Comparison of 40G RDMA and Traditional Ethernet Technologies

Nichole Boscia, Harjot S. Sidhu[1]
NASA Advanced Supercomputing Division
NASA Ames Research Center
Moffett Field, CA 94035-1000
Nichole.K.Boscia@nasa.gov
Harjot.S.Sidhu@nasa.gov

I. Introduction

In the world of computer networking, Ethernet has long been the standard for local connectivity. It is highly manageable, versatile, and supported by every industry vendor. Over the years, Ethernet speed has increased from 10 megabits per second (Mbps) to 100 gigabits per second (Gbps) and researchers are already planning to scale up to 1 terabits per second (Tbps). While servers with 10 gigabit Ethernet (10 GbE) network interfaces have been around for many years, the capability to have a single file transfer saturate a link is becoming more common due to more efficient processors, a faster PCI Express (PCIe) bus, and more sophisticated transfer protocols. Centralized services, such as backup servers, often need to handle parallel data transfers contending for bandwidth. As 10 GbE is becoming an inexpensive commodity with a growing installation base, there is a need to deploy faster technologies to accommodate the aggregation of those flows. In a supercomputing environment where systems are capable of processing large data sets at a high-speed rate, 40 Gb Ethernet (40 GbE) can reduce overall transfer times from days to hours.

As the number of data-traversing networks grows, network capabilities must continue to scale with it.

II. Converged vs. Traditional Ethernet

One of the desirable features associated with InfiniBand, another network fabric technology, is its Remote Direct Memory Access (RDMA) capability. RDMA allows for communication between systems but can bypass the overhead associated with the operating system kernel, so applications have reduced latency and much lower CPU utilization. This results in much faster network performance rates than traditional TCP/IP.

Although InfiniBand has historically been the network technology used for RDMA applications because of the lossless fabric requirement, RDMA does not have all of the functionality that traditional socket-based API (Application Programming Interface) provides and cannot detect when a frame or packet is dropped. Ethernet, however, relies on TCP to ensure that all data sent across the network arrives at its destination and retransmits data as required, and recent advances in priority queuing can now ensure a completely lossless Ethernet fabric. This capability allows other data center technologies such as Fibre Channel over Ethernet (FCoE) and InfiniBand/RDMA (RoCE) to converge over Ethernet.

RoCE is picking up momentum in the networking industry by bridging the "best of both worlds" – organizations can continue to use their existing Ethernet infrastructure, yet still benefit from what

1 Employees of CSC, Inc. through contract no. NNA07CA29C with NASA Ames

RDMA has to offer. The improvement in performance is substantial, as we will show in the "Testing Results" section of this document.

III. Setup and Configuration

This section provides a description of the test environment components used to evaluate the benefits and determine the optimal expected performance of using a 40 GbE Local Area Network (LAN) in a high-end computing environment.

Server Configuration

- Intel S2600IP4 Motherboard with PCIe 3.0
- Two Intel Xeon E5-2667 "Sandy Bridge" Processors
- 32 GB DDR3 1600Mhz dual inline memory modules (DIMM)
- Four 1 TB, 7200 RPM, 64 MB SATA-II hard disk drives (HDD),
- Mellanox 40GbE ConnectX-3 VPI Network Interface Cards (NIC) with firmware version 2.11.500
- CentOS 6.2 Linux OS
- Mellanox OFED 2.0 driver version

Demo Network Equipment

- Cisco Nexus 3016Q switch
- Cisco QSFP-40G-SR4 Transceiver Module with MPO Connector

The test environment contains two high-end servers equipped with Intel Sandy Bridge processors, 32 GB of RAM, and Mellanox 40GbE Connect-X3 Network Interface Cards (NIC). Both servers run the CentOS 6.2 Linux operating system with Mellanox OFED 2.0 drivers. Standard 12-strand optical MTP cable was used with Cisco QSPF transceiver connectors to connect the servers to the Cisco Nexus 3016 network switch, which is a high-performance, ultra-low-latency Ethernet switch. To achieve the rated speed of the Mellanox ConnectX-3 NIC, a PCIe Gen3 slot was used. Figure 1, below, shows a diagram of the test environment.

Note: Gen3 slots are available on newer systems, such as those equipped with an Intel Romley motherboard. If an older system is used, the NIC will be limited by the speed of the older PCIe Gen2 bus.

