



# **Performance Tuning Guidelines for Mellanox Network Adapters**

**Revision 1.10**

**Last Updated December 04, 2013**

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](http://www.mellanox.com)  
Tel: (408) 970-3400  
Fax: (408) 970-3403

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

© Copyright 2013. Mellanox Technologies. All Rights Reserved.

Mellanox®, Mellanox logo, BridgeX®, ConnectX®, CORE-Direct®, InfiniBridge®, InfiniHost®, InfiniScale®, MLNX-OS®, PhyX®, SwitchX®, UFM®, Virtual Protocol Interconnect® and Voltaire® are registered trademarks of Mellanox Technologies, Ltd.

Connect-IB™, ExtendX™, FabricIT™, Mellanox Open Ethernet™, Mellanox Virtual Modular Switch™, MetroX™, MetroDX™, ScalableHPC™, Unbreakable-Link™ are trademarks of Mellanox Technologies, Ltd.

All other trademarks are property of their respective owners.

# Contents

<b>Revision History .....</b>	<b>6</b>
<b>1 Introduction .....</b>	<b>8</b>
1.1 Relevant Mellanox Drivers.....	8
<b>2 General System Configurations .....</b>	<b>9</b>
2.1 PCI Express (PCIe) Capabilities.....	9
2.2 Memory Configuration .....	9
2.3 Recommended BIOS Settings.....	10
2.3.1 General .....	10
2.3.2 Intel® Sandy Bridge Processors.....	10
2.3.3 Intel® Nehalem/Westmere Processors .....	11
2.3.4 AMD Processors .....	11
<b>3 Performance Tuning for Linux.....</b>	<b>12</b>
3.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance .....	12
3.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance .....	12
3.3 Preserving Your Performance Settings after a Reboot .....	13
3.4 Tuning Power Management .....	13
3.4.1 Checking Core Frequency .....	13
3.4.2 Setting the Scaling Governor.....	14
3.4.3 Kernel Idle Loop Tuning.....	14
3.4.4 OS Controlled Power Management.....	14
3.5 Interrupt Moderation .....	15
3.6 Tuning for NUMA Architecture.....	16
3.6.1 Tuning for Intel® Sandy Bridge Platform .....	16
3.6.2 Tuning for AMD® Architecture.....	16
3.6.3 Recognizing NUMA Node Cores .....	17
3.7 IRQ Affinity.....	17
3.7.1 IRQ Affinity Configuration .....	17
3.7.2 Auto Tuning Utility.....	18
3.7.3 Tuning for Multiple Adapters.....	18
3.8 Tuning Multi-Threaded IP Forwarding.....	18
3.9 Tuning VMA Parameters .....	19
3.9.1 Handling Huge Pages.....	19
3.9.2 Reducing Memory Footprint .....	19
3.9.3 Polling Configurations.....	20
3.9.4 Handling Single-Threaded Processes .....	20
3.9.5 Reducing DMAs .....	20

<b>4</b>	<b>Performance Tuning for Virtualized Environment .....</b>	<b>21</b>
4.1	Tuning for Hypervisor .....	21
<b>5</b>	<b>Performance Tuning for Windows .....</b>	<b>22</b>
5.1	Tuning the Network Adapter .....	22
5.2	Tuning for NUMA Architecture .....	23
5.2.1	Tuning for Intel® Microarchitecture Code name Sandy Bridge .....	23
5.2.2	Tuning for AMD® Architecture .....	23
5.2.3	Running an Application on a Certain NUMA Node .....	23
5.3	Tuning for Windows Server 2012 .....	23
5.3.1	Recognizing NUMA Node Cores .....	23
5.4	Finding the Closest NUMA Node to the NIC .....	24
5.5	Tuning for Windows 2008 R2 .....	25
5.5.1	Tuning for Multiple Adapters .....	25
5.5.2	Recognizing NUMA Node Cores .....	25
5.6	Performance Testing .....	26

