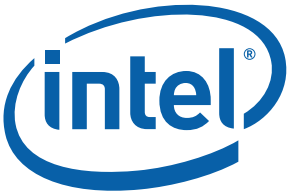

Intel® Ethernet Controller X710/XL710 and Intel® Ethernet Converged Network Adapter X710/XL710 Family

Linux* Performance Tuning Guide

March 2016

Revision 1.0
Order Number: 334019-001

Revision History

Revision	Date	Comments
1.0	March 2016	Initial Release (Intel Public)

1.0 Introduction

This guide is intended to provide guidance for tuning environments for optimal networking performance using an Intel® Ethernet Converged Network Adapter XL710 and/or an Intel® Ethernet Converged Network Adapter X710 in Linux environments. It focuses on hardware, driver, and operating system conditions and settings that might improve network performance. It should be noted that networking performance can be affected by any number of outside influences, only the most common and dramatic of these are covered in this guide.

2.0 Initial Checklist

2.1 Update Driver/Firmware Versions

Check driver/firmware versions with **ethtool -i ethX**.

Update i40e driver — Get the latest from:

http://sourceforge.net/projects/e1000/files/i40e%20stable/ or https://downloadcenter.intel.com/download/24411/Network-Adapter-Driver-for-PCI-E-40-Gigabit-Network-Connections-under-Linux-

Update firmware — Get the latest from:

https://downloadcenter.intel.com/download/24769/NVM-Update-Utility-for-Intel-Ethernet-Converged-Network-Adapter-XL710-X710-Series

2.2 Read the README

Check for known issues and get the latest configuration instructions from the README text included in the i40e source package.

2.3 Check That Your PCI Express* (PCIe*) Slot is x8

Some PCIe x8 slots are actually configured as x4 slots. These slots have insufficient bandwidth for full line rate with dual port and quad port devices. In addition, if you put a PCIe v3.0-capable adapter into a PCIe v2.x slot, you cannot get full bandwidth. The software device driver detects this situation and writes the following message in the system log:

```
PCI-Express bandwidth available for this card is not sufficient for optimal
performance. For optimal performance a x8 PCI-Express slot is required.
```

If this error occurs, moving your adapter to a true PCIe v3.0 x8 slot resolves the issue.

2.4 Check System Hardware Capabilities

At 10 Gb/s and 40 Gb/s Ethernet, there are some minimum CPU and system requirements. In general, a modern server class processor and optimal memory configuration for your platform should be sufficient, but the needs vary depending on your workload. All memory channels should be populated and memory performance mode should be enabled in the BIOS. Verify that your CPU and memory configuration are capable of supporting the level of network performance you require for your workload.

Note: The XL710 is a 40 GbE controller. The 2 x 40 GbE adapter using this controller is not intended to be a 2 x 40 GbE but a 1 x 40 GbE with an active back-up port. When attempting to use line-rate traffic involving both ports, the internal switch is saturated and the combined bandwidth between the two ports are limited to a total of 50 Gb/s.

2.4.1 Kernel Boot Parameters

If Intel® Virtualization Technology for Directed I/O (Intel® VT-d) is enabled in the BIOS, Intel recommends that IOMMU be in pass-through mode for optimal host network performance. This eliminates DMA overhead on host traffic while enabling Virtual Machines (VMs) to still have the benefits of Intel® VT-d. This is accomplished by adding the following line to the kernel boot parameters:

```
iommu=pt
```

3.0 Baseline Performance Measurements and Tuning Methodology

3.1 Network Performance Benchmarks

Before beginning a tuning exercise, it is important to have a good baseline measurement of your network performance. Usually in addition to getting an initial measurement of your specific application/ workload's performance, it's a good idea to also use a standard network performance benchmark to verify that your network device is in a good state.

For single system optimization, **netperf** or **iperf** and **NetPIPE** are all solid open-source free tools that enable you to stress a connection and diagnose performance issues. **Netperf** is strong for both throughput and latency testing. **NetPIPE** is a latency-specific tool but can be compiled for any sort of environment.

Note: The TCP_RR test in **netperf** returns latency in a value of transactions/sec. This is a round-trip number. The one-way latency can be calculated using the following equation:

```
Latency(usec) = (½) / [Transactions/sec] * 1,000,000
```

3.2 Tuning Methodology

Focus on one tuning change at a time so you know what impact each change makes to your test. The more methodical you are in the tuning process, the easier it will be to identify and address the causes of performance bottlenecks.

