Performance Tuning Guidelines

for Mellanox Network Adapters

Revision 1.16

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

Document Revision History **6**

1 Introduction **9**

1.1 Relevant Mellanox Drivers 9

2 General System Configurations **10**

2.1 PCI Express (PCIe) Capabilities 10

2.2 Memory Configuration 10

2.3 System Monitoring and Profilers 10

2.4 Recommended BIOS Settings 11

2.4.1 General 11

2.4.2 Intel® Haswell Processors 11

2.4.3 Intel® Sandy Bridge Processors / Ivy Bridge Processors 11

2.4.4 Intel® Nehalem/Westmere Processors 12

2.4.5 AMD Processors 12

3 Performance Tuning for Linux **14**

3.1 IRQ Affinity 14

3.1.1 IRQ Affinity Hints 14

3.1.2 IRQ Affinity Configuration 14

3.1.3 Auto Tuning Utility 15

3.1.4 Tuning for Multiple Adapters 15

3.2 ConnectX-4 100GbE Tuning 16

3.3 Power Management Tuning 16

3.3.1 OS Controlled Power Management 16

3.3.2 Checking Core Frequency 17

3.3.3 Setting the Scaling Governor 17

3.3.4 Kernel Idle Loop Tuning 17

3.4 NUMA Architecture Tuning 18

3.4.1 Tuning for Intel® Sandy Bridge Platform / Ivy Bridge Processors 18

3.4.2 Tuning for AMD® Architecture 19

3.4.3 Recognizing NUMA Node Cores 19

3.4.4 Running an Application on a Certain NUMA Node 19

3.5 Interrupt Moderation Tuning 19

3.6 Multi-Threaded IP Forwarding Tuning 20

3.7 Receive Side Scaling (RSS) 21

3.7.1 RSS Hash tuning 21

3.7.2 ConnectX®-3/Connect-X® 3 Pro Optimized Steering 21
3.8 Receive Packet Steering (RPS) 21
3.9 Tuning with sysctl 22
3.9.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance 22
3.9.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance 22
3.9.3 Preserving Your sysctl Settings after a Reboot 23
3.10 Verbs Applications Optimization 23
3.10.1 Single Thread Applications 23
3.11 Performance Tuning for iSER 23
3.12 Tuning VMA Parameters 24
3.12.1 Memory Allocation Type 24
3.12.2 Reducing Memory Footprint 24
3.12.3 Polling Configurations 25
3.12.4 Handling Single-Threaded Processes 25
3.13 Performance Tuning for Virtualized Environment 25
3.13.1 Tuning for Hypervisor 25
4 Performance Tuning for Windows 27
4.1 Tuning the Network Adapter 27
4.2 Tuning for NUMA Architecture 29
4.2.1 Tuning for Intel® Microarchitecture Code name Sandy Bridge / Ivy Bridge Platforms 29
4.2.2 Tuning for AMD® Architecture 29
4.2.3 Running an Application on a Certain NUMA Node 29
4.3 Tuning for Windows Server 2012 / 2012 R2 29
4.3.1 Recognizing NUMA Node Cores 29
4.4 Finding the Closest NUMA Node to the NIC 30
4.5 Tuning for Windows 2008 R2 30
4.5.1 Tuning for Multiple Adapters 31
4.5.2 Recognizing NUMA Node Cores 31
4.6 Performance Testing 32

List of Tables

Table 1: Document Revision History 6
Table 2: Recommended PCIe Configuration 10

Document Revision History

Table 1: Document Revision History

Revision	Date	Description
1.16	December, 2015	• Added a note on the recommended CPU architectures to achieve maximum performance in section ConnectX-4 100GbE Tuning.
	November, 2015	• Added section ConnectX-4 100GbE Tuning
1.15	May, 2015	• Added section Intel® Haswell Processors
1.14	January, 2015	• Added section System Monitoring and Profilers
1.13	September, 2014	• Removed section Multi Thread Applications