## List of Tables

Table 1: Document Revision History .....	6
Table 2: Recommended PCIe Configuration.....	9

## Revision History

**Table 1: Document Revision History**

Revision	Date	Description
1.10	December, 2013	<ul style="list-style-type: none"> <li>Updated section <a href="#">Performance Testing</a> (on page <a href="#">26</a>)</li> </ul>
1.10	October, 2013	<ul style="list-style-type: none"> <li>Updated section <a href="#">Kernel Idle Loop Tuning</a> (on page <a href="#">14</a>)</li> <li>Added section <a href="#">Performance Tuning for Virtualized Environment</a> (on page <a href="#">21</a>)</li> </ul>
1.9	September, 2013	<ul style="list-style-type: none"> <li>Updated section <a href="#">Interrupt Moderation</a> (on page <a href="#">15</a>)</li> </ul>
1.8	June, 2013	<ul style="list-style-type: none"> <li>Removed section <i>Tuning for Windows Server 2008</i> and its sub-sections</li> <li>Added the following sections: <ul style="list-style-type: none"> <li><a href="#">Recognizing NUMA Node Cores</a> (on page <a href="#">23</a>)</li> <li><a href="#">Finding the Closest NUMA Node to the NIC</a> (on page <a href="#">24</a>)</li> </ul> </li> </ul>
1.7	April, 2013	<ul style="list-style-type: none"> <li>Updated the following sections: <ul style="list-style-type: none"> <li><a href="#">Recommended BIOS Settings</a> (on page <a href="#">10</a>)</li> <li><a href="#">Tuning Power Management</a> (on page <a href="#">13</a>)</li> <li><a href="#">Tuning for Intel® Sandy Bridge</a> (on page <a href="#">16</a>)</li> <li><a href="#">IRQ Affinity Configuration</a> (on page <a href="#">17</a>)</li> <li><a href="#">Tuning Multi-Threaded IP Forwarding</a> (on page <a href="#">18</a>)</li> <li><a href="#">Tuning for Multiple Adapters</a> (on page <a href="#">18</a>)</li> </ul> </li> <li>Replaced “Tuning for IPoIB Interfaces” with <a href="#">Auto Tuning Utility</a> (on page <a href="#">18</a>)</li> <li>Added section <a href="#">Improving Application Performance on Remote NUMA Node</a> (on page <a href="#">16</a>)</li> </ul>
1.6	October, 2012	<ul style="list-style-type: none"> <li>Added the following sections: <ul style="list-style-type: none"> <li><a href="#">Recognizing NUMA Node Cores</a> (on page <a href="#">17</a>)</li> <li><a href="#">Running an Application on a Certain NUMA Node</a> (on page <a href="#">17</a>)</li> <li><a href="#">Running an Application on a Certain NUMA Node</a> (on page <a href="#">23</a>)</li> <li><a href="#">Recognizing NUMA Node Cores</a> (on page <a href="#">23</a>)</li> <li><a href="#">Recognizing NUMA Node Cores</a> (on page <a href="#">25</a>)</li> </ul> </li> <li>Updated the following sections: <ul style="list-style-type: none"> <li><a href="#">Tuning the Network Adapter</a> (on page <a href="#">22</a>)</li> </ul> </li> </ul>