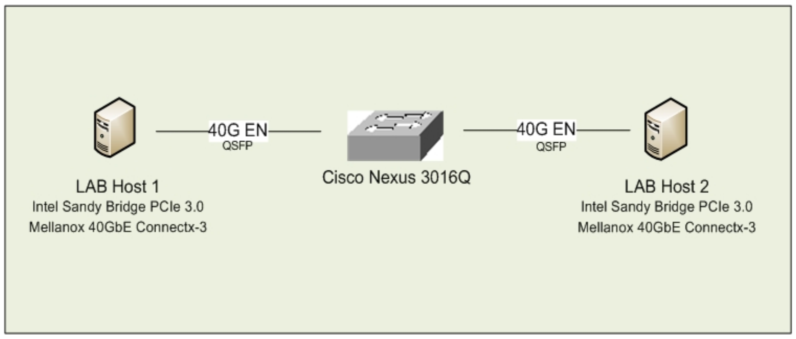
40G EN
QSFP
40G EN
QSFP
LAB Host 1
Intel Sandy Bridge PCIe 3.0
Mellanox 40GbE Connectx-3
Cisco Nexus 3016Q
LAB Host 2
Intel Sandy Bridge PCIe 3.0
Mellanox 40GbE Connectx-3

Figure 1. Test environment diagram.

IV. System Tuning

It is important to remember that the testing described in this document is intended explicitly for low-latency LAN environments. Many of the modifications listed in this section will cause very poor results in wide-area network (WAN) environments. The bottleneck for a 40 GbE NIC is generally the CPU. While RDMA-based applications greatly reduce CPU utilization and can eliminate system interrupts, traditional Ethernet needs a lot of optimized tuning to accomplish the same results. Linux offers a variety of mechanisms to customize relevant settings, as shown in the following sections.

NIC Offloading

Most NICs come with a variety of options for offloading tasks to the NIC itself. By default, all driver options are enabled, as shown in Table 1:

Table 1. NIC Driver Offload Settings

Offloading Type	Default
TCP Segmentation Offloading (TSO)	On
Rx Checksumming	On
Tx Checksumming	On
Large Receive Offload (LRO)	On
Scatter-Gatter (SG)	On
Generic Segmentation Offload (GRO)	On

Kernel Tuning

Disabling these parameters can help reduce CPU processing and overhead:

```
sysctl -w net.ipv4.tcp_low_latency=1
sysctl -w net.ipv4.tcp_timestamps=0
sysctl -w net.ipv4.tcp_sack=0
sysctl -w net.core.netdev_max_backlog=30000
```

CPU Affinity

Associating the test software application with the same CPU as the NIC driver can provide a substantial performance improvement. To do this, you must first identify which interrupt request (IRQ) line is associated with your NIC by looking at the /proc/interrupts file. Then, enter the CPU you want in the /proc/irq/#/smp_affinity file, where # is the IRQ number associated with the NIC. Make sure the irqbalance service is not running. Finally, when the application is run you can use the *taskset* command to set the CPU affinity.[2]

[2] For more information on CPU affinity tuning, please see the guide at **http://fasterdata.es.net/host-tuning/interrupt-binding/**

DMA Ring Buffer Sizes

The pre-set maximum is 8192 packets. The default is 256 packets. For purposes of LAN testing, using the smallest possible ring size provides the best results.

Coalescing

Using interrupt coalescing is a trade-off between lower latency and reducing CPU overhead. By default, the adaptive interrupt moderation (adaptive-rx) feature is enabled**.** Not all drivers support this setting, however, so manual tuning should be done for each situation, based on the environment. Fortunately, the Mellanox NICs support the adaptive feature and dynamically optimize the settings for reach flow.

The coalescing settings can be adjusted using the *ethtool* command.

Process Priority

On a heavily used or multi-user system, it might be necessary to set a higher scheduling priority using a command such as *nice*.

Note: The lower the *nice* value, the higher the priority of the process.

New API (NAPI)

This newer, improved API for packet processing helps mitigate interrupts. NAPI is supported on newer Linux kernels (2.5+, generally) and requires driver code modification. For the Mellanox 40GbE NICs, NAPI is enabled by default. It can be disabled (or re-enabled) via driver compilation as a CFLAG setting. Please refer to your driver documentation for further information.

TX Queue Length

This queue parameter is mostly applicable for high-speed WAN transfers. For low-latency networks, the default setting of 1000 is sufficient. The receiving end is configured with the *sysctl* setting *net.core.netdev_max_backlog*. The default for this setting is also 1000 and does not need to be modified unless there is significant latency.

Firewalls

Firewall software programs such as *iptables* are built into the kernel and consume CPU resources. For the purpose of obtaining the highest performance rates during testing, the system firewalls are disabled on client and server.

