SOFTWARE MANUAL

Mellanox DPDK

Quick Start Guide

Rev 16.11_4.0

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 PRODUCT(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

Table of Contents

List of Tables

Document Revision History

Table 1: Document Revision History

Revision	Date	Description

16.11_4.0	November 20th, 2017	• Added the following sections:
		  • VXLAN RSS and its subsections
		  • Hardware RX Checksum Offloads for VXLAN Packets without the Inner Ethernet Header and its subsections
		  • VF Representors and its subsections
		• Updated MLNX_OFED and Firmware versions to the latest across the document
		• Updated section mlx5 PMD Command Line Parameters: added the follow parameters:
		  • `inner_rss`
		• Removed ConnectX-3/ConnectX-3 Pro related sections.
		• Removed section ConnectX-4/ConnectX-4 Lx ESX6.0 UP1

16.11_3.0	June 30th, 2017	• Updated MLNX_OFED and Firmware versions to the latest across the document
		• Updated section mlx5 PMD Command Line Parameters, added the following parameters:
		  • delay_drop_rxq
		  • swp_en
		• Updated sections:
		  • System Requirements
		  • Command Line Requirements

16.11_2.3	April 06th, 2017	• Updated the supported MLNX-OFED/ESXi/Firmware versions across the document.
		• Updated section mlx5 PMD Command Line Parameters: added the follow parameters:
		  • `txq_mpw_hdr_dseg_en`
		  • `txq_max_inline_len`
		  • `tso`
		  • `udp_port`

16.11_1.5	January 31st, 2017	• Updated the supported MLNX-OFED/ESXi/Firmware versions across the document.
		• Updated the following sections:
		  • Configuring PMD Debug Mode
		  • ConnectX®-4 / ConnectX®-4 Lx / ConnectX®-5

2.2_4.2	October 09th, 2016	• Added MLNX_EN support and updated the following sections:
		  • MLNX_OFED/Firmware Versions
		  • ConnectX-4/ConnectX-4 Lx /ConnectX-5 Bare Metal
		  • MLNX_OFED/Firmware Versions
		  • **Error! Reference source not found.**
		• Added the following sections:

Revision	Date	Description
		• **Error! Reference source not found.** • ConnectX-4 / ConnectX-4 Lx /ConnectX-5 Settings • Updated MLNX_OFED/ESXi and ConnectX-4/Lx versions across the document • Updated the following sections: • Configuring PMD Debug Mode • mlx5 PMD Command Line Parameters
2.2_3.9	September 09th, 2016	• Updated MLNX_OFED version across the document • Updated section mlx5 PMD Command Line Parameters: added parameter `txq_inline_auto_en`
2.2_2.7	June 09th, 2016	• Added the following sections: • **Error! Not a valid result for table.** • **Error! Reference source not found.** • **Error! Reference source not found.** • Updated the following sections: • **Error! Reference source not found.** • **Error! Reference source not found.** • **Error! Reference source not found.** • Removed the following sections: • ConnectX-3/ConnectX-3 Pro ESX: VMWare • ConnectX-3/ConnectX-3 Pro ESX: VM Virtual Machine
2.2_2.3	May 16th, 2016	• Updated the following sections: • ConnectX-4/ConnectX-4 Lx /ConnectX-5 Bare Metal • **Error! Reference source not found.Error! Reference source not found.** • ConnectX-3/ConnectX-3 Pro ESX: VM Virtual Machine • ConnectX-4/ConnectX-4 Lx /ConnectX-5 Bare Metal • General Settings • ConnectX®-4 / ConnectX®-4 Lx • Removed the following section: • PowerKVM Settings

Related Documents

The following table lists the documents referenced in this User Manual.

Table 2: Reference Documents

Document Name	Description
Mellanox OFED Linux Release Notes	Describes the new features and changes of the latest MLNX_OFED release.
Mellanox OFED Linux User Manual	Provides general information on the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards.
Mellanox DPDK Release Notes	Describes the new features and changes of the latest Mellanox DPDK

1 Overview

Mellanox Poll Mode Driver (PMD) is designed for fast packet processing and low latency by providing kernel bypass for receive, send, and by avoiding the interrupt processing performance overhead.

This Quick Start Guide provides information of how to install and configure mlx5 DPDK Poll-Mode Driver (PMD) for Mellanox ConnectX®-4 onwards Ethernet adapters.

2 MLNX_DPDK Installation and Configuration

2.1 Installing ConnectX-4/ConnectX-4 Lx /ConnectX-5

2.1.1 MLNX_OFED/Firmware Versions

Table 3: MLNX_OFED/Firmware Versions

Driver/Firmware	Version	Download Link
MLNX_OFED	4.2-1.x.x.x	MLNX_OFED website
ConnectX-4 Firmware	12.21.1000	Firmware download page
ConnectX-4 Lx Firmware	14.21.1000	Firmware download page
ConnectX-5 Firmware	16.21.1000	Firmware download page

2.1.2 ConnectX-4/ConnectX-4 Lx /ConnectX-5 Bare Metal

➢ ***To clean installation or upgrade from any previous MLNX_DPDK GA version:***

1. Download MLNX_OFED. (See Table 3: MLNX_OFED/Firmware Versions for the Drivers/Firmware versions)
2. Install the MLNX_OFED driver.

```
./mlnxofedinstall
```

3. Verify that ConnectX-4 firmware is 12.21.1000, ConnectX-4 Lx firmware is 14.21.1000 and ConnectX-5 firmware is 16.21.1000

```
ibv_devinfo
```

4. Set all the ports to Ethernet if MLNX_OFED has been installed

```
mst start
mlxconfig -d <device> set LINK_TYPE_P1/2=1/2/3
* LINK_TYPE_P1=<1|2|3>, 1=Infiniband 2=Ethernet 3=VPI(auto-sense)
```

For example:

```
mlxconfig -d <device> set LINK_TYPE_P1=2
mlxfwreset -d <device> reset
```

For further instructions on how to run the script, please refer to the MLNX_OFED User Manual.

