the Mellanox driver for FreeBSD package on a single host machine with Mellanox InfiniBand and/or Ethernet adapter hardware installed.

2.1 Software Dependencies

- To install the driver software, kernel sources must be installed on the machine.
- To run the Packet Pacing feature, the attached kernel patch and firmware must be installed on the machine.

2.2 Downloading Mellanox Driver for FreeBSD

Step 1. Verify that the system has a Mellanox network adapter (HCA/NIC) installed.

The following example shows a system with an installed Mellanox HCA:

```
# pciconf -lv | grep Mellanox -C 3
mlx4_core0@pci0:7:0:0:  class=0x028000 card=0x000615b3 chip=0x100315b3 rev=0x00
hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27500 Family [ConnectX-3]'
    class      = network
```

Step 2. Download the tarball image to your host.

The image name has the format `MLNX_OFED_FreeBSD-<ver>.tgz`. You can download it from
http://www.mellanox.com > Products > Software > Ethernet Drivers > FreeBSD

or

http://www.mellanox.com > Products > Software > InfiniBand/VPI drivers > FreeBSD

Step 3. Use the md5sum utility to confirm the file integrity of your tarball image.

2.3 Installing Mellanox Driver for FreeBSD

Prior to installing the driver, please re-compile (and install) the kernel with NO OFED options/ devices enabled.

Step 1. Extract the tarball.

Step 2. Compile and load needed modules in the following order of dependencies:

- Ethernet Driver:

  mlx4_core

  a. Go to the mlx4 directory. Run:

  ```
  # cd modules/mlx4
  ```

  b. Clean any previous dependencies. Run:

  ```
  # make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
  ```

c. Compile the mlx4_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

d. Install the mlx4_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the mlx4_core module. Run:

```
# kldload mlx4
```

mlxen

a. Go to the mlxen directory. Run:

```
# cd modules/mlxen
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the mlxen module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

Note: For packet pacing support, add `CONFIG_RATELIMIT=yes`. This option has a kernel patch dependency.

d. Install the mlxen module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the mlxen module. Run:

```
# kldload mlxen
```

- InfiniBand Driver:

mlx4_core

Run the same steps as specified for Ethernet driver above.

ib_core

a. Go to the ib_core directory. Run:

```
# cd modules/ibcore
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the ib_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

d. Install the ib_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the ib_core module. Run:

```
# kldload ibcore
```

mlx4_ib

a. Go to the mlx4_ib directory. Run:

```
# cd modules/mlx4ib
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the mlx4_ib module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

d. Install the mlx4_ib module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the mlx4_ib module. Run:

```
# kldload mlx4ib
```

ipoib

Run the same steps specified above for $HEAD/sys/modules/ipoib.

To load a module on reboot, add "mlx4_core=="YES"/mlxen_load="YES"/ibcore_load="YES"/ mlx4ib_load="YES"" to the '/boot/ loader.conf' file (create if does not exist).

Run "kdlstat" in order to verify which modules are loaded on your server.

2.4 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.

2.4.1 Installing Firmware Tools

Step 1. 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 FreeBSD" (tarball name is mft-X.X.X.tgz).

Step 2. Extract the tarball and run the installation script.

2.4.2 Downloading Firmware

Step 1. Retrieve device's PCI slot (i.e. pci0:x:0:0). Run:

```
$ mst status
```

Step 2. Verify your card's PSID.

```
$flint -d <pci> q
```

Step 3. Download the desired firmware from the Mellanox website.

http://www.mellanox.com/page/firmware_download

2.4.3 Undating Firmware

Step 1. Unzip the firmware binary file, run.

```
$flint -d <pci> -i <img.bin> b
```

Step 2. Reboot the server.

3 Ethernet Driver

3.1 Usage and Configuration

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

```
#> ifconfig mlxen<x> <ip>
```

Note: <x> is the OS assigned interface number

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

```
#> pciconf -lv | grep mlx
#> flint -d pci<w:x:y:z> q
```

Example:

```
#> pciconf -lv | grep mlx
mlx4_core0@pci0:7:0:0:  class=0x028000 card=0x003715b3 chip=0x100315b3 rev=0x00 hdr=0x00
#> flint -d pci0:7:0:0 q
Image type:      FS2
FW Version:      2.11.1294
Device ID:       4099
Description:     Node             Port1            Port2            Sys image
GUIDs:           0002c903002ffcc0 0002c903002ffcc1 0002c903002ffcc2 0002c903002ffcc3
MACs:                                 0002c92ffcc0     0002c92ffcc1
VSD:
PSID:            MT_1020120019
```

- ***To check driver version:***

```
#> dmesg
```

Example:

```
#> dmesg
mlx4_core: Mellanox ConnectX core driver v2.1 (Aug 21 2014)
mlx4_en: Mellanox ConnectX HCA Ethernet driver v2.1 (Aug 18 2014)
```

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

```
#> ifconfig mlxen<x>
```

Note: <x> is the OS assigned interface number

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

```
#> ifconfig mlxen<x> [rxcsum|-rxcsum] [txcsum|-txcsum] [tso|-tso] [lro|-lro]
```

Note: <x> is the OS assigned interface number

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