Revision	Date	Description
1.5	May, 2012	<ul style="list-style-type: none"> <li>Added the following sections: <ul style="list-style-type: none"> <li><a href="#">Memory Configuration</a> (on page 9)</li> <li><a href="#">Tuning for IPoIB/EoIB Interfaces</a> (on page 18)</li> <li><a href="#">Kernel Idle Loop Tuning</a> (on page 14)</li> </ul> </li> <li>Updated the following sections: <ul style="list-style-type: none"> <li><a href="#">IRQ Affinity Configuration</a> (on page 17)</li> <li><a href="#">Recommended BIOS Settings</a> (on page 9)</li> <li><a href="#">Tuning for Multiple Adapters</a> (on page 18)</li> <li><a href="#">Tuning for Windows 2008 R2</a> (on page 23)</li> </ul> </li> </ul>
1.4	April, 2012	<ul style="list-style-type: none"> <li>Added “Tuning for NUMA Architecture” sections.</li> <li>Rearranged section in chapter 3.</li> </ul>
1.3	March, 2012	<ul style="list-style-type: none"> <li>Added new section “Tuning Power Management”.</li> </ul>
1.2	January, 2012	<ul style="list-style-type: none"> <li>Updated versions of adapters to make the document more generic.</li> <li>Merged sections on BIOS Power Management Settings and Intel Hyper-Threading Technology to new section, “Recommended BIOS Settings”.</li> <li>Added sections to “Performing Tuning for Linux”.</li> <li>Added section, “Tuning for Windows 2008 R2”.</li> <li>Added new chapter, “Tuning VMA Parameters”.</li> </ul>
1.1		<ul style="list-style-type: none"> <li>Updated the following sections: <ul style="list-style-type: none"> <li>“Intel® Hyper-Threading Technology”</li> <li>“Tuning the Network Adapter for Improved IPv4 Traffic Performance”</li> <li>“Example: Script for Setting Interrupt Affinity”</li> </ul> </li> <li>Added new section, “Tuning IP Forwarding”.</li> </ul>

# 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 1.5.10 and later
- On *Linux*: Mellanox VPI Driver *MLNX\_OFED* for *Linux* version 2.0.x and later
- On *Windows*: Mellanox OFED for Windows *MLNX\_VPI* version 4.40 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 Recommended BIOS Settings



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

### 2.3.1 General

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

### 2.3.2 Intel® Sandy Bridge Processors

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

BIOS Option		Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Enabled
	Hyper-Threading <sup>1</sup>	HPC: disabled Data Centers: enabled
	CPU frequency select	Max performance
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

<sup>1</sup> 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.

### 2.3.3 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 <sup>2</sup>	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

### 2.3.4 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
	Channel Interleaving	Enabled
	Thermal Mode	Performance

<sup>2</sup> 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.

## 3 Performance Tuning for Linux

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

### 3.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance

The following changes are recommended for improving IPv4 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

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

- Enable the TCP selective acks option for better throughput:

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

- Increase the maximum length of processor input queues:

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

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

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

- Increase memory thresholds to prevent packet dropping:

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

- Enable low latency mode for TCP:

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

### 3.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance

The following changes are recommended for improving IPv6 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

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

- Enable the TCP selective acks option for better throughput:

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

### 3.3 Preserving Your Performance Settings after a Reboot

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

```
<sysctl name1>=<value1>
```

```
<sysctl name2>=<value2>
```

```
<sysctl name3>=<value3>
```

```
<sysctl name4>=<value4>
```

For example, [Tuning the Network Adapter for Improved IPv4 Traffic Performance](#) (on page 12) 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
```

### 3.4 Tuning Power Management

#### 3.4.1 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 9) 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
```

### 3.4.2 Setting the Scaling Governor

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

- *freq\_table*
- *acpi\_cpufreq*: this module is architecture dependent.

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

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

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

➤ *To disable cpuspeed, use:*

```
# service cpuspeed stop
```

### 3.4.3 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, it must be created having the same name as the one stated above.

- For MLNX\_OFED 2.0.x

```
options mlx4_core enable_sys_tune=1
```

- For MLNX\_EN 1.5.10

```
options mlx4_en enable_sys_tune=1
```

### 3.4.4 OS Controlled Power Management

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

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

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

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

3. Reboot the system.

Example:

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

### 3.5 Interrupt Moderation

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

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

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

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

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

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

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



**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 eth1 adaptive-rx off rx-usecs 0 rx-frames 0`

## 3.6 Tuning for NUMA Architecture

### 3.6.1 Tuning for Intel® Sandy Bridge Platform

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

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

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

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

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

Example for supported system:

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

Example for unsupported system:

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

#### 3.6.1.1 Improving Application Performance on Remote NUMA Node

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

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

However, the NUMA node recognition must be enabled as described in section [3.6.1](#).