Pause Frames

A lossless fabric is a key requirement for RDMA testing. Priority Flow Control (PFC) can provide this through the use of pause frames. Ensure that pause frames are enabled on the system and on the switch by using the *ethtool* command.

V. Testing Results

Our throughput testing was performed using a data-sink network application called *nuttcp*, which is an enhanced version of the well-known *ttcp* software. This application was chosen because it provides per-interval metrics such as packet loss. It was also important to ensure that data was not read or written to the disk during the test, because that could become an additional performance bottleneck. The TCP protocol was used for metrics, as all data transfer applications use it as a transport protocol.

With both RDMA and traditional Ethernet testing, we used 65KB packets and single-stream flows. Traditional Ethernet results varied slightly test to test, even when identical parameters were used. There was a deviation of around 300 Mbps between tests, which was a result of other system processes running in parallel with our tests. However, each test was run multiple times to ensure the results were consistent. Also, the system's route cache was flushed between each test to ensure no metrics or settings were caches, which can give false improvement rates.

Offloading

Table 2. Performance Implications of Offloading Settings

Offloading Type	On - MTU 1500	On - MTU 9000	Off - MTU 1500	Off - MTU 9000
TCP Segmentation Offloading (TSO)	15100 Mbps	19100 Mbps	12900 Mbps	19000 Mbps
Rx Checksumming [1]	15100 Mbps	19100 Mbps	7500 Mbps	17300 Mbps
Tx Checksumming [2]	15100 Mbps	19100 Mbps	10000 Mbps	50 Mbps
Large Receive Offload (LRO)	15100 Mbps	19100 Mbps	13300 Mbps	18700 Mbps
Scatter-Gatter (SG) [3]	15100 Mbps	19100 Mbps	10300 Mbps	70 Mbps
Generic Segmentation Offload (GRO)	15100 Mbps	19100 Mbps	12800 Mbps	19100 Mbps

Note: Standard deviation is +/- 300 Mbps.

1. Changes to Rx Checksumming also include the same change to GRO.
2. Changes to Tx Checksumming also include the same change to SG and TSO.
3. Changes to SG also include the same change to TSO.

Our results show that the offloading capabilities play a larger role with smaller frame size, as shown in Table 2. With jumbo frames, the changes were negligible except for the SG-related parameters. It is not known if the poor performance is related to 40 GbE or is an issue with the SG code itself. In the past, bugs have been noted with TSO and therefore it is generally recommended that it be disabled.

The key meta parameters for 40 GbE tuning appear to be offloading Tx/Rx checksumming capabilities. With small maximum transmission units (MTUs), there are also many retransmits due to the extra overhead associated with the interrupts.

Kernel Tuning

For the kernel settings we tested to improve throughput on the high-speed LAN, only the timestamps setting improved performance by reducing CPU overhead.

Note: If 40 GbE is used in a WAN environment, the kernel settings should be tuned to better be able to handle packet loss and larger window buffer sizes.

Table 3. Performance Implications of Kernel Settings

Kernel Setting	Affect on performance
tcp_low_latency=1	None
tcp_timestamps=0	Improved performance 300-400 Mbps.
tcp_sack=0	Negligible.
netdev_max_backlog=30000	Negligible.

CPU Affinity

For testing, CPU 0 was chosen for interrupt binding. Assigning the test software application to also use CPU 0 on the client and server yielded the best results. Testing was also done with IRQ balancing; as expected, this resulted in many packet retransmits. To further demonstrate the impact of binding, the application was set to run on different CPUs than CPU 0 (both client and server), using the *nuttcp* parameters *-xcs#* and *-xc#* for server/client CPU affinity settings. The results are shown in Table 4:

Table 4. Performance Implications of CPU Affinity

Client/Server CPU	Rate	Packet Loss (30 sec)
0/0	18500 - 19500 Mbps	0
2/2	11600 - 17400 Mbps	43447
6/6	7100 - 7400 Mbps	15149
13/13	12000 - 12200 Mbps	44891

It is easy to monitor CPU usage and find the bottleneck by using the *top* command. Once *top* is running, press the "*1*" key to see an individual list of each CPU and its utilization. For example, when running *nuttcp* with CPU 0 affinity, *top* shows the following output:

```
(on server)
Cpu0:  0.3%us, 49.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi, 50.0%si,  0.0%st

(on client)
Cpu0:  0.9%us, 50.7%sy,  0.0%ni, 35.6%id,  0.0%wa,  0.0%hi, 12.9%si,  0.0%st
```

In this case, the server on CPU 0 is at 100% utilization. Approximately half of the processing is soft interrupts (represented by "*si*") and the other half is used by the system ("*sy*"); a fraction is associated with userland space ("*us*"). The key to tuning in this scenario is to lower the IRQ interrupts so that more processing can be used by the system.

