Mellanox OFED for FreeBSD for ConnectX-4 User Manual

Rev 3.0.0

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

Table of Contents

List of Tables

Document Revision History

Table 1 - Document Revision History

Revision	Date	Description
3.0.0	November 2015	Initial release

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) in ETH mode adapter cards.

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

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 EN driver for FreeBSD and instructions for installing the driver on Mellanox ConnectX® adapter cards supporting the following uplinks to servers:

- ConnectX®-4
  - Ethernet: 10GigE, 25GigE, 40GigE, 50GigE and 100GigE
- ConnectX®-4 Lx
  - Ethernet: 10GigE, 25GigE, 40GigE and 50GigE

The driver release introduces the following capabilities:

- Single/Dual port
- Number of RX queues per port - according to number of CPUs.
- Number of TX queues per port - according to number of CPUs.
- MSI-X or INTx
- Hardware Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- VLAN Tx/Rx acceleration (Hardware VLAN stripping/insertion)
- ifnet statistics

1.1 Mellanox OFED EN for FreeBSD Package Contents

1.1.1 Tarball Package

Mellanox OFED EN for FreeBSD package includes the following directories:

- mlx5 modules - contains the relevant Makefiles for mlx5 core and EN
- drivers/net/mlx5/ - EN source code
- drivex/mlx5/generated/freebsd/ - core source code

1.1.2 mlx5 driver

mlx5 is the low level driver implementation for the ConnectX-4/ConnectX-4 Lx adapters designed by Mellanox Technologies.

1.1.2.1 Software Components

Mellanox OFED EN for FreeBSD contains the following software components:

Table 3 - Mellanox OFED EN for FreeBSD Software Components

Components	Description
mlx5	Acts as a library of common functions required by the ConnectX®-4/ConnectX-4 Lx adapter cards. For example: initializing the device after reset.
mlx5en	Handles Ethernet specific functions and plugs into the ifnet mid-layer.
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 adapter hardware installed.

2.1 Software Dependencies

- To install the driver software, kernel sources must be installed on the machine.
- To load mlx5, linuxapi must be loaded as well.
  - Compile and install linuxapi module under /sys/modules/linuxapi.

2.2 Downloading Mellanox Driver for FreeBSD

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

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

```
mlx5_core0@ pci0:6:0:0:      class=0x020000 card=0x000815b3 chip=0x101315b3 rev=0x00 hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27620 Family'
    class      = network
    subclass   = ethernet
mlx5_core1@pci0:6:0:1:       class=0x020000 card=0x000815b3 chip=0x101315b3 rev=0x00 hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27620 Family'
    class      = network
    subclass   = ethernet
```

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

2.3 Installing Mellanox Driver for FreeBSD

FreeBSD v3.0.0 supports adapter cards based on the Mellanox ConnectX®-4 family of adapter IC devices only. If you have ConnectX-3 and ConnectX-3 Pro on your server, you will need to install FreeBSD v2.1.6 driver.
For details on how to install FreeBSD v2.1.6 driver, please refer to FreeBSD v2.1.6 User Manual.

1. Extract the tarball.
2. Compile and load needed modules in the following order of dependencies:

   mlx5 core

   a. Go to the mlx5 directory. Run:

```
# cd mlx5_modules/mlx5
```

   b. Clean any previous dependencies. Run:

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

c. Compile the mlx5_core module. Run:

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

d. Install the mlx5_core module. Run:

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

e. Load the mlx5_core module. Run:

```
# kldload mlx5
```

mlx5en

a.Go to the mlx5en directory. Run:

```
# cd mlx5_modules/mlx5en
```

b. Clean any previous dependencies. Run:

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

c. Compile the mlx5en module. Run:

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

d. Install the mlx5en module. Run:

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

e. Load the mlx5en module. Run:

```
# kldload mlx5en
```

To load a module on reboot, add "mlx5_load="YES"/mlx5en_load="YES" to the '/boot/loader.conf' file (create if does not exist).

Run "kldstat" 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).
For ConnectX®-4, you will need at least MFT-4.1.X.X.X.

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

2.4.2 Downloading Firmware

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

```
#> mst status
```

2. Verify your card's PSID.

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

3. Download the desired firmware from the Mellanox website.

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

2.4.3 Updating Firmware Using flint

1. Before burning a new firmware, make sure the modules are unloaded. To unload the modules, run:

```
#> kldunload mlx5en
#>  kldunload mlx5
```

2. Unzip the firmware binary file.
3. Burn the firmware on your server:

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

4. Reboot the server.

2.4.4 Setting the Ports to ETH

If you have a VPI HCA, you will need to set the ports to ETH. This is done by using the mlxconfig tool (part of the MFT).