4.0 Tuning i40e Driver Settings

4.1 IRQ Affinity

Configuring IRQ affinity so that interrupts for different network queues are affinitized to different CPU cores can have a huge impact on performance, particularly multi-thread throughput tests.

To configure IRQ affinity, stop **irqbalance** and then either use the `set_irq_affinity` script from the i40e source package or pin queues manually.

Disable user-space IRQ balancer to enable queue pinning:

- **systemctl** disable **irqbalance**
- **systemctl** stop **irqbalance**

Using the `set_irq_affinity` script from the i40e source package (recommended):

- To use all cores:

```
[path-to-i40epackage]/scripts/set_irq_affinity -x all ethX
```

- To use only cores on the local NUMA socket:

```
[path-to-i40epackage]/scripts/set_irq_affinity -x local ethX
```

- You can also select a range of cores. Avoid using cpu0 because it runs timer tasks.

```
[path-to-i40epackage]/scripts/set_irq_affinity 1-2 ethX
```

Manually:

- Find the processors attached to each node using:

```
numactl --hardware
lscpu
```

- Find the bit masks for each of the processors:
- Assuming cores 0-11 for node 0: [1,2,4,8,10,20,40,80,100,200,400,800]
- Find the IRQs assigned to the port being assigned:

```
grep ethX /proc/interrupts and note the IRQ values
For example, 181-192 for the 12 vectors loaded.
```

- Echo the SMP affinity value into the corresponding IRQ entry. Note that this needs to be done for each IRQ entry:

```
echo 1 > /proc/irq/181/smp_affinity
echo 2 > /proc/irq/182/smp_affinity
echo 4 > /proc/irq/183/smp_affinity
```

4.2 Interrupt Moderation

Adaptive interrupt moderation is on by default, and is designed to provide a balanced approach between low CPU utilization and high performance. However, you might try tuning interrupt settings manually to fit your use case.

The range of 0-235 microseconds provides an effective range of 4,310 to 250,000 interrupts per second. The value of rx-µsecs-high can be set independent of rx-µsecs and tx-µsecs in the same **ethtool** command, and is also independent of the adaptive interrupt moderation algorithm. The underlying hardware supports granularity in 2-microsecond intervals, so adjacent values might result in the same interrupt rate.

- To turn off adaptive interrupt moderation

```
ethtool -C ethX adaptive-rx off adaptive-tx off
```

- To turn on adaptive interrupt moderation

```
ethtool -C ethX adaptive-rx on adaptive-tx on
```

A good place to start for general tuning is 84 µs, or ~12000 interrupts/s. If you see rx_dropped counters are running during traffic (using **ethtool -S ethX**) then you probably have too slow of a CPU, not enough buffers from the adapter's ring size (**ethtool -G**) to hold packets for 84 µs or to low of an interrupt rate.

- To set interrupt moderation to a fixed interrupt rate of 84 µs between interrupts (12000 interrupts/s):

```
ethtool -C ethX adaptive-rx off adaptive-tx off rx-usecs 84 tx-usecs 84
```

The next value to try, if you are not maxed out on CPU utilization, is 62 µs. This uses more CPU, but it services buffers faster, and requires fewer descriptors (ring size, **ethtool -G**).

- To set interrupt moderation to fixed interrupt rate of 62 usecs between interrupts (16000 interrupts/s).

```
ethtool -C ethX adaptive-rx off adaptive-tx off rx-usecs 62 tx-usecs 62
```

If your CPU is at 100%, then increasing the interrupt rate is not advised. In certain circumstances such as a CPU bound workload, you might want to increase the µs value to enable more CPU time for other applications.

If you require low latency performance and/or have plenty of CPU to devote to network processing, you can disable interrupt moderation entirely, which enables the interrupts to fire as fast as possible.

- To disable interrupt moderation

```
ethtool -C ethX adaptive-rx off adaptive-tx off rx-usecs 0 tx-usecs 0
```

4.3 Ring Size

If you are seeing rx_dropped counters in **ethtool -S ethX**, or suspect cache pressure with multiple queues active, you might try adjusting the ring size from the default value. The default value is 512, the max is 4096.

If it is suspected that lack of buffering is causing drops at the current interrupt rate, you might try the maximum first, then the minimum, then continue on in a binary search until you see optimal performance.

If cache pressure is suspected (many queues active) reducing buffers from default can help Intel® Data Direct I/O (Intel® DDIO) operate with better efficiently. Intel recommends trying 128 or 256 per queue, being aware that an increase in interrupt rate via **ethtool -C** might be necessary to avoid an increase in rx_dropped.