DMA Ring Buffer Sizes

The number of packets held in the ring queue can further increase throughput rates, as shown in Table 5. The smaller the queue, the faster the packets are processed, and therefore the latency is lower.

Note: in WAN environments, larger ring sizes are preferred to better shape and buffer the data flows with higher latency.

Table 5. Performance Implications of DMA Ring Size

Ring Size (Packets)	Throughput
16	21000 Mbps
1024	18700 Mbps
4096	18100 Mbps
8192	18100 Mbps

Coalescing

For the LAN testing environment, it is best to tune interrupt coalescing for reducing interrupts, since latency is already inherently low.

Turning off the adaptive-tx and adaptive-rx features and manually setting higher rates for rx-frames and rx-usecs resulted in erratic results, as shown in Table 6. These settings are best managed using the adaptive technologies that dynamically adjust values to optimize tuning during a flow.

Table 6. Performance Implications of Setting Interrupt Coalescing

Adaptive-rx	Adaptive-tx	rx-usecs	rx-frames	Throughput
On	Off	16	88	21000 Mbps
Off	Off	16	88	110 Mbps
On	On	16	88	21000 Mbps
Off	On	16	88	112 Mbps

Miscellaneous Tuning

Increasing the TX queue length of an interface had no noticeable results; because the testing was done between two hosts on a single switch, there was little need for a queue. As long as the TX queue length is set to the default of 1000, it is enough to handle the output queuing.

VI. Conclusion

Systems today are capable of handling 40 Gb Ethernet with slight optimization tuning. While the best deployment scenario is to use RDMA-enabled applications, the industry still needs time to catch up to this scenario, as common software must be rewritten to support InfiniBand verbs with RoCE. Converged Ethernet, which unites the best parts of the HPC and commodity networks worlds, is therefore a long-term path. It is currently very limited in use due to the expectation of a lossless fabric (although that is being addressed by new technologies such as Data Center Bridging and Priority Flow Control).

Traditional 40 Gb Ethernet is still very much in the running to become the next-generation technology. Its simple upgrade path requires no software to be modified, and it can more than double the performance of existing 10 GbE products on a per-stream basis. While "out-of-the-box" settings provide acceptable transfer rates, some fine-tuning will allow you to achieve even better rates. In our tests, the following configuration changes provided the best improvements:

- Disable TCP timestamps
- Reduce the DMA Ring buffer to minimum
- Disable *iptables* or any system-level firewall service
- Interrupt coalescing: bind applications to the same CPU as interrupts
- Ensure as much offloading to the NIC as possible: Checksumming, SG, and LRO are all critical
- Jumbo frames are critical; the larger the frame size, the better

The purpose of the settings listed above is simply to reduce CPU overhead and keep latency low. The overall performance of 40 GbE could be improved with the use of larger frame sizes. Generally, jumbo frames are supported at 9000 MTU. This support was introduced with gigabit Ethernet networking; now that the rates have increased 40x, however, the size of the jumbo frame itself has not. It would be encouraging to see vendors supporting jumbo frames of at least 64K MTU. Currently, there is no standardization on the size of large Ethernet frames, and that is hurting the industry's progression; each vendor supports different settings, which limits MTU size to the lowest common MTU end-to-end.

The best-sustained rates achieved in our tests were 39 Gbps with RDMA software, and 21 Gbps with traditional Ethernet. At a sustained rate of 39 Gbps using RoCE, it is possible to transfer approximately 420 TB of data in 24 hours. At a sustained rate of 21 Gbps using traditional Ethernet, it is possible to transfer 225 TB in 24 hours.

Keep in mind that these tests are for pure network performance—in a production environment, there can be other limiting factors, such as the disk read/write speeds or contention for system resources. Ideally, 40 GbE is best suited for two environments:

- High-Performance Computing (HPC) with large data set transfers
- Centralized services in data centers or campuses where 10 GbE systems aggregate data, such as backup or file servers

While 40 GbE networking is the logical next step beyond 10 GbE, it imposes the same problems that were manifested by upgrading system NICs from 1 GbE to 10 GbE. Largely, this is due to various queuing and shallow buffer issues across the network—the system sends out data faster than the lower-speed WAN links and older networking equipment can handle, causing excessive packet loss and rate drops. It is expected that the upgrade path to 40 GbE will further exasperate this known problem, therefore, 40 GbE system deployments should be carefully architected to avoid such hazards. In an optimal environment, 40 GbE will provide 2x to 4x performance increases over existing 10 GbE technology.