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 NUMA node which its cores are 0-7:

```
MLX4_LOCAL_CPUS=0xff
```

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

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



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

### 3.6.2 Tuning for AMD® Architecture

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

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



### 3.6.3 Recognizing NUMA Node Cores

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

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

Example:

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

#### 3.6.3.1 Running an Application on a Certain NUMA Node

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

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

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

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

or:

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

## 3.7 IRQ Affinity

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

The following command turns off the IRQ balancer:

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

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

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

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

#### 3.7.1 IRQ Affinity Configuration



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

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

- For optimizing single-port traffic, run:

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

- For optimizing dual-port traffic, run:

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

- To show the current irq affinity settings, run:

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

### 3.7.2 Auto Tuning Utility

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

Usage:

- Start

```
# mlnx_affinity start
```

- Stop

```
# mlnx_affinity stop
```

- Restart

```
# mlnx_affinity restart
```

`mlnx_affinity` can also be started by driver load/unload

➤ *To enable `mlnx_affinity` by default:*

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

```
RUN_AFFINITY_TUNER=yes
```

### 3.7.3 Tuning for Multiple Adapters

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

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

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

Example:

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

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

## 3.8 Tuning Multi-Threaded IP Forwarding

➤ *To optimize NIC usage as IP forwarding:*

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

- For MLNX\_OFED-2.0.x:

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

- For MLNX\_EN-1.5.10:

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

2. Apply interrupt affinity tuning.

3. Forwarding on the same interface:

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

4. Forwarding from one interface to another:

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

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

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

## 3.9 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
```

### 3.9.1 Handling Huge Pages

Improve the handling of huge pages:

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

```
echo 1000000000 > /proc/sys/kernel/shmmax
echo 400 > /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.

- Enable *VMA\_HUGETBL* to improve receive and send performance. When enabled, VMA attempts to allocate data buffers as huge pages.

### 3.9.2 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 RX buffer is 200 KB. Reduce your RX buffers to 30 – 60 KB using:

```
VMA_RX_BUFS=30000
```

**Note:** This value must not be less than the value for *VMA\_RX\_WRE*

### 3.9.3 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=100000
```

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
```

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\_SKIP\_OS
  - VMA\_RX\_POLL\_OS\_RATIO - When disabled, only offloaded sockets are polled.
  - VMA\_SELECT\_POLL\_OS\_RATIO - When disabled, only offloaded sockets are polled.
  - VMA\_SELECT\_SKIP\_OS

### 3.9.4 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.

### 3.9.5 Reducing DMAs

- Reduce the number of DMAs (direct memory access actions) the NIC performs by using:

```
VMA_TX_SGE=5
```

If you are testing 128 Bytes UDP Multicast payload (add headers 8 UDP + 20 IP + 14 MAC), you need the inline value to be just above 170 Bytes. VMA\_TX\_SGE=5 will give the best value in this case since it is more than the total packet size but very little extra buffer space wasted.

## 4 Performance Tuning for Virtualized Environment

### 4.1 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
```

## 5 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](http://www.microsoft.com).

### 5.1 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
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).

## 5.2 Tuning for NUMA Architecture

### 5.2.1 Tuning for Intel® Microarchitecture Code name Sandy Bridge

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.

### 5.2.2 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).

### 5.2.3 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>
```

## 5.3 Tuning for Windows Server 2012

### 5.3.1 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.





## 5.5 Tuning for Windows 2008 R2



**NOTE:** perf\_tuning.exe is supported in Windows 2008 R2 only .

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 22).

### 5.5.1 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
```

### 5.5.2 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.

## 5.6 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>
```

More details and tool binaries can be found here:

<http://gallery.technet.microsoft.com/NTttcp-Version-528-Now-f8b12769>