5. Restart the driver:

```
/etc/init.d/openibd restart
```

6. Extract the package MLNX_DPDK_16.11_4.0.tar.gz

The default mlx5 configuration in `config/common_base` is the following:

```
# Compile burst-oriented Mellanox ConnectX-4 (MLX5) PMD
#
CONFIG_RTE_LIBRTE_MLX5_PMD=y
CONFIG_RTE_LIBRTE_MLX5_DEBUG=n
CONFIG_RTE_LIBRTE_MLX5_TX_MP_CACHE=8
```

7. Compile DPDK.

```
On x86_64:
make install T=x86_64-native-linuxapp-gcc
on Power8:
```

```
make install T=ppc_64-power8-linuxapp-gcc
```

For more advanced DPDK compilation options, please refer to dpdk.org documentation:

http://www.dpdk.org/doc/guides-16.11/

8. Configure huge pages according to the NUMA the card is connected to.

```
echo $PAGE_NUM >
/sys/devices/system/node/node$NUMA_NODE/hugepages/hugepages-
"$PAGE_SIZE"kB/nr_hugepages

mkdir -p /mnt/huge

HTLB_MOUNTED=$( mount | grep "type hugetlbfs" | wc -l)
if [ $HTLB_MOUNTED -eq 0 ]; then
        mount -t hugetlbfs hugetlb /mnt/huge
fi
```

2.1.3 ConnectX-4/ConnectX-4 Lx / ConnectX-5 KVM

2.1.3.1 MLNX_OFED/Firmware Versions

Table 4: MLNX_OFED/MLNX_EN/Firmware Versions

Driver/Firmware	Version	Download Link
MLNX_OFED	4.2-1.x.x.x	MLNX_OFED website
ConnectX-4 Firmware	12.21.1000	Firmware download page
ConnectX-4 Lx Firmware	14.21.1000	Firmware download page
ConnectX-5 Firmware	16.21.1000	Firmware download page

2.1.3.2 ConnectX-4/ConnectX-4 Lx /ConnectX-5 KVM Hypervisor

1. Download MLNX_OFED. (See Table 3: MLNX_OFED/Firmware Versions for the Drivers/Firmware versions)
2. Install the MLNX_OFED driver.

```
./mlnxofedinstall
```

3. Restart the driver.

```
/etc/init.d/openibd restart
```

4. Verify that ConnectX-4 firmware is 12.21.1000, ConnectX-4 Lx firmware is 14.21.1000 and ConnectX-5 firmware is 16.21.1000

```
ibv_devinfo
```

5. Check if SR-IOV is enabled in the firmware.

```
mlxconfig -d <device> q
Device #1: ----------
Device type: Connect4
PCI device: /dev/mst/mt4115_pciconf0
Configurations: Current
SRIOV_EN 1
NUM_OF_VFS 8
```

6. If needed, use mlxconfig to set the relevant fields:

```
mlxconfig -d <device> set SRIOV_EN=1 NUM_OF_VFS=16
mlxfwreset -d <device> reset
```

7. Set all the ports to Ethernet.

```
mst start
mlxconfig -d <device> set LINK_TYPE_P1/2=1/2/3
* LINK_TYPE_P1=<1|2|3> , 1=Infiniband 2=Ethernet 3=VPI(auto-sense)
For example:
mlxconfig -d <device> set LINK_TYPE_P1=2
mlxfwreset -d <device> reset
```

For further instructions on how to run the script, please refer to the MLNX_OFED User Manual.

8. Write to the sysfs file the number of Virtual Functions you need to create for the PF.

   The following is an example of a standard Linux kernel generated file that is available in the new kernels.

```
echo [num_vfs] > /sys/class/infiniband/mlx5_0/device/sriov_numvfs
```

   For further information, please refer to the MLNX_OFED User Manual section "Configuring SR-IOV for ConnectX-4/Connect-IB".

2.1.3.3 ConnectX-4/ConnectX-4 Lx/ConnectX-5 KVM Virtual Machine

1. Download MLNX_OFED. (See Table 4: MLNX_OFED/MLNX_EN/Firmware Versions for the Drivers/Firmware versions)
2. Install the MLNX_OFED driver.

```
./mlnxofedinstall
```

3. Extract the package MLNX_DPDK-16.11_4.0.tar.gz

   The default mlx5 configuration in `config/common_base` is the following:

```
#
# Compile burst-oriented Mellanox ConnectX-4 (MLX5) PMD
#
CONFIG_RTE_LIBRTE_MLX5_PMD=y
CONFIG_RTE_LIBRTE_MLX5_DEBUG=n
CONFIG_RTE_LIBRTE_MLX5_TX_MP_CACHE=8
```

4. Compile DPDK.

```
On x86_64:
make install T=x86_64-native-linuxapp-gcc
on Power8:
make install T=ppc_64-power8-linuxapp-gcc
```

   For more advanced DPDK compilation options, please refer to dpdk.org documentation:

   http://www.dpdk.org/doc/guides-16.11/

5. Configure huge pages.

```
echo $PAGE_NUM > /sys/devices/system/node/node0/hugepages/hugepages-
"$PAGE_SIZE"kB/nr_hugepages

mkdir -p /mnt/huge

HTLB_MOUNTED=$( mount | grep "type hugetlbfs" | wc -l)
if [ $HTLB_MOUNTED -eq 0 ]; then
        mount -t hugetlbfs hugetlb /mnt/huge
fi
```