	August, 2014	• Added the following sections:
		  • Performance Tuning for iSER
		• Updated the following sections:
		  • Tuning the Network Adapter
		  • Performance Testing
		  • IRQ Affinity Hints
		  • Multi-Threaded IP Forwarding Tuning
		  • ConnectX®-3/Connect-X® 3 Pro Optimized Steering

1.12	May, 2014	• Added the following sections:
		  • IRQ Affinity Hints
		  • Receive Side Scaling (RSS) and its subsections
		  • Receive Packet Steering (RPS)
		• Updated the following sections:
		  • IRQ Affinity Configuration
		  • OS Controlled Power Management
		  • Setting the Scaling Governor
		  • Verbs Applications Optimization

1.11	March, 2014	• Updated Tuning the Network Adapter for Improved IPv4 Traffic Performance

	February, 2014	• Updated the following sections:
		  • Relevant Mellanox Drivers
		  • Intel® Sandy Bridge Processors / Ivy Bridge Processors
		  • Setting the Scaling Governor
		  • Kernel Idle Loop Tuning
		  • Interrupt Moderation Tuning
		  • Tuning for Intel® Sandy Bridge Platform / Ivy Bridge Processors
		  • Improving Application Performance on Remote Sandy Bridge Node
		  • Auto Tuning Utility
		  • Multi-Threaded IP Forwarding Tuning

Revision	Date	Description
		• Memory Allocation Type • Reducing Memory Footprint • Polling Configurations • Tuning for Intel® Microarchitecture Code name Sandy Bridge / Ivy Bridge Platforms • Tuning for Windows Server 2012 / 2012 R2 • Removed the following section: • *Reducing DMAs* • Added the following section: • Verbs Applications
1.10	December, 2013	• Updated section Performance Testing
	October, 2013	• Updated section Kernel Idle Loop Tuning • Added section Performance Tuning for Virtualized Environment
1.9	September, 2013	• Updated section Interrupt Moderation
1.8	June, 2013	• Removed section *Tuning for Windows Server 2008* and its sub-sections • Added the following sections: • Recognizing NUMA Node Cores • Finding the Closest NUMA Node to the NIC
1.7	April, 2013	• Updated the following sections: • Recommended BIOS Settings • Power Management • Tuning for Intel® Sandy Bridge • IRQ Affinity Configuration • Multi-Threaded IP Forwarding Tuning • Tuning for Multiple Adapters • Replaced "Tuning for IPoIB Interfaces" with Auto Tuning Utility • Added section Improving Application Performance on Remote Sandy Bridge Node
1.6	October, 2012	• Added the following sections: • Recognizing NUMA Node Cores • Running an Application on a Certain NUMA Node • Recognizing NUMA Node Cores • Updated the following sections: • Tuning the Network Adapter
1.5	May, 2012	• Added the following sections: • Memory Configuration • Tuning for IPoIB/EoIB Interfaces

Revision	Date	Description

		• Kernel Idle Loop Tuning
		• Updated the following sections:
		  • IRQ Affinity Configuration
		  • Recommended BIOS Settings
		  • Tuning for Multiple Adapters
		  • Tuning for Windows 2008 R2

1.4	April, 2012	• Added "Tuning for NUMA Architecture" sections. • Rearranged section in chapter 3.
1.3	March, 2012	• Added new section "Tuning Power Management".
1.2	January, 2012	• Updated versions of adapters to make the document more generic. • Merged sections on BIOS Power Management Settings and Intel Hyper-Threading Technology to new section, "Recommended BIOS Settings". • Added sections to "Performing Tuning for Linux". • Added section, "Tuning for Windows 2008 R2". • Added new chapter, "Tuning VMA Parameters".

1.1		• Updated the following sections:
		  • "Intel® Hyper-Threading Technology"
		  • "Tuning the Network Adapter for Improved IPv4 Traffic Performance"
		  • "Example: Script for Setting Interrupt Affinity"
		• Added new section, "Tuning IP Forwarding".