- To set ring size to fixed value:

```
ethtool -G eth12 rx 256 tx 256
```

4.4 Flow Control

Layer 2 flow control can impact TCP performance considerably and is recommended to be disabled for most workloads. A potential exception is bursty traffic where the bursts are not long in duration.

Flow control is disabled by default.

To enable flow control:

```
ethtool -A ethX off rx on tx on
```

To disable flow control:

```
ethtool -A ethX rx off tx off
```

Note: You must have a flow control capable link partner to successfully enable flow control.

5.0 System Tuning (i40e Non-specific)

5.1 BIOS Settings

Enable Intel® VT-d for virtualization workloads.

Hyper threading (logical processors) can affect performance. Experiment with on or off for your workload.

5.2 Power Management

Power management can impact performance, particularly in low latency workloads. If performance is a higher priority than lowering power consumption, it is recommended to experiment with limiting the effects of power management. There are many different ways to limit power management, through operating system tools, BIOS settings, and kernel boot parameters. Choose the best method and level to suit your environment.

5.2.1 C-state Control

Limiting C-state entry to C0 or C1 improves performance and increases power utilization.

One way to dynamically control C-state entry is by opening a file (`/dev/cpu_dma_latency`) and writing the maximum allowable latency to it. There is a small program called `cpudmalatency.c` that can be downloaded from the open source community, compiled, and run from the command line to do exactly this. This example allows five µs of wake time, and thus allows C1 entry:

```
cpudmalatency 5 &
```

Another option is to limit the maximum C-state in the kernel boot settings:

`intel_idle.max_cstates=1 (for Intel CPUs) or processor.max_cstates=1` (for non-Intel CPUs)

You can also disallow any C-state entry by adding the following to the kernel boot line:

```
idle=poll
```

The third option is through the system's BIOS power management settings, which might have a performance profile available.

Tools such as **turbostat** or **x86_energy_perf_policy** can be used to check or set power management settings.

5.2.2 PCIe Power Management

Active-State Power Management (ASPM) enables a lower power state for PCIe links when they are not in active use. This can cause higher latency on PCIe network devices, so it is recommended to disable ASPM for latency-sensitive workloads.

Disable ASPM by adding the following to the kernel boot line:

```
pcie_aspm=off
```

5.2.3 CPU Frequency Scaling

CPU frequency scaling (or CPU speed scaling) is a Linux power management technique in which the system clock speed is adjusted on the fly to save power and heat. Just like C-states, this can cause unwanted latency on network connections.

To disable CPU frequency scaling, disable the CPU power service by the following commands:

```
systemctl stop cpupower.service
systemctl disable cpupower.service
```

5.3 Firewalls

Firewalls can impact performance, particularly latency performance.

Disable `iptables/firewalld` if not required.

5.4 Application Settings

Often a single thread (which corresponds to a single network queue) is not sufficient to achieve maximum bandwidth.

Experiment with increasing the number of threads used by your application if possible.

5.5 Kernel Version

Most modern in-box kernels are reasonably well optimized for performance but, depending on your use case, updating the kernel might provide improved performance. Downloading the source also enables you to enable/disable certain features before building the kernel.

5.6 Operating System/Kernel Settings

Consult operating system tuning guides such as the *Red Hat Enterprise Linux Network Performance Tuning Guide* for more insight on general operating system tuning (https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf).

Some common parameters to tune are listed in the following table. Note that these are only suggested starting points, and changing them from the defaults might increase the resources used on the system. Though increasing the values can help improve performance, it is necessary to experiment with different values to determine what works best for a given system, workload and traffic type.

The kernel parameters are configurable using the **sysctl** utility in Linux as indicated in this example:

```
sysctl -w net.core.rmem_default=524287.
```

Running **sysctl** without the **-w** argument lists the parameter with its current setting.

Stack Setting	Description
net.core.rmem_default	Default Receive Window Size
net.core.wmem_default	Default Transmit Window Size
net.core.rmem_max	Maximum Receive Window Size
net.core.wmem_max	Maximum Transmit Window Size

Stack Setting	Description
net.core.optmem_max	Maximum Option Memory Buffers
net.core.netdev_max_backlog	Backlog of unprocessed packets before kernel starts dropping
net.ipv4.tcp_rmem	Memory reserver for TCP read buffers
net.ipv4.tcp_wmem	Memory reserver for TCP send buffers

6.0 Performance Troubleshooting

6.1 CPU Utilization

Check CPU utilization per core while workload is running. Note that utilization per core is more relevant to performance than overall CPU utilization, since it provides an idea of the CPU utilization per network queue. If you have only a few threads running network traffic then you might only have a few cores being used. However, if those cores are at 100% then your network throughput is likely limited by CPU utilization and it is time to:

1. Tune IRQ moderation/ring size as detailed in Section 4.3.
2. Increase the number of application threads to spread out the CPU load over more cores. If all cores are running at 100% then your application might be CPU bound rather than network bound.