2.2 Configuring PMD Debug Mode

➢ ***To enable Debug mode:***

1. Modify the `config/common_base` file.
   - `CONFIG_RTE_LOG_LEVEL=RTE_LOG_DEBUG`
   - For mlx4 PMD: `CONFIG_RTE_LIBRTE_MLX4_DEBUG=y`
   - For mlx5 PMD: `CONFIG_RTE_LIBRTE_MLX5_DEBUG=y`
2. Compile DPDK.

```
On x86_64:
rm -rf x86_64-native-linuxapp-gcc
make install T=x86_64-native-linuxapp-gcc
On Power8:
rm -rf ppc_64-power8-linuxapp-gcc
make install T=ppc_64-power8-linuxapp-gcc
```

3. Add `--log-level=9` to EAL command line parameters when you run your application.

2.3 mlx5 PMD Command Line Parameters

Parameter	Description
txq_inline_auto_en	**[Beta]** This mode is disabled by default and supported only on ConnectX-4 single port card. It should not be used to measure 0 packet loss performance. When enabled, the running algorithm for auto inline is: • The PPS is measured with and without inline, and the decision is taken according to the highest PPC. • After the decision was taken, every 100 ms of the packet rate is tested and if there was 10% change in the packet rate, the measurements with and without inline are done again. It is expected that at the beginning of the test the performance will not be at maximum, but after 1 sec, the performance should be stabilized. When enabled, txq_inline option is ignored.
txq_inline	Amount of data to be inlined during TX operations. This improves latency. It can improve PPS performance when PCI back pressure is detected and may be useful for scenarios involving heavy traffic on many queues. On ConnectX-4: It is recommended to set it to 128 for messages <=128Band to set it to 64 for messages > 128B On ConnectX-4 Lx: When is used with txq_mpw_en option (see below), in case of PCI back pressure, it is recommended to set to 200 On ConnectX-5: It is set to 896 by default if Enhanced MPW is enabled. If not, it is disabled by default (set to 0) since the additional software logic necessary to handle this mode can lower performance when back pressure is not expected. (see below explanation regarding Enhanced MPW)
txqs_min_inline	Enable inline Send only when the number of TX queues is higher or equal to this value. This option should be used in combination with txq_inline above. It is set to 8 by default only if Enhanced MPW is enabled. If not, it is set to zero by default.
txq_mpw_en	When set to 1 MPW/Enhanced MPW is enabled When set to 0 MPW is disabled This option is to enable multi-packet send. This feature allows the TX burst function to pack up to multiple packets in a single descriptor session in order to save PCI bandwidth. MPW is enabled for ConnectX-4 Lx and Enhanced MPW is enabled for ConnectX-5. A MPW session can carry only identical size of packets and the number of packets in a session is also limited while Enhanced MPW does not have such limitations. When txq_inline is set along with this option, TX burst function tries to copy entire packet data on to TX descriptor instead of including pointer of packet only if there is enough room remained in the descriptor. txq_inline sets per-descriptor space for either pointers or inlined packets. Enhanced MPW supports hybrid mode - mixing inlined packets and pointers in a descriptor. This option cannot be used in conjunction with tso. When tso is set, txq_mpw_en is disabled. It is set to 1 by default. To disable please set to 0.

Parameter	Description
	When multi-packet send is enabled, there is no support for TX VLAN HW insertion. To use TX VLAN HW insertion please disable it.
txq_mpw_hdr_dseg_en	Enables including two pointers in the first block of TX descriptor. This can be used to lessen CPU load for memory copy. Effective only when Enhanced MPW is supported. Disabled by default.
txq_max_inline_len	Maximum size of packet to be inlined. This option limits the size of packet to be inlined. If the size of a packet is larger than configured value, the packet isn't inlined even though there's enough space remained in the descriptor. Instead, the packet is included with pointer. Effective only when Enhanced MPW is supported. The default value is 256.
tso	Enables hardware TSO. When hardware TSO is enabled, packets marked with TCP segmentation offload will be divided into segments by the hardware. It is disabled by default.
rxq_cqe_comp_en	A nonzero value that enables optimized received packets' completion algorithm when PCI back pressure is detected. This improves performance at the cost of a slightly higher CPU usage. It is enabled by default. To disable this please set to 0
udp_port	A nonzero value enables traffic dispatching between the kernel and DPDK according to the packet UDP port. UDP packets with destination port equals to udp_port will be directed to DPDK. RSS will be support for those packets, the RSS configuration must be ETH_RSS_UDP.
delay_drop_rxq	A nonzero value enables tolerance mode for rxq tail drop. In case of no available descriptors on the rxq, the NIC will tail drop packets. The options forces the NIC to check the descriptors count multiple times, before the drop decision. Thus providing the PMD the option to repost descriptions without packet loss. The feature is valuable for cases where the DPDK application wants to keep the Rx descriptors count small without experiencing packet lost. The feature should be used with care, as delaying the drop may cause backpressure to propagate toward the link. It is disabled by default.
swp_en	A nonzero value enables the SW parser feature. When SW parser is enabled, the packet headers location is determined according to the length fields provided in the mbuf. The feature is useful in case application wants to use checksum and TSO Tx offload for non-standard packets. Software parser is supported only for PFs. In case of VF, option is ignored. See section 2.4 for more details It is disabled by default.
inner_rss	Enables RSS based on the inner packet for standard and non-standard VXLAN.