1 Introduction

Depending on the application of the user's system, it may be necessary to modify the default configuration of network adapters based on the ConnectX® adapters. This document describes important tuning parameters and settings that can improve performance for Mellanox drivers. Each setting, along with its potential effect, is described to help in making an informed judgment concerning its relevance to the user's system, the system workload, and the performance goals.

Tuning is relevant for both Ethernet and IPoIB network interfaces.

1.1 Relevant Mellanox Drivers

The tuning guidelines described in this document apply to the following Mellanox Software drivers:

- On *Linux*: Mellanox Ethernet Driver *MLNX_EN for Linux* version 2.x and later
- On Linux: Mellanox VPI Driver *MLNX_OFED for Linux* version 2.x and later
- On *Windows*: Mellanox OFED for Windows *MLNX_VPI* version 4.80 and later

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.

2.1 PCI Express (PCIe) Capabilities

Table 2: Recommended PCIe Configuration

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

Note: 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.

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.

2.3 System Monitoring and Profilers

It is recommended to disable system profilers and/or monitoring tools while running performance benchmarks. System profilers and/or monitoring tools use the host’s resources, hence running them in parallel to benchmark jobs may affect the performance in various degrees based on the traffic type and/or pattern, and the nature of the benchmark.

In order to measure optimal performance, make sure to stop all system profilers and monitoring tools (such as sysstat, vmstat, iostat, mpstat, dstat, etc.), before running any benchmark tool.

2.4 Recommended BIOS Settings

Note: These performance optimizations may result in higher power consumption.

2.4.1 General

Set BIOS power management to **Maximum Performance**.

2.4.2 Intel® Haswell Processors

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

BIOS Option		Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Enabled
	Hyper-Threading	HPC: disabled Data Centers: enabled
	IO non posted prefetching	Enabled (If the BIOS option does not exists, please contact your BIOS vendor)
	CPU frequency select	Max performance
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

2.4.3 Intel® Sandy Bridge Processors / Ivy Bridge Processors

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

BIOS Option		Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Enabled
	Hyper-Threading	HPC: disabled Data Centers: enabled
	CPU frequency select	Max performance

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

2.4.4 Intel® Nehalem/Westmere Processors

The following table displays the recommended BIOS settings in machines with Intel Nehalem-based processors.

BIOS Option		Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Disabled
	Hyper-Threading[1]	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

2.4.5 AMD Processors

The following table displays the recommended BIOS settings in machines with AMD based 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
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA

[1] 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.

BIOS Option		Values
	Channel Interleaving	Enabled
	Thermal Mode	Performance

3 Performance Tuning for Linux

3.1 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.

3.1.1 IRQ Affinity Hints

As of MLNX_OFED-2.2-1.x-x, the driver uses affinity hints API that allows the irqbalance service to set the affinity automatically. On some kernels the irqbalance service needs to be restarted in order for these changes to take effect.

To check if affinity hints is working properly run the following command at least 10 seconds after the interface is up:

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

If all the rows are "`fffff`" or "`00000`", it means it did not work and the irqbalance needs to be restarted.

3.1.2 IRQ Affinity Configuration

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

For optimal functionality it is recommended to download the latest tuning scripts from the web:

```
cd /tmp ; wget http://www.mellanox.com/related-
docs/prod_software/mlnx_irq_affinity.tgz ; tar xzf
/tmp/mlnx_irq_affinity.tgz --directory=/usr/sbin/ --overwrite
```

For systems that have Sandy Bridge, Ivy Bridge or AMD CPUs 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>
```

3.1.3 Auto Tuning Utility

MLNX_OFED 2.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. This tool will disable the IRQ balancer service from running at boot time. To disable it immediately, need to stop the service manually (`service irqbalance stop`).

Usage:

- Start

```
# service irqbalance stop
# mlnx_affinity start
```

- Stop

```
# mlnx_affinity stop
# service irqbalance start
```

- 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
```

Note: This tool is not a service, it run once and exits.