```
#> sysctl -a | grep adaptive
```

Example:

```
#> sysctl -a | grep adaptive
hw.mlxen0.conf.adaptive_rx_coal: 1
hw.mlxen1.conf.adaptive_rx_coal: 1
```

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

```
#>sysctl hw.mlxen<x>.conf.adaptive_rx_coal=[1/0]
```

Note: <x> is the OS assigned interface number

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

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

```
#> sysctl -a | grep pkt_rate
#> sysctl -a | grep rx_usecs
```

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

```
#> sysctl hw.mlxen<x>.conf.pkt_rate_low=[N]
#> sysctl hw.mlxen<x>.conf.pkt_rate_high=[N]
#> sysctl hw.mlxen<x>.conf.rx_usecs_low=[N]
#> sysctl hw.mlxen<x>.conf.rx_usecs_high=[N]
```

Note: <x> is the OS assigned interface number

Example:

```
#> sysctl -a | grep pkt_rate
hw.mlxen0.conf.pkt_rate_low: 400000
hw.mlxen0.conf.pkt_rate_high: 450000
hw.mlxen1.conf.pkt_rate_low: 400000
hw.mlxen1.conf.pkt_rate_high: 450000
sysctl -a | grep rx_usecs
hw.mlxen0.conf.rx_usecs_low: 0
hw.mlxen0.conf.rx_usecs_high: 128
hw.mlxen1.conf.rx_usecs_low: 0
hw.mlxen1.conf.rx_usecs_high: 128
```

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 query pause frame settings:***

```
#> ifconfig -m mlxen<x>
```

Note: <x> is the OS assigned interface number

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

```
#> ifconfig -m mlxen<x> [-]mediaopt [rxpause,txpause]
```

Note: <x> is the OS assigned interface number

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

```
#> sysctl -a | grep _size
```

Example:

```
#> sysctl -a | grep _size
hw.mlxen0.conf.rx_size: 1024
hw.mlxen0.conf.tx_size: 1024
hw.mlxen1.conf.rx_size: 1024
hw.mlxen1.conf.tx_size: 1024
```

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

```
#> sysctl hw.mlxen<x>.conf.rx_size=[N]
#> sysctl hw.mlxen<x>.conf.tx_size=[N]
```

Note: <x> is the OS assigned interface number

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

```
#> sysctl -a | grep mlx | grep stat
```

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

4 Driver Features

4.1 RDMA over Converged Ethernet (RoCE)

RoCE allows InfiniBand (IB) transport applications to work over an Ethernet network. RoCE encapsulates the InfiniBand transport and the GRH headers in Ethernet packets bearing a dedicated ether type (0x8915). Thus, any VERB application that works in an InfiniBand fabric can also work in an Ethernet fabric.

RoCE is enabled only for drivers that support VPI (currently, only mlx4).

➢ ***When working with RDMA applications over Ethernet link layer, the following points should be noted:***

- The presence of a Subnet Manager (SM) is not required in the fabric. Thus, operations that require communication with the SM are managed in a different way in RoCE. This does not affect the API.
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with the relevant values before establishing a connection. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure.
- Since LID is a layer 2 attribute of the InfiniBand protocol stack, it is not set for a port and is displayed as zero when querying the port.
- With RoCE, APM is not supported.
- The GID table for each port is populated with N+1 entries where N is the number of IP addresses that are assigned to all network devices associated with the port including VLAN devices.
- The first entry in the GID table (at index 0) for each port is always present and equal to the link local IPv6 address of the net device that is associated with the port. Note that even if the link local IPv6 address is not set, index 0 is still populated.
- GID format can be of 2 types, IPv4 and IPv6. IPv4 GID is a IPv4-mapped IPv6 address[1] while IPv6 GID is the IPv6 address itself.
- Load the following modules for RoCE support: mlx4_core, ib_core, mlx4_ib, and mlx4_en.

4.2 Packet Pacing - Beta Version

Packet pacing, also known as the “Rate limit,” defines a maximum bandwidth allowed for a TCP connection. Limitation is done by HW where each QP (transmit queue) has a rate limit value from which it calculates the delay between each packet sent.

4.2.1 Flow

1. User will create a rate limited ring according to desired rate. Upon success, the ring ID is returned.
2. User will associate the ring ID to a specific TCP socket.

1. For the IPv4 address A.B.C.D the corresponding IPv4-mapped IPv6 address is ::ffff.A.B.C.D

3. Rate Limited TCP traffic will be transmitted when using the TCP socket.
4. User will be able to modify the rate limit value.
5. When closing the TCP socket, user will need to destroy the relevant ring.

4.2.2 Feature Characteristics

- Supports up to 30,000 rate limited TCP connections.
- Each TCP connection is mapped to a specific QP
- **User interface**
  - SIOCARATECTL: Create rate limit ring (QP)
  - SIOCSRATECTL: Modify rate limit value for a specific ring
  - SIOCGRATECTL: Query rate limit value for a specific ring
  - SIOCDRATECTL: Destroy rate limited ring (QP)
  - SIOCSTXRINGID: Associate ring id with a TCP socket
- All Rate limited rings statistics will be available via Sysctl:
  - rate_limit_val
  - packets
  - bytes
- **Limitations**
  - Max rate limit rings 30000
  - Min rate: 500Kbps
  - Max rate: HCA's max BW
  - Different rate limits to be configured per NIC/port: 1024

5 Performance Tuning

5.1 Interrupt Moderation

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

In order to manually set Rx interrupt moderation, use sysctl:

Step 1. Turn OFF the interrupt moderation. Run:

```
#> sysctl hw.mlxen<x>.conf.adaptive_rx_coal=0
```

Note: <x> is the OS assigned interface number

Step 2. Turn ON the interruption moderation. Run:

```
#> sysctl hw.mlxen<xa>.conf.adaptive_rx_coal=1
```

Step 3. Set the threshold values for packet rate limits and for moderation time. Run:

```
#> sysctl hw.mlxen<xa>.conf.pkt_rate_low=[N]
#> sysctl hw.mlxen<xa>.conf.rx_usecs_low=[N]
#> sysctl hw.mlxen<xa>.conf.pkt_rate_high=[N]
#> sysctl hw.mlxen<x>.conf.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.

5.2 Tuning for NUMA Architecture

5.2.1 Single NUMA Architecture

When using a server with single NUMA, no tuning is required. Also, make sure to avoid using core number 0 for interrupts and applications.

1. Find a CPU list:

```
sysctl -a | grep "group level=\"2\"" -A 1
<group level="2" cache-level="2">
<cpu count="12" mask="fff">0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</cpu>
```

2. Tune Mellanox NICs to work on desirable cores
   - Find the NIC's PCI location:

```
pciconf -lv | grep mlx
mlx4_core0@pci0:2:0:0:  class=0x028000 card=0x000315b3 chip=0x100715b3 rev=0x00
hdr=0x00
```

   - Find the NIC's device name by its PCI location

```
sysctl -a | grep pci2
dev.mlx4_core.0.%parent: pci2
```

This means the NIC on PCI number 2 has a logic device called mlx4_core0.

- Find the device interrupts.

```
vmstat -ia | grep mlx4_core0 | awk '{print $1}' | sed s/irq// | sed s/://
285
286
287
…
```

- Bind each interrupt to a desirable core.

```
cpuset -x 285 -l 1
cpuset -x 286 -l 2
cpuset -x 287 -l 3
…
```

- Bind the application to the desirable core.

```
cpuset -l 1-11 <app name> <sever flag>
cpuset -l 1-11 <app name> <client flag> <IP>
```

Specifying a range of CPUs when using the cpuset command will allow the application to choose any of them. This is important for applications that execute on multiple threads.
The range argument is not supported for interrupt binding.

5.2.2 Dual NUMA Architecture

1. Find the CPU list closest to the NIC

- Find the NIC's PCI location:

```
pciconf -lv | grep mlx
mlx4_core0@pci0:4:0:0:  class=0x028000 card=0x000315b3 chip=0x100715b3 rev=0x00
hdr=0x00
```

Usually, low PCI locations are closest to NUMA number 0, and high PCI locations are closest to NUMA number 1. Here is how to verify the locations:

- Find the NIC's pcib by PCI location (in this example, try PCI 4)

```
sysctl -a | grep pci.4.%paren
dev.pci.4.%parent: pcib3
```

- Find the NIC's pcib location:

```
sysctl -a | grep pcib.3.%location
dev.pcib.3.%location: slot=2 function=0 handle=\_SB_.PCI0.NPE3
```

In "handle", PCI0 is the value for locations near NUMA0, and PCI1 is the value for locations near NUMA1.

- Find the cores list of the closest NUMA:

```
sysctl -a | grep "group level=\"2\"" -A 1
<group level="2" cache-level="2">
<cpu count="12" mask="fff">0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</cpu>
--
<group level="2" cache-level="2">
<cpu count="12" mask="fff000">12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23</cpu>
```

Note: Each list of cores refers to a different NUMA.

2. Tune Mellanox NICs to work on desirable cores

Step 1. Pin both interrupts and application processes to the relevant cores.

Step 2. Find the closest NUMA to the NIC

- Find the NIC's device name by its PCI location.

```
sysctl -a | grep pci4
dev.mlx4_core.0.%parent: pci4
```

This means the NIC on PCI number 4 has a logic device called mlx4_core0.

- Find the device interrupts.

```
vmstat -ia | grep mlx4_core0 | awk '{print $1}' | sed s/irq// | sed s/://
338
339
340
…
```

- Bind each interrupt to a core from the closest NUMA cores list
Note: It's best to avoid core number 0.

```
cpuset -x 338 -l 1
cpuset -x 339 -l 2
cpuset -x 340 -l 3
...
```

- Bind the application to the closest NUMA cores list.
Note: It's best to avoid core number 0

```
cpuset -l 1-11 <app name> <sever flag>
cpuset -l 1-11 <app name> <client flag> <IP>
```

For best performance, change CPU's BIOS configuration to performance mode.