Commonly available tools:

- **top**

  Press 1 to expand list of CPUs and check which ones are being used.

  Notice level of utilization.

  Notice which processes are listed as most active (top of list).

- **mpstat**

The following example command line was tested on Red Hat Enterprise Linux 7.x. It displays CPU utilization per core (by finding the total percent idle and subtracting from 100) and highlights the values above 80% in red.

```
mpstat -P ALL 1 1 | grep -v Average | tail -n +5 | head -n -1 | awk '{ print (100-$13)}' | egrep --
color=always '[^\.][8-9][0-9][\.]?.*|^[8-9][0-9][\.]?.*|100|' | column
```

- **perf top**

  Look for where cycles are being spent.

6.2 i40e Counters

The i40e driver provides a long list of counters for interface debug and monitoring through the **ethtool -S ethX** command. It can be helpful to watch the output while a workload is running and/or compare the counter values before and after a workload run.

To get a full dump of i40e counters:

```
ethtool -S ethX
```

To watch just non-zero counters:

```
watch -d (ethtool -S ethX) | egrep -v :\ 0 | column
```

Some things to look for:

rx_dropped: means CPU not servicing buffers fast enough.

port.rx_dropped means something is not fast enough in the slot/memory/system.

6.3 Network Counters

Check **netstat -s** before/after a workload run.

Netstat collects network information from all network devices in the system, so results might be impacted from other networks other than your network under test. However, the output from **netstat -s** can be a good indicator of performance issues in the Linux operating system or kernel. Consult operating system tuning guides such as the *Red Hat Enterprise Linux Network Performance Tuning Guide* for more insight on general operating system tuning (https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf).

6.4 System Logs

Check system logs for errors and warnings (`/var/log/messages`, `dmesg`).

7.0 Recommendations for Common Performance Scenarios

7.1 IP Forwarding

- Update the kernel

Some recent in-distro kernels have degraded routing performance due to kernel changes in the routing code starting with the removal of the routing cache due to security. Recent out-of-distro kernels should have patches that alleviate the performance impact of these changes and might provide improved performance.

- Disable hyperthreading (logical cores)
- Edit kernel boot parameters

  Force **iommu off** (**intel_iommu=off** or **iommu=off**) from the kernel boot line unless required for virtualization

  Turn off power management:

  ```
  processor.max_cstates=1 idle=poll pcie_aspm=off
  ```

- Limit number of queues to be equal to the number of cores on the local socket (12 in this example):

  ```
  ethtool -L ethX combined 12
  ```

- Pin interrupts to local socket only

  ```
  set_irq_affinity -x local ethX
  ```

- Change the Tx and Rx ring sizes as needed, a larger value takes more resources but can provide better forwarding rates:

  ```
  ethtool -G ethX rx 4096 tx 4096
  ```

- Disable GRO when routing

  Due to a known kernel issue, GRO must be turned off when routing/forwarding.

  ```
  ethtool -K ethX gro off
  ```

  where ethX is the Ethernet interface to be modified.

- Disable adaptive interrupt moderation and set a static value:

  ```
  ethtool -C ethX adaptive-rx off adaptive-tx off
  ethtool -C ethX rx-usecs 64 tx-usecs 64
  ```

- Disable the firewall

  ```
  sudo systemctl disable firewalld
  sudo systemctl stop firewalld
  ```

- Enable IP forwarding:

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

7.2 Low Latency

- Turn hyperthreading (logical cores) OFF
- Ensure network device is local to numa core 0
- Pin benchmark to core 0 using **taskset -c** 0
- Turn **irqbalance** off **systemctl** stop **irqbalance** or **systemctl** disable **irqbalance**
- Run affinity script to spread across cores - try either local or all
- Turn off interrupt moderation:

  ```
  ethtool -C ethX rx-usecs 0 tx-usecs 0 adaptive-rx off adaptive-tx off rx-usecs-high 0
  ```

- Use an established benchmark like **netperf -t** TCP_RR or **netperf -t** UDP_RR or **NetPipe**.

LEGAL