2.4 TX checksum and TSO offload for VXLAN Packets without the Inner Ethernet Header.

The feature provides the ability for HW to compute the checksums and segment non-standard VXLAN packets which do not contain the inner Ethernet header.

2.4.1 System Requirements

- Must run on PF

2.4.2 Command Line Requirements

- For ConnectX-4 Lx, force txq_mpw_en=0
- Enable software parser swp_en=1
- If TSO is required, set tso=1

2.4.2.1 Using Checksum and TSO offload for VXLAN Packets without the Inner Ethernet Header on ConnectX-4/ConnectX-4 Lx

➢ ***To use the feature, the application should:***

1. Prepare mbuf to send with the non-standard VXLAN packet
2. Set PKT_TX_TUNNEL_VXLAN on the mbuf
3. Fill all mbuf header length fields correctly
4. Set the checksum and TSO flags according to the application needs.

2.5 VXLAN RSS

2.5.1 On the Inner Header

VXLAN RSS on the inner header is enabled by default and does not require any special settings.

2.5.1.1 Configuring the Software

- Set the udp_port.
- Enable the inner_rss.

Example:

```
./testpmd -n 4 -w 00:09.0,udp_port=4789,inner_rss=1 -w
00:0a.0,udp_port=4789,inner_rss=1 -- -i
```

2.5.2 On the Inner Header without Inner Ethernet Header VXLAN (non-standard)

This option is not enabled by default and requires the actions below.

2.5.2.1 Configuring the NIC

Step 1: Enable non-standard VXLAN.

```
mlxconfig -d <device> s IP_OVER_VXLAN_EN=1
mlxconfig -d <device> s IP_OVER_VXLAN_PORT=<port>
```

Step 2: Reset the driver.

```
mlxfwreset -d mlx5_0 reset
```

Example:

```
mlxconfig -d /dev/mst/mt4115_pciconf0 set IP_OVER_VXLAN_EN=1
mlxconfig -d /dev/mst/mt4115_pciconf0 set IP_OVER_VXLAN_PORT=<port_num>

mlxfwreset -d /dev/mst/mt4115_pciconf0 reset
```

2.5.2.2 Configuring the Software

- Set the udp_port.
- Enable the inner_rss.

Example:

```
./testpmd -n 4 -w 00:09.0,udp_port=250,inner_rss=1 -w
00:0a.0,udp_port=250,inner_rss=1 -- -i
```

2.6 Hardware RX Checksum Offloads for VXLAN Packets without the Inner Ethernet Header

2.6.1 Configuring the Software

- Set the udp_port.
- Enable the rx-cksum

Example:

```
./testpmd -n 4 -w 00:09.0,udp_port=250,swp_en=1 -w
0:0a.0,udp_port=250,swp_en=1 -- -I -enable-rx-cksum
```

2.7 VF Representors

The VF Representor is the port representor of the Virtual Function.

2.7.1 Configuring the System

- Enable the VR Representor identification in the system:

  ```
  echo switchdev > /sys/kernel/debug/mlx5/<MLX5 PCI>/compat/mod
  ```

- Disable the VR Representor:

  ```
  echo legacy > /sys/kernel/debug/mlx5/<MLX5 PCI>/compat/mod
  ```

NOTE: In ConnectX-4 Lx, the MPW should be disabled for the traffic to pass between the VF's representors.

2.7.2 Getting the DPDK Representor from the Port ID using testpmd

In testpmd, run the following command:

```
testpmd> show port info 0
```

If the port is the physical port, the port's output is as follow:

```
testpmd> show port info 0

********************* Infos for port 0  *********************
MAC address: EC:0D:9A:33:38:F7
Connect to socket: 0
Physical function: 0000:81:00.1 => mlx5_1 port 1
memory allocation on the socket: 0
Link status: up
Link speed: 25000 Mbps
Link duplex: full-duplex
Promiscuous mode: enabled
Allmulticast mode: disabled
Maximum number of MAC addresses: 128
Maximum number of MAC addresses of hash filtering: 0
VLAN offload:
  strip on
  filter on
  qinq(extend) off
Hash key size in bytes: 40
Redirection table size: 1
No flow type is supported.
Max possible RX queues: 65535
Max possible number of RXDs per queue: 65535
Min possible number of RXDs per queue: 0
RXDs number alignment: 1
Max possible TX queues: 65535
Max possible number of TXDs per queue: 65535
Min possible number of TXDs per queue: 0
TXDs number alignment: 1
```

If the port is the VF representor, the port's output is as follow:

```
Testpmd> show port info 0

********************* Infos for port 0  *********************
MAC address: EC:0D:9A:33:38:F7
Connect to socket: 0
Virtual Function: 0000:81:01.3 => mlx5_12 port 1
memory allocation on the socket: 0
Link status: up
Link speed: 25000 Mbps
Link duplex: full-duplex
Promiscuous mode: enabled
Allmulticast mode: disabled
Maximum number of MAC addresses: 128
Maximum number of MAC addresses of hash filtering: 0
VLAN offload:
  strip on
  filter on
  qinq(extend) off
Hash key size in bytes: 40
Redirection table size: 1
No flow type is supported.
Max possible RX queues: 65535
Max possible number of RXDs per queue: 65535
Min possible number of RXDs per queue: 0
RXDs number alignment: 1
Max possible TX queues: 65535
Max possible number of TXDs per queue: 65535
Min possible number of TXDs per queue: 0
TXDs number alignment: 1
```

2.7.3 Getting the DPDK Representor from the port ID using the "rte_eth_dev_get_ib_rep_vf" Function

Provide a string which can contain the PCI address of the VF.