3.1.4 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
```

3.2 ConnectX-4 100GbE Tuning

Line-rate performance with ConnectX-4 100GbE can be achieved by most operation systems without special tuning. The number of streams needed varies from 4 to 16, depending on the system strength and OS/kernel.

Note: It is recommended to use systems with IvyBridge or Haswell CPUs for achieving maximum performance. For further information, please refer to section 2.4: Recommended BIOS Settings.

In some Linux distributions, Hardware LRO (HW LRO) must be enabled to reach the required line-rate performance.

- ***To enabled HW LRO:***

```
# ethtool --set-priv-flags <interface>  hw_lro on ( default off)
```

In case "`tx-nocache-copy`" is enabled, (this is the case for some kernels, e.g. kernel 3.10, which is the default for RH7.0) "`tx-nocache-copy`" should be disabled.

- ***To disable "tx-nocache-copy":***

```
# ethtool -K <interface>  tx-nocache-copy off
```

3.3 Power Management Tuning

3.3.1 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.

There are several options to resolve:

- When using MLNX_OFED-2.2-x.x.x or higher , Ethernet interfaces can be configured to automatically request for low latency from the OS

  This can be done using ethtool:

```
# ethtool --set-priv-flags <interface>  pm_qos_request_low_latency on (
default off)
```

  This is to improve latency and packet loss while power consumption can remain low when traffic is idle

- When using IPoIB or an older driver, it is possible to force high power by kernel parameters:

  a. Edit the /boot/grub/grub.conf file or any other bootloader configuration file.

  b. Add the following kernel parameters to the bootloader command.

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

  c. Reboot the system.

  Example:

```
title RH6.4x64
root (hd0,0)
kernel /vmlinuz-RH6.4x64-2.6.32-358.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
```

- Temporarily request for low CPU latency from user mode. This can be done by a program that opens /dev/cpu_dma_latency and writing the required latency, while keeping the file descriptor opened.

  For further information, please refer to kernel documents:

  Linux/Documentation/power/pm_qos_interface.txt

3.3.2 Checking 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 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 section Recommended BIOS Settings (on page 10) 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
```

3.3.3 Setting the Scaling Governor

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

➢ ***To set the scaling mode to performance, use this command for every cpu:***

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

3.3.4 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 wakeup time but may result in higher power consumption.

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

Please be aware that if the file does not exist, is must be created having the same name as the one stated above.

- For MLNX_OFED 2.x

```
options mlx4_core enable_sys_tune=1
```

3.4 NUMA Architecture Tuning

3.4.1 Tuning for Intel® Sandy Bridge Platform / Ivy Bridge Processors

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
```

3.4.1.1 Improving Application Performance on Remote Sandy Bridge Node

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

libmlx4 and libmlx5 have a build-in enhancement that recognizes an application that is pinned to a remote Sandy Bridge node and activates a flow that improves the out-of-the-box latency and throughput.

However, the Sandy Bridge node recognition must be enabled as described in section 3.4.1.

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 Sandy Bridge node which its cores are 0-7:

- When using ConnectX®-3 adapter cards:

```
MLX4_LOCAL_CPUS=0xff
```

- When using Connect-IB® adapter cards:

```
MLX5_LOCAL_CPUS=0xff
```

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

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

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

3.4.2 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).

3.4.3 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
```

3.4.4 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
```

3.5 Interrupt Moderation Tuning

Note: This section applies to both Ethernet and IPoIB interfaces.

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: 100000
pkt-rate-high: 400000

rx-usecs: 16
rx-frames: 128
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: 100000
pkt-rate-high: 400000

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

Note: When working with a 1GbE network, it is recommended to disable the interrupt moderation in order to get a full 1GbE throughput.
To do so, run: `ethtool -C eth11 adaptive-rx off rx-usecs 0 rx-frames 0`

3.6 Multi-Threaded IP Forwarding Tuning

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

1. Set the following options in /etc/modprobe.d/mlx4.conf.