1. If you have a card with two ports, run:

```
#> mlxconfig -d <pci> set  LINK_TYPE_P1=2 (For the first port)
#> mlxconfig -d <pci> set  LINK_TYPE_P2=2 (For the second port)
```

2. Reboot the server.

2.5 Driver Usage and Configuration

Interface name has changed from mlx5en to mce. Note that ifconfig and sysctl commands were updated accordingly.

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

```
#> ifconfig mce<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
#> flint -d pci0:6:0:0 dc | grep Description
```

Example:

```
#> pciconf -lv  | grep mlx -C 3
mlx5_core0@pci0:33:0:0:     class=0x020000 card=0x001415b3 chip=0x101315b3 rev=0x00 hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27620 Family'
    class      = network
    subclass   = ethernet
mlx5_core1@pci0:33:0:1:     class=0x020000 card=0x001415b3 chip=0x101315b3 rev=0x00 hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27620 Family'
    class      = network
#> flint -d pci0:33:0:0: q
Image type:      FS3
FW Version:      12.12.0610
FW Release Date: 3.9.2015
Description:     UID                GuidsNumber
Base GUID:       e41d2d03006094ec        20
Base MAC:        0000e41d2d6094ec        20
Image VSD:
Device VSD:
PSID:             MT_2190110032
#> flint -d pci0:6:0:0 dc | grep Description
;;Description = ConnectX-4 VPI adapter card; EDR IB (100Gb/s) and 100GbE; dual-port QSFP28;
PCIe3.0 x16; ROHS R6
```

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

```
#>sysctl -a
```

Example:

```
sysctl -a | grep Mellanox
dev.mlx5_core.1.%desc: Mellanox Ethernet driver (3.0.0-RC2)
dev.mlx5_core.0.%desc: Mellanox Ethernet driver (3.0.0-RC2)
```

- ***To check firmware version:***
  - dmesg

    ```
    #> dmesg
    ```

    Example:

    ```
    Mlx5_core0: INFO: firmware version: 12.12.2008
    ```

  - sysctl

    ```
    #> sysctl -a
    ```

    Example:

    ```
    dev.mlx5_core.0.hw.fw_version: 12.12.2008
    ```

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

```
#> ifconfig mce<x>
```

Note: <x> is the OS assigned interface number

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

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

Note: <x> is the OS assigned interface number

➢ ***To query and set interrupt coalescing modes:***

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

Example:

```
#> sysctl -a | grep coalesce_mode
dev.mce.0.conf.rx_coalesce_mode: 1
dev.mce.1.conf.rx_coalesce_mode: 1
```

- coalesce mode '0' indicates interrupt timer is resetting with each interrupt event.
- coalesce mode '1' indicates interrupt timer is resetting with each received packet.

➢ ***To query and modify values for timer initialization between interrupts:***

```
#> sysctl -a |  grep tx_coalesce_usecs
#> sysctl -a |  grep rx_coalesce_usecs
```

➢ ***To query and modify values for number of received packets between interrupts:***

```
#> sysctl -a | grep tx_coalesce_pkts
#> sysctl -a | grep rx_coalesce_pkts
```

Example:

```
#> sysctl -a | grep rx_coalesce_usecs
dev.mce.1.conf.rx_coalesce_usecs: 3
dev.mce.0.conf.rx_coalesce_usecs: 3
#> sysctl -a | grep rx_coalesce_pkts
dev.mce.1.conf.rx_coalesce_pkts: 32
dev.mce.0.conf.rx_coalesce_pkts: 32
```

➢ ***To query ring size values:***

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

Example:

```
#> sysctl -a | grep mlx | grep _size
dev.mce.1.conf.rx_queue_size: 1024
dev.mce.1.conf.tx_queue_size: 1024
dev.mce.1.conf.rx_queue_size_max: 8192
dev.mce.1.conf.tx_queue_size_max: 8192
```

➢ ***To modify rings size:***

```
#> sysctl dev.mce.0.conf.rx_queue_size=[N]
#> sysctl dev.mce.0.conf.tx_queue_size=[N]
```

Note: <x> is the OS assigned interface number

➢ ***To obtain device statistics:***

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

➢ ***To obtain additional device statistics:***

```
#> sysctl dev.mce.0.conf.debug_stats=1
#> sysctl -a | grep mce | grep stats
```