The return value of the PCI is the VF PCI of which this port represents.

```
char pci[] = "0000:00:00.0";
rte_eth_dev_get_ib_rep_vf(port_id, pci);
if (strcmp(pci, "0000:00:00.0") != 0)
printf("\nVirtual Function: %s => %s", pci,
          rte_eth_devices[port_id].data->name);
```

3 System Performance Configuration

3.1 General Settings

- Use server with at least 4 memory channels, for better performance. In order to check the number of memory channels use:

```
# lshw -C memory
```

The output will show the server banks with their utilization

```
    *-bank:1
         description: DIMM Synchronous 2400 MHz (0.4 ns)
         product: 18ASF2G72PDZ-2G3B1
         vendor: 002C00B3002C
         physical id: 1
         serial: 15B8ED61
         slot: A2
         size: 16GiB
         width: 64 bits
         clock: 2400MHz (0.4ns)
    *-bank:2
         description: [empty]
         physical id: 2
         slot: A3
```

- Use the CPU near local NUMA node to which the PCIe adapter is connected, for better performance.

  For Virtual Machines (VM), verify that the right CPU and NUMA node are pinned for the VM according to the above. If possible, connect you NIC near NUMA node 0. Run `mst status -v` to identify the NUMA node to which the PCIe adapter is connected

```
mst status -v
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded
PCI devices:
------------
DEVICE_TYPE             MST                           PCI       RDMA
NET                     NUMA
ConnectX3Pro(rev:0)     /dev/mst/mt4103_pciconf0
ConnectX3Pro(rev:0)     /dev/mst/mt4103_pci_cr0       04:00.0   mlx4_0
net-eth4,net-eth5
```

- If more than one adapter is used, verify that both adapters are located on the same PCI bus (as each CPU socket on a Crown Pass platform manages its own PCI bus) in order to forward packets from one to the other without NUMA performance penalty.

- **Check the Core Frequency**

  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 the core frequencies are consistent

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

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

- 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
```

When the following CPU frequency modules are loaded, CPU scaling is enabled, and you can improve performance by setting the scaling mode to performance:

- **Set the scaling mode to performance for every CPU**

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

For further information, please refer to Performance Tuning Guide for Mellanox Adapters (http://www.mellanox.com/related-docs/prod_software/Performance_Tuning_Guide_for_Mellanox_Network_Adapters.pdf)

- **Verify that Max_Read_Req BIOS parameter is set to 1K**

To obtain the current setting for the Max_Read_Req BIOS parameter:

```
setpci -s <NIC BIOS address> 68.w

example:
setpci -s 21:00.0 68.w
3026
```

If the output is different than 3XXX, set it by:

```
setpci -s <NIC BIOS address> 68.w=3XXX
```

NOTE: The XXX can be different on different systems. Make sure to configure according to the setpci output..

For example:

```
setpci -s 84:00.0 68.w
2026
Run: setpci -s 84:00.0 68.w=3026
```

- **Disable pause frames on all network interfaces managed by mlx4_en/mlx5_en**

```
lldpad stop
```

```
ethtool -A eth16 rx off tx off
```

NOTE: In certain systems, pause frames are used to increase performance.

- **Use 1Gb huge pages**
- **Hyper threading**

In certain DPDK application, enabling hyper threading results in better performance. For benchmarking purposes, it is recommended to disable hyper threading

- **Make sure that unnecessary System Management Interrupts (SMIs) are disabled**

SMI that are used for Power Monitoring and for Memory PreFailure Notification are recommended to be disabled. Please refer to your server provider guides for recommended platform tuning.

- **Isolate used cores**

  Use `isolcpus` command for boot configuration.

  For example, add the following to kernel boot parameters:

```
isolcpus=2,3
```

- **Disable kernel memory compaction**

```
echo never > /sys/kernel/mm/transparent_hugepage/defrag
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo 0 > /sys/kernel/mm/transparent_hugepage/khugepaged/defrag
sysctl -w vm.swappiness=0
sysctl -w vm.zone_reclaim_mode=0
```

- **Interrupts configuration**

  NOTE: Interrupts configuration should be performed only if the needed performance was not achieved**.**

  - Stop irqbalancer

```
service irqbalance stop
```

  - Set all possible interrupts to different NUMA:

```
Example: echo '6-9' | sudo tee /proc/irq/*/smp_affinity_list
```

  - Set NIC interrupts to same NUMA:

    Example:

```
set_irq_affinity_cpulist.sh 0-1 ethX
```

  - Set other NIC interrupts to different NUMA:

    Example:

```
set_irq_affinity_cpulist.sh 6-9 ethY
```

- **IO non-posted prefetch settings**

  Verify IO non-posted prefetch is disabled by default. This can be checked via the BIOS configuration.

  Please contact you server provider for more information about the settings.

3.1.1 ConnectX-4 / ConnectX-4 Lx /ConnectX-5 Settings

- Configure aggressive CQE Zipping for maximum performance

```
mlxconfig -d mlx5_0 s CQE_COMPRESSION=1
```

- To set it back to the default CQE Zipping mode

```
mlxconfig -d mlx5_0 s CQE_COMPRESSION=0
```

- x86_64

  Use command line parameter *txq_inline* and *txqs_min_inline* to get maximum performance**:**

  - Messages sizes <=128B: txq_inline=128, txqs_min_inline=4
  - Message sizes > 128: txq_inline=64, txqs_min_inline=4

- ConnectX-4 LX: txq_inline=200, txqs_min_inline=4
- ConnectX-5: use default values

  Example:

```
./testpmd -c 0x1ff -n 4 -w
0000:08:00.0,txq_inline=128,txqs_min_inline=4  -w
0000:08:00.1,txq_inline=128,txqs_min_inline=4  --socket-mem=2048,0 --
--port-numa-config=0,0,1,0 --socket-num=0 --burst=64 --txd=1024 --
rxd=256 --mbcache=512 --rxq=4 --txq=4 --nb-cores=8 -i
```

3.2 KVM Settings

1. Make sure that Hypervisor kernel is 3.16 or newer (For example Ubuntu 14.10 or Fedora 20/21 can be used).
2. Configure boot with "`iommu=pt`".
3. Use 1G huge pages.
4. Make sure to allocate a VM on huge pages

   Example:

   qemu is started with the following commands:

```
umount /mnt/huge 2> /dev/null
mount -t hugetlbfs none /mnt/huge &&
echo 8192 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages &&

numactl --cpunodebind 1 --membind 1 -- \
qemu-system-x86_64 \
-smp 24 \
-m 4G \
-mem-path /mnt/huge \
-mem-prealloc \
-enable-kvm \
-cpu host \
-serial tcp::999,server,nowait \
-nographic \
-vga none \
- -device pci-assign,host=83:00.1 \
-device pci-assign,host=84:00.1 \
-drive snapshot=on,file=/opt/vm/ubuntu-14.04-template.qcow2 \
-drive file=/data/data.img
```

   Since both adapters are installed on NUMA node 1 PCI slots (CPU socket 1), `*numactl` is used to bind qemu to CPU threads and memory from that node only, which makes a Virtual Machine without NUMA internally.

   All its memory is allocated from huge pages in /mnt/huge.

   - After loading VM, verify huge pages on your Hypervisor is used by VM:

```
cat /sys/devices/system/node/node<NUM>/hugepages/hugepages-<PAGE-
SIZE>/free_hugepages
```

- Make sure to set CPU pining

  For example if you run qemu:

```
(qemu) info cpus
CPU #0: pc=0xffffffff81056306 (halted) thread_id=2719
CPU #1: pc=0xffffffff81056306 (halted) thread_id=2720
CPU #2: pc=0xffffffff81056306 (halted) thread_id=2723
CPU #3: pc=0xffffffff81056306 (halted) thread_id=2724
```

```
taskset -p 0x1 2719
taskset -p 0x2 2720
taskset -p 0x4 2723
taskset -p 0x8 2724
```

- It is recommended to use small TX and RX queues, the number of descriptors should be <= 512

4 Running DPDK Application with Mellanox Poll-Mode Driver

Mellanox NICs must not be unbound from the kernel. Although the control path is not through uio/vfio, it still uses the existing kernel drivers while the data path bypasses kernel.

4.1 ConnectX®-4 / ConnectX®-4 Lx / ConnectX®-5

For example:

Specific PCI address can be used to specify NIC's ports:

```
./testpmd -c 0x1ff -n 4 -w 0000:08:00.0
-w 0000:08:001   --socket-mem=2048,0 -- --port-numa-config=0,0,1,0 --socket-
num=0 --burst=64 --txd=1024 --rxd=256 --mbcache=512 --rxq=4 --txq=4 --nb-
cores=8   -i
```

NOTE: In ConnectX-4 NICs, a PCI address represents each port.

For best performance with ConnectX-4 NIC:

```
./testpmd -c 0x1ff -n 4 -w 0000:08:00.0,txq_inline=128
-w 0000:08:00.1,txq_inline=128  --socket-mem=2048,0 -- --port-numa-
config=0,0,1,0 --socket-num=0 --burst=64 --txd=1024 --rxd=256 --mbcache=512
--rxq=4 --txq=4 --nb-cores=8  --rss-udp -i
```

NOTE: ConnectX-4: It is recommended to use **`txq_inline=128`** parameter for best performance for 64B and 128B messages with more multiple cores to achive max perfromance.

NOTE: ConnectX-4: It is recommended to use **`txq_inline=64`** parameter for best performance for messages >= 256B with multiple cores to achive max perfromance.

NOTE: `--rss-udp testpmd` option should be used to achieve best spreading of UDP flows.

Some ConnectX-4 Lx cards are single port cards. To run testpmd fwd test on one port:

```
./testpmd -c 0xe000 -n 4 -w 0000:08:00.0,txq_inline=200    --socket-
mem=2048,0 -- --port-numa-config=0,0,1,0 --socket-num=0 --burst=64 --
txd=4096 --rxd=1024 --mbcache=512 --rxq=4 --txq=4 --nb-cores=4 --rss-udp --i
```

NOTE: ConnectX-4 LX: It is recommended to use **`txq_inline=200`** parameter for best performance for 64B messages with multiple cores to achive maximum perfromance

NOTE: `--rss-udp testpmd` option should be used to achieve best spreading of UDP flows.

For ConnectX-5 cards, default values will bring the best performance:

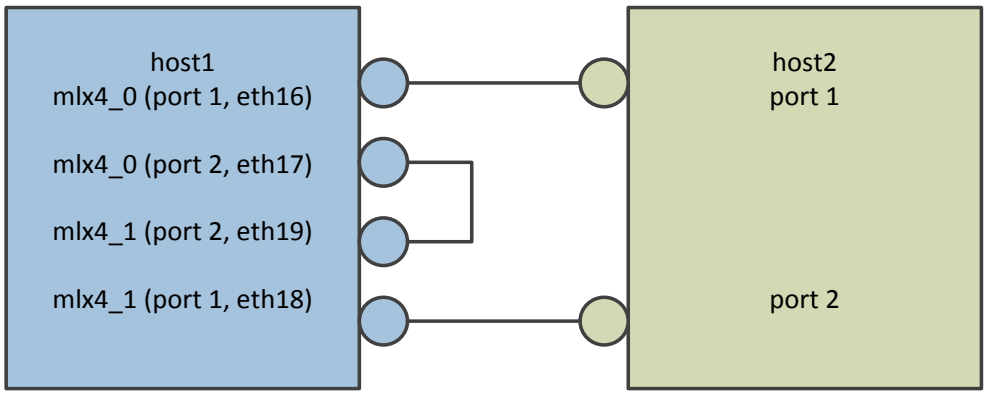
```
./testpmd -c 0xff00ff00 -n 4 -w 84:00.0 --socket-mem=0,8192 -- --port-numa-
config=0,1 --socket-num=1 --burst=128 --txd=512 --rxd=512 --mbcache=512 –
rxq=15 --txq=15 --nb-cores=15 -i --rss-udp -a
```

5 Sanity Check

Provided that all software components have been successfully installed and at least one ConnectX® adapter is present in the host system, run testpmd to test PMD.

These examples assume that there is a host with two dual port adapters that:

- First port of each NIC is linked to another similar host
- Second port of each NIC is linked with each other



Run *testpmd* interactively from the DPDK build tree (for more information about its command-line options, please refer to its documentation: http://www.dpdk.org/doc/guides/testpmd_app_ug/index.html):

```
 root# ~/DPDK/x86_64-native-linuxapp-gcc/app/test-pmd/testpmd -c
0xf000f000 -n 4 -d -- -i
 EAL: Detected lcore 0 as core 0 on socket 0
 EAL: Detected lcore 1 as core 1 on socket 0
 EAL: Detected lcore 2 as core 2 on socket 0
 EAL: Detected lcore 3 as core 3 on socket 0
 EAL: Detected lcore 4 as core 4 on socket 0
 [...]
 EAL: Detected lcore 27 as core 3 on socket 1
 EAL: Detected lcore 28 as core 4 on socket 1
 EAL: Detected lcore 29 as core 5 on socket 1
 EAL: Detected lcore 30 as core 6 on socket 1
 EAL: Detected lcore 31 as core 7 on socket 1
 EAL: Support maximum 64 logical core(s) by configuration.
 EAL: Detected 32 lcore(s)
 EAL:   cannot open VFIO container, error 2 (No such file or directory)
 EAL: VFIO support could not be initialized
 EAL: Setting up memory...
 EAL: Ask a virtual area of 0x6400000 bytes
 EAL: Virtual area found at 0x7f15fd600000 (size = 0x6400000)
 EAL: Ask a virtual area of 0x200000 bytes
[...]
 EAL: PCI device 0000:83:00.0 on NUMA socket 1
 EAL:   probe driver: 15b3:1007 librte_pmd_mlx4
 PMD: librte_pmd_mlx4: PCI information matches, using device "mlx4_0"
(VF: false)
 PMD: librte_pmd_mlx4: 2 port(s) detected
 PMD: librte_pmd_mlx4: bad state for port 1: "down" (1)
 PMD: librte_pmd_mlx4: port 1 MAC address is 00:02:c9:b5:b7:50
 PMD: librte_pmd_mlx4: bad state for port 2: "down" (1)
 PMD: librte_pmd_mlx4: port 2 MAC address is 00:02:c9:b5:b7:51
 EAL: PCI device 0000:84:00.0 on NUMA socket 1
 EAL:   probe driver: 15b3:1007 librte_pmd_mlx4
```

```
 PMD: librte_pmd_mlx4: PCI information matches, using device "mlx4_1"
(VF: false)
 PMD: librte_pmd_mlx4: 2 port(s) detected
 PMD: librte_pmd_mlx4: bad state for port 1: "down" (1)
 PMD: librte_pmd_mlx4: port 1 MAC address is 00:02:c9:b5:ba:b0
 PMD: librte_pmd_mlx4: bad state for port 2: "down" (1)
 PMD: librte_pmd_mlx4: port 2 MAC address is 00:02:c9:b5:ba:b1
 Interactive-mode selected
 Configuring Port 0 (socket 0)
 PMD: librte_pmd_mlx4: 0x7f35e0: TX queues number update: 0 -> 1
 PMD: librte_pmd_mlx4: 0x7f35e0: RX queues number update: 0 -> 1
 Port 0: 00:02:C9:B5:B7:50
 Configuring Port 1 (socket 0)
 PMD: librte_pmd_mlx4: 0x7f3620: TX queues number update: 0 -> 1
 PMD: librte_pmd_mlx4: 0x7f3620: RX queues number update: 0 -> 1
 Port 1: 00:02:C9:B5:B7:51
 Configuring Port 2 (socket 0)
 PMD: librte_pmd_mlx4: 0x7f3660: TX queues number update: 0 -> 1
 PMD: librte_pmd_mlx4: 0x7f3660: RX queues number update: 0 -> 1
 Port 2: 00:02:C9:B5:BA:B0
 Configuring Port 3 (socket 0)
 PMD: librte_pmd_mlx4: 0x7f36a0: TX queues number update: 0 -> 1
 PMD: librte_pmd_mlx4: 0x7f36a0: RX queues number update: 0 -> 1
 Port 3: 00:02:C9:B5:BA:B1
 Checking link statuses...
 Port 0 Link Up - speed 10000 Mbps - full-duplex
 Port 1 Link Up - speed 40000 Mbps - full-duplex
 Port 2 Link Up - speed 10000 Mbps - full-duplex
 Port 3 Link Up - speed 40000 Mbps - full-duplex
 Done
 testpmd>
```

The following commands are typed from the *testpmd* interactive prompt.

1. Check port status:

```
testpmd> show port info all
******************** Infos for port 0  ********************
   MAC address: 00:02:C9:B5:B7:50
   Connect to socket: 0
   memory allocation on the socket: 0
   Link status: up
   Link speed: 10000 Mbps
   Link duplex: full-duplex
   Promiscuous mode: enabled
   Allmulticast mode: disabled
   Maximum number of MAC addresses: 128
   Maximum number of MAC addresses of hash filtering: 0
   VLAN offload:
     strip on
     filter on
     qinq(extend) off

   ******************** Infos for port 1  ********************
   MAC address: 00:02:C9:B5:B7:51
   Connect to socket: 0
   memory allocation on the socket: 0
   Link status: up
   Link speed: 40000 Mbps
   Link duplex: full-duplex
   Promiscuous mode: enabled
   Allmulticast mode: disabled
   Maximum number of MAC addresses: 128
   Maximum number of MAC addresses of hash filtering: 0
   VLAN offload:
     strip on
     filter on
```

```
    qinq(extend) off

  ******************** Infos for port 2  ********************
  MAC address: 00:02:C9:B5:BA:B0
  Connect to socket: 0
  memory allocation on the socket: 0
  Link status: up
  Link speed: 10000 Mbps
  Link duplex: full-duplex
  Promiscuous mode: enabled
  Allmulticast mode: disabled
  Maximum number of MAC addresses: 128
  Maximum number of MAC addresses of hash filtering: 0
  VLAN offload:
    strip on
    filter on
    qinq(extend) off

  ******************** Infos for port 3  ********************
  MAC address: 00:02:C9:B5:BA:B1
  Connect to socket: 0
  memory allocation on the socket: 0
  Link status: up
  Link speed: 40000 Mbps
  Link duplex: full-duplex
  Promiscuous mode: enabled
  Allmulticast mode: disabled
  Maximum number of MAC addresses: 128
  Maximum number of MAC addresses of hash filtering: 0
  VLAN offload:
    strip on
    filter on
    qinq(extend) off
  testpmd>
```

2. Start IO forwarding between ports 1 and 3. The *tx_first* argument tells *testpmd* to send a single packet burst which will be always forwarded by both ports:

```
testpmd> set fwd io
   Set io packet forwarding mode
   testpmd> set portlist 1,3
   previous number of forwarding ports 4 - changed to number of
configured ports 2
   testpmd> start tx_first
     io packet forwarding - CRC stripping disabled - packets/burst=32
     nb forwarding cores=1 - nb forwarding ports=2
     RX queues=1 - RX desc=128 - RX free threshold=0
     RX threshold registers: pthresh=8 hthresh=8 wthresh=0
     TX queues=1 - TX desc=512 - TX free threshold=0
     TX threshold registers: pthresh=32 hthresh=0 wthresh=0
     TX RS bit threshold=0 - TXQ flags=0x0
   testpmd>
```

3. Display *testpmd* port statistics:

```
testpmd> show port stats all

     ######################## NIC statistics for port 0  ########################
     RX-packets: 0          RX-missed: 0          RX-bytes:  0
     RX-badcrc:  0          RX-badlen: 0          RX-errors: 0
     RX-nombuf:  0
     TX-packets: 0          TX-errors: 0          TX-bytes:  0
     ############################################################################

     ######################## NIC statistics for port 1  ########################
     RX-packets: 60800584   RX-missed: 0          RX-bytes:  3891239534
     RX-badcrc:  0          RX-badlen: 0          RX-errors: 0
     RX-nombuf:  0
     TX-packets: 61146609   TX-errors: 0          TX-bytes:  3913382976
```

```
  ############################################################################

  ######################## NIC statistics for port 2  ########################
  RX-packets: 0          RX-missed: 0          RX-bytes:  0
  RX-badcrc:  0          RX-badlen: 0          RX-errors: 0
  RX-nombuf:  0
  TX-packets: 0          TX-errors: 0          TX-bytes:  0
  ############################################################################

  ######################## NIC statistics for port 3  ########################
  RX-packets: 61146920   RX-missed: 0          RX-bytes:  3913402990
  RX-badcrc:  0          RX-badlen: 0          RX-errors: 0
  RX-nombuf:  0
  TX-packets: 60800953   TX-errors: 0          TX-bytes:  3891262080
  ############################################################################
testpmd>
```

4. Stop forwarding:

```
 testpmd> stop
Telling cores to stop...
Waiting for lcores to finish...

  ---------------------- Forward statistics for port 1  ----------------------
  RX-packets: 78238689       RX-dropped: 0             RX-total: 78238689
  TX-packets: 78681769       TX-dropped: 0             TX-total: 78681769
  ----------------------------------------------------------------------------

  ---------------------- Forward statistics for port 3  ----------------------
  RX-packets: 78681737       RX-dropped: 0             RX-total: 78681737
  TX-packets: 78238721       TX-dropped: 0             TX-total: 78238721
  ----------------------------------------------------------------------------

  ++++++++++++++++ Accumulated forward statistics for all ports++++++++++++++++
  RX-packets: 156920426      RX-dropped: 0             RX-total: 156920426
  TX-packets: 156920490      TX-dropped: 0             TX-total: 156920490
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Done.
testpmd>
```

5. Exit testpmd.

```
testpmd> quit
Stopping port 0...done
Stopping port 1...done
Stopping port 2...done
Stopping port 3...done
bye...
root#
```