```
options mlx4_en inline_thold=0
```

   - For MLNX_OFED-2.3-1.0.0:

```
options mlx4_core log_num_mgm_entry_size=-7
```

   - For MLNX_OFED-2.2-1.x.x and lower:

```
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 <interface> adaptive-rx off rx-usecs 0 tx-frames 64
```

3.7 Receive Side Scaling (RSS)

3.7.1 RSS Hash tuning

The default RSS hash calculated by the adapter is a Toeplitz function. On some workloads it is possible that small number of connections will not be distributed optimally across receive queues.

If this occurs, change the hash type to XOR, which is more optimal to small number of connections.

```
# ethtool --set-priv-flags <interface> mlx4_rss_xor_hash_function on
```

Note: RPS does not work when using XOR hash type.

3.7.2 ConnectX®-3/Connect-X® 3 Pro Optimized Steering

As of MLNX_OFED-2.3-1.0.0 ConnectX®-3/ ConnectX®-3 Pro adapter cards can be configured for optimized steering mode.

Note: Optimized steering mode may improve Ethernet packet rate, however, sideband management is not functional in this mode.

To use this optimization:

1. Edit /etc/modprobe.d/mlnx.conf:

   ```
   options mlx4_core log_num_mgm_entry_size=-7
   ```

2. Restart the driver.

3.8 Receive Packet Steering (RPS)

Receive Packet Steering (RPS) is a software implementation of RSS called later in the datapath. Contrary to RSS which selects the queue and consequently the CPU that runs the hardware interrupt handler, RPS selects the CPU to perform protocol processing on top of the interrupt handler. RPS requires a kernel compiled with the CONFIG_RPS kconfig symbol (ON by default for SMP). Even when compiled in, RPS remains disabled until explicitly configured. The list of CPUs to which RPS may forward traffic can be configured for each receive queue using a sysfs file entry:

```
/sys/class/net/<dev>/queues/rx-<n>/rps_cpus
```

For interfaces that have a single queue or its number of queues is less than the number of NUMA node cores, it is recommended to configure the rps_cpus mask to the device NUMA node core list to gain the better parallelism of multi queue interfaces.

Example:

When IPoIB is used in “connected” mode, it has only a single rx queue.

➢ ***To enable RPS:***

```
LOCAL_CPUS=`cat /sys/class/net/ib0/device/local_cpus`
echo $ LOCAL_CPUS > /sys/class/net/ib0/queues/rx-0/rps_cpus
```

For further information, please refer to
https://www.kernel.org/doc/Documentation/networking/scaling.txt

3.9 Tuning with sysctl

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.

3.9.1 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
```

The following variable is used to tell the kernel how much of the socket buffer space should be used for TCP window size, and how much to save for an application buffer.

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

A value of 1 means the socket buffer will be divided evenly between TCP windows size and application.

3.9.2 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 throughput:

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

3.9.3 Preserving Your sysctl 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 16) 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
```

3.10 Verbs Applications Optimization

3.10.1 Single Thread Applications

When running verbs applications that only have a single thread per process, it is recommended to enable the following environment variable:

- For ConnectX®-3 adapter family:

```
MLX4_SINGLE_THREADED=1
```

- When using Connect-IB® adapter family:

```
MLX5_SINGLE_THREADED=1
```

When single thread is enabled, the hardware library will remove expensive locks from the code and improve performance.

3.11 Performance Tuning for iSER

➢ ***To perform tuning for iSER:***

1. Set the SCSI scheduler to noop.

```
# echo noop > /sys/block/<block_dev>/queue/scheduler
```

2. Disable the SCSI `add_random`.

```
# echo 0 > /sys/block/<block_dev>/queue/add_random
```

3. Disable IO merges.

```
# echo 2 > /sys/block/<block_dev>/queue/nomerges
```

4. Disable the hyper-threading in BIOS configuration.

5. Set the CPU scaling governor to performance (if supported) (see Setting the Scaling Governor (on page 17)).
6. Increase the number of persistent huge pages in the kernel's huge page pool for user-space targets such as TGT.