➢ ***To show out of receive buffers counter:***

```
#> sysctl -a | grep out_of_rx_buffer
dev.mce.1.pstats.out_of_rx_buffer: 0
dev.mce.0.pstats.out_of_rx_buffer: 0
```

➢ ***To verify support for Rx/Tx pause frames:***

- ifconfig

```
#> ifconfig
media: Ethernet autoselect (100GBase-CR4 <full-duplex,rxpause,txpause>)
```

- sysctl

```
#> sysctl dev.mce.0.conf.rx_pauseframe_control
dev.mce.0.conf.rx_pauseframe_control: 1
#> sysctl dev.mce.0.conf.tx_pauseframe_control
dev.mce.0.conf.tx_pauseframe_control: 1
```

➢ ***To enable/disable Rx/Tx pause frames:***

```
sysctl dev.mce.0.conf.rx_pauseframe_control=1
sysctl dev.mce.0.conf.tx_pauseframe_control=1
```

Note: 0 = disable, 1 = enable

➢ ***To show all supported media:***

```
#> ifconfig -m mce<x>
supported media:
 media autoselect
 media 50GBase-CR2 mediaopt full-duplex
 media 25GBase-SR mediaopt full-duplex
 media 25GBase-CR mediaopt full-duplex
 media 100GBase-LR4 mediaopt full-duplex
 media 100GBase-SR4 mediaopt full-duplex
 media 100GBase-CR4 mediaopt full-duplex
 media 40Gbase-LR4 mediaopt full-duplex
```

Note: <x> is the OS assigned interface number

The list of supported media is different in ConnectX-4 and ConnectX-4 Lx.

➢ ***To set new media:***

```
#> ifconfig -m mce<x> media <y> mediaopt full-duplex
```

Note: <x> is the OS assigned interface number. <y> is the relevant media

When updating the media, make sure to choose the right cable type.

Once the driver is loaded, both ports will be activated, meaning that an ifnet will be created for each port.

3 Features Overview and Configuration

3.1 Hardware Large Receive Offload (HW LRO)

HW LRO is supported in ConnectX®-4 only.

Large Receive Offload (LRO) increases inbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by aggregating multiple incoming packets from a single stream into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed.

- ***In order to turn on the LRO device, run:***

```
#> ifconfig mce<x> lro
```

- ***In order to turn off the LRO device, run:***

```
#> ifconfig mce<x> -lro
```

When the LRO device is on, HW LRO can be turned on. HW LRO is off by default.

- ***In order to turn on HW LRO run:***

```
#> sysctl dev.mce.0.conf.hw_lro=1
```

- ***In order to turn off HW LRO run:***

```
#> sysctl dev.mce.0.conf.hw_lro=0
```

3.2 EEPROM Cable Information Reader

EEPROM is supported in ConnectX®-4 only.

EEPROM cable reading feature allows reading important information about the plugged cable, such as cable type, cable speed, vendor and more.

In order to read the cable EEPROM info:

1. Read the cable information by enabling the following sysctl parameter. Output will be printed in dmesg:

```
#> sysctl dev.mce.<X>.conf.eeprom_info:=1
```

Example:

```
#>sysctl dev.mce.1.conf.eeprom_info=1
dev.mce.1.conf.eeprom_info: 0 -> 0

 #>dmesg -s
   Offset          Values
   ------          ------
```

```
   0x0000          0d 05 06 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0010          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0020          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0030          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0040          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0050          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0060          00 00 00 00 00 00 00 00 00 00 00 00 01 00 04 00
   0x0070          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0080          0d 00 23 88 00 00 00 00 00 00 00 00 ff 00 00 00
   0x0090          00 00 01 a0 4d 65 6c 6c 61 6e 6f 78 20 20 20 20
   0x00a0          20 20 20 20 1f 00 02 c9 4d 43 50 31 36 30 30 2d
   0x00b0          45 30 30 41 20 20 20 20 41 32 02 03 04 07 00 3f
   0x00c0          0b 00 00 00 4d 54 31 35 32 31 56 53 30 36 34 38
   0x00d0          34 20 20 20 31 35 30 35 32 36 20 20 00 00 67 5e
   0x00e0          31 32 38 38 35 35 32 33 38 44 33 33 00 00 00 00
   0x00f0          00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 00
```

2. Another option for reading cable information is by using the ifconfig:

```
#>ifconfig -v mce<X>
#>ifconfig -vv mce<X>
#>ifconfig -vvv mce<X>
```