```
# echo 3000 > /proc/sys/vm/nr_hugepages
```

   For kernel space targets such as LIO/SCST, decrease the number of persistent huge pages or set to zero.

```
# echo 0 > /proc/sys/vm/nr_hugepages
```

7. Set the IRQ Affinity hints (see IRQ Affinity Hints (on page 14)).

3.12 Tuning VMA Parameters

This section provides guidelines for improving performance with VMA. It is intended for administrators who are familiar with VMA and should be used in conjunction with the *VMA User Manual* and the *VMA Release Notes*.

You can minimize latency by tuning VMA parameters. It is recommended to test VMA performance tuning on an actual application.

We suggest that you try the following VMA parameters one by one and in combination to find the optimum for your application.

For more information about each parameter, see the *VMA User Manual*.

To perform tuning, add VMA configuration parameters when you run VMA, after **LD_PRELOAD**, for example:

```
LD_PRELOAD=libvma.so VMA_MTU=200 ./my-application
```

3.12.1 Memory Allocation Type

We recommend using contiguous pages (default). However, in case you want to use huge pages, do the following::

- Before running VMA, enable Kernel and VMA huge table, for example:

```
echo 1000000000 > /proc/sys/kernel/shmmax
echo 800 > /proc/sys/vm/nr_hugepages
```

  Note: Increase the amount of shared memory (bytes) and huge pages if you receive a warning about insufficient number of huge pages allocated in the system.

- Set VMA_MEM_ALLOC_TYPE. When set, VMA attempts to allocate data buffers as huge pages.

3.12.2 Reducing Memory Footprint

A smaller memory footprint reduces cache misses thereby improving performance. Configure the following parameters to reduce the memory footprint:

- If your application uses small messages, reduce the VMA MTU using:

```
VMA_MTU=200
```

- The default number of RX buffers is 200 K. Reduce the amount of RX buffers to 30 – 60 K using:

```
VMA_RX_BUFS=30000
```

Note: This value must not be less than the value of VMA_RX_WRE times the number of offloaded interfaces.

- The same can be done for TX buffers by changing VMA_TX_BUFS and VMA_TX_WRE

3.12.3 Polling Configurations

You can improve performance by setting the following polling configurations:

- Increase the number of times to unsuccessfully poll an Rx for VMA packets before going to sleep, using:

```
VMA_RX_POLL=200000
```

Or infinite polling, using:

```
VMA_RX_POLL=-1
```

This setting is recommended when Rx path latency is critical and CPU usage is not critical.

- Increase the duration in micro-seconds (usec) in which to poll the hardware on Rx path before blocking for an interrupt , using:

```
VMA-SELECT-POLL=100000
```

Or infinite polling, using:

```
VMA-SELECT-POLL=-1
```

This setting increases the number of times the selected path successfully receives poll hits, which improves the latency and causes increased CPU utilization.

- Disable the following polling parameters by setting their values to 0:
  - VMA_RX_POLL_OS_RATIO
  - VMA_SELECT_POLL_OS

    When disabled, only offloaded sockets are polled.

3.12.4 Handling Single-Threaded Processes

You can improve performance for single-threaded processes:

- Change the threading parameter to:

```
VMA_THREAD_MODE=0
```

This setting helps to eliminate VMA locks and improve performance.

3.13 Performance Tuning for Virtualized Environment

3.13.1 Tuning for Hypervisor

It is recommended to configure the "`iommu`" to "`pass-thru`" option in order to improve hypervisor performance.

➢ ***To configure the "iommu" to "pass-thru" option :***

- Add to kernel parameters:

```
Intel_iommu=on iommu=pt
```

The virtualization service might enable the global IPv4 forwarding, which in turn will cause all interfaces to disable their large receive offload capability.

➢ ***To re-enable large receive offload capability using ethtool:***