Example:

```
#> ifconfig -vvv mce1
plugged: QSFP+ 40GBASE-CR4 (No separate connector)
vendor: Mellanox PN: MCP1600-E00A SN: MT1521VS06484 DATE: 2015-05-26
compliance level: SFF-8636 rev <=1.5
nominal bitrate: 25750 Mbps

 SFF8436 DUMP (0xA0 128..255 range):
 0D 00 23 88 00 00 00 00 00 00 00 00 FF 00 00 00
 00 00 01 A0 4D 65 6C 6C 61 6E 6F 78 20 20 20 20
 20 20 20 20 1F 00 02 C9 4D 43 50 31 36 30 30 2D
 45 30 30 41 20 20 20 20 41 32 02 03 04 07 00 3F
 0B 00 00 00 4D 54 31 35 32 31 56 53 30 36 34 38
 34 20 20 20 31 35 30 35 32 36 20 20 00 00 67 5E
 31 32 38 38 35 35 32 33 38 44 33 33 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 00

 SFF8436 DUMP (0xA0 0..81 range):
 0D 05 06 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00
```

4 Performance Tuning

In order to improve performance, please make sure the HW LRO is enabled.

4.1 Receive Queue Interrupt Moderation

An armed CQ will generate an event when either of the following conditions is met:

- The number of completions generated since the one which trigged the last event generation reached a set in advance number.
- The timer has expired and an event is pending.

The timer can be set to be restarted either upon event generation or upon completion generation.

Setting the timer to be restarted upon completion generation affects the interrupt receiving rate. When receiving a burst of incoming packets, the timer will not reach its limit, therefore, the interrupt rate will be associated to the size of the packets.

➢ ***In order to modify the timer restart mode, run:***

```
#> sysctl dev.mce.1.conf.rx_coalesce_mode=[0/1]
```

0: For timer restart upon event generation.

1: For timer restart upon completion generation.

➢ ***In order to modify the number of completions generated between interrupts, run:***

```
#> sysctl dev.mce.1.conf.rx_coalesce_pkts=<x>
```

➢ ***In order to modify the time for the timer to finish, run:***

```
#> sysctl dev.mce.1.conf.rx_coalesce_usecs=<x>
```

Note: The default values are:

- dev.mce.1.conf.rx_coalesce_mode: 1 - Timer restarts upon completion generation.
- dev.mce.1.conf.rx_coalesce_pkts: 32 - 32 completions generate interrupts.
- dev.mce.1.conf.rx_coalesce_usecs: 3 - Timer count down 3 micro sec.

4.2 Tuning for NUMA Architecture

4.2.1 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

a.Find the device that matches the interface:

```
#> sysctl -a | grep mce | grep mlx
dev.mce.1.conf.device_name: mlx5_core1
dev.mce.0.conf.device_name: mlx5_core0
```

b. Find the device interrupts.

```
vmstat -ia | grep mlx5_core0 | awk '{print $1}' | sed s/irq// | sed s/:://
269
270
271
…
```

c. Bind each interrupt to a desirable core.

```
cpuset -x 269 -l  1
cpuset -x 270 -l  2
cpuset -x 271 -l  3
…
```

d. 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.

4.2.2 Dual NUMA Architecture

1. Find the CPU list closest to the NIC

a. Find the device that matches the interface:

```
#> sysctl -a | grep mce | grep mlx
dev.mce.3.conf.device_name: mlx5_core3
dev.mce.2.conf.device_name: mlx5_core2
dev.mce.1.conf.device_name: mlx5_core1
dev.mce.0.conf.device_name: mlx5_core0
```

b. Find the NIC's PCI location:

```
#> sysctl -a | grep
mlx5_core.0 | grep parent
dev.mlx5_core.0.%parent: pci3
```

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:

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

```
#> sysctl -a | grep pci.3.%
parent dev.pci.3.%parent: pcib3
```

d. Find the NIC's pcib location:

```
#> sysctl -a | grep pcib.3.%location

dev.pcib.3.%location: pci0:0:2:0 handle=\_SB_.PCI0.PEX2
```

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

e. 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.

a.Pin both interrupts and application processes to the relevant cores.

b. Find the closest NUMA to the NIC

c. Find the device interrupts.

```
vmstat -ia | grep mlx5_core0 | awk '{print $1}' | sed s/irq// | sed s/://
304
305
306
…
```

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

```
cpuset -x 304 -l 1
cpuset -x 305 -l 2
cpuset -x 306 -l 3
...
```

e. Bind the application to the closest NUMA cores list.
Note: It is 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.

Due to FreeBSD internal card memory allocation mechanism on boot, it is preferred to insert the NIC to a NUMA-0 slot for max performance.