```
ethtool -K <interface> lro on
```

4 Performance Tuning for Windows

This document describes how to modify Windows registry parameters in order to improve performance. Please note that modifying the registry incorrectly might lead to serious problems, including the loss of data, system hang, and you may need to reinstall Windows. As such it is recommended to back up the registry on your system before implementing recommendations included in this document. If the modifications you apply lead to serious problems, you will be able to restore the original registry state. For more details about backing up and restoring the registry, please visit www.microsoft.com.

4.1 Tuning the Network Adapter

➢ ***To improve the network adapter performance, activate the performance tuning tool as follows:***

1. Select **Start-->Control Panel**.
2. Open **Network Connections**.
3. Right click on one of the entries **Mellanox ConnectX Ethernet Adapter** and select **Properties**.
4. Select the **Performance** tab.
5. Choose one of the Tuning Scenarios:
   - Single port traffic - Improves performance when running a single port traffic each time
   - Dual port traffic - Improves performance when running on both ports simultaneously
   - Forwarding traffic - Improves performance when running routing scenarios (for example via IXIA)
   - [Available in Mellanox WinOF v4.2 and above] Multicast traffic - Improves performance when the main traffic runs on multicast
   - [Available in Mellanox WinOF v4.2 and above] Single stream traffic - Optimizes tuning for applications with single connection
   - [Default] Balanaced tuning - Applies default values to various factors which may affect performance
6. Click the **Run Tuning** button.

   Clicking the **Run Tuning** button will change several registry entries (described below), and will check for system services that might decrease network performance. It will also generate a log including the applied changes.

   Users can view this log to restore the previous values. The log path is:

   %HOMEDRIVE%\Windows\System32\LogFiles\PerformanceTunning.log

This tuning is needed on one adapter only, and only once after the installation (as long as these entries are not changed directly in the registry, or by some other installation or script).

4.2 Tuning for NUMA Architecture

4.2.1 Tuning for Intel® Microarchitecture Code name Sandy Bridge / Ivy Bridge Platforms

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.

4.2.2 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).

4.2.3 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:***

```
start /affinity 0xff00 nd_write_bw -S/C <ip>
```

4.3 Tuning for Windows Server 2012 / 2012 R2

4.3.1 Recognizing NUMA Node Cores

➢ ***To recognize NUMA node cores, perform the following:***

1. Open the Task Manager.
2. Go to the "Performance" tab.
3. Choose "CPU".
4. Right click on graph and choose "Change graph to" -> "Logical processors".

   Hovering over a CPU will display its NUMA node.

4.4 Finding the Closest NUMA Node to the NIC

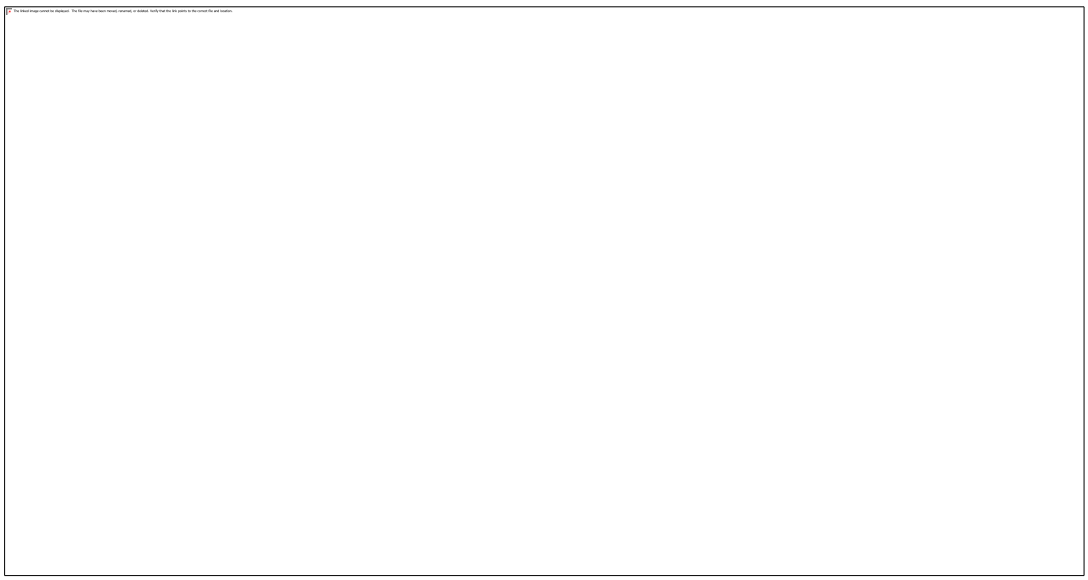
Note: BIOS support for ACPI SLIT must be enabled.

➢ ***To find the closest NUMA node to the NIC, perform the following:***

1. Open a PowerShell window.
2. Execute Get-NetAdapterRss –name <Connection Name>.

   Where `<Connection Name>` is the name assigned to the desired interface, e.g. “Ethernet 1”.

   Expected output:



The “`RssProcessorArray`” field displays the closer NUMA node.

The array should have entries that are of the form G:C/D.

- G - The processor group
- C - The processor core ID
- D - The distance between the NUMA node closest to the physical PCI slot where the NIC is installed, to the NUMA node where processor core C resides.

We recommend using only cores that have `D=0`, implying they are within the closest NUMA node to the NIC.

4.5 Tuning for Windows 2008 R2

Please use the perf_tuning.exe tool that comes with MLNX_VPI driver.

It will recognize the adapter's NUMA node automatically and set the relevant registry keys accordingly.

This tool is based on information retrieved from a tuning document that can be found here:

http://msdn.microsoft.com/en-us/windows/hardware/gg463392.aspx

The following are the auto-tuning options:

- **Optimized for single port** - use when most of the traffic is utilizing one of the NIC ports.

```
# perf_tuning.exe -s -c1 <connection name>
```

- **Optimized for dual port** - use when most of the traffic is utilizing both of the NIC ports.

```
# perf_tuning.exe -d -c1 <first connection name> -c2 <second connection
name>
```

- **Optimized for IP Routing (RFC2544)**

```
# perf_tuning.exe -f -c1 <first connection name> -c2 <second connection
name>
```

- **For multicast streams tuning**

```
# perf_tuning.exe -mc -c1 <first connection name> -c2 <second connection
name>
```

- **For single connection applications**

```
# perf_tuning.exe -st -c1 <first connection name>
```

Auto tuning can be performed using the User Interface as well. For further information, please refer to section Tuning the Network Adapter (on page 27).

4.5.1 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.

Please use the perf_tuning.exe manual option to separate each adapter's cores to different set of cores:

```
# perf_tuning.exe -m -c1 <first connection name> -b <base RSS processor
number> -n <number of RSS processors>
```

Example:

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

```
# perf_tuning.exe -m -c1 <first connection name> -b 0 -n 2
# perf_tuning.exe -m -c1 <first connection name> -b 2 -n 2
# perf_tuning.exe -m -c1 <first connection name> -b 4 -n 2
# perf_tuning.exe -m -c1 <first connection name> -b 6 -n 2
```

4.5.2 Recognizing NUMA Node Cores

➢ ***To recognize NUMA node cores, perform the following:***

1. Open the Task Manager.
2. Go to the "Processes" tab.
3. Right click on one of the processes and choose "Set affinity".

   A table of the available cores and NUMA nodes will be displayed.

4.6 Performance Testing

The preferred tool for performance testing is NTttcp. The tool was developed by Microsoft and it is well optimized for Windows operating systems.

Command line example:

- Receiver:

```
ntttcp_x64.exe -r -t 15 -m 16,*,<interface IP>
```

- Sender:

```
ntttcp_x64.exe -s -t 15 -m 16,*,<same address as above>
```

Note: Running the commands above with the `'-a 8'` parameter, may result in performance improvement due to higher overlapped IO/s allowed.

More details and tool binaries can be found here:
http://gallery.technet.microsoft.com/NTttcp-Version-528-Now-f8b12769