VII. Appendix

Supporting data tests and their results are listed in this section.

Appendix A – RDMA Test Results

Tests were run using the Mellanox OFED software package, using a simple bandwidth test with a packet default of 65K bytes. Note that no additional tuning was required, and the same results were achieved without having to assign CPUs or process priority. Rate was easily sustained. Although the CPU utilization reached 100%, the utilization was entirely in user space and there was no contention from interrupts, which is seen with traditional Ethernet.

RDMA Test 1

```
# ib_send_bw --run_infinitely  --report_gbits 10.1.1.2

---------------------------------------------------------------------------------------

                    Send BW Test

 Dual-port       : OFF          Device         : mlx4_0
 Number of qps   : 1            Transport type : IB
 Connection type : RC
 TX depth        : 128
 CQ Moderation   : 100
 Mtu             : 4096B
 Link type       : Ethernet
 Gid index       : 0
 Max inline data : 0B
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet

---------------------------------------------------------------------------------------

local address: LID 0000 QPN 0x004c PSN 0xa8431e
GID: 254:128:00:00:00:00:00:00:02:02:201:255:254:51:18:240
remote address: LID 0000 QPN 0x004b PSN 0x5edb12
GID: 254:128:00:00:00:00:00:00:02:02:201:255:254:51:19:80

---------------------------------------------------------------------------------------

 #bytes     #iterations    BW peak[Gb/sec]    BW average[Gb/sec]   MsgRate[Mpps]

 65536      372712         0.00               39.08                 0.074547
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372802         0.00               39.09                0.074565
 65536      372605         0.00               39.07                0.074526
 65536      372682         0.00               39.08                0.074541
 65536      372808         0.00               39.09                0.074566
 65536      372839         0.00               39.10                0.074573
 65536      372838         0.00               39.10                0.074572
 65536      372841         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372841         0.00               39.10                0.074573
 65536      372841         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00               39.10                0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372841         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573
 65536      372842         0.00                39.10               0.074573


Cpu4  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
```

RDMA Test 2

Another RDMA test was run to determine the optimum packet size:

```
# ib_send_bw -a --report_gbits

---------------------------------------------------------------------------------------

                    Send BW Test

 Dual-port       : OFF          Device         : mlx4_0
 Number of qps   : 1            Transport type : IB
 Connection type : RC
 RX depth        : 512
 CQ Moderation   : 100
 Mtu             : 4096B
 Link type       : Ethernet
 Gid index       : 0
 Max inline data : 0B
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------

 local address: LID 0000 QPN 0x0060 PSN 0xaabfb2
 GID: 254:128:00:00:00:00:00:00:02:02:201:255:254:51:18:240
 remote address: LID 0000 QPN 0x0061 PSN 0x43a77e
 GID: 254:128:00:00:00:00:00:00:02:02:201:255:254:51:19:80

---------------------------------------------------------------------------------------
 #bytes         #iterations    BW peak[Gb/sec]    BW average[Gb/sec]    MsgRate[Mpps]

 2              1000            0.00              0.07                  4.419106
 4              1000            0.00              0.16                  4.942357
 8              1000            0.00              0.29                  4.603339
 16             1000            0.00              0.61                  4.780794
 32             1000            0.00              1.18                  4.603163
 64             1000            0.00              2.51                  4.902224
 128            1000            0.00              4.68                  4.569052
 256            1000            0.00              9.90                  4.831632
 512            1000            0.00              18.73                 4.571882
 1024           1000            0.00              29.74                 3.630029
 2048           1000            0.00              34.32                 2.094550
 4096           1000            0.00              36.71                 1.120326
 8192           1000            0.00              37.88                 0.578001
 16384          1000            0.00              38.18                 0.291316
 32768          1000            0.00              38.75                 0.147832
 65536          1000            0.00              38.94                 0.074267
 131072         1000            0.00              39.00                 0.037195
 262144         1000            0.00              39.05                 0.018618
 524288         1000            0.00              39.06                 0.009313
 1048576        1000            0.00              39.07                 0.004657
 2097152        1000            0.00              39.08                 0.002329
 4194304        1000            0.00              37.05                 0.001104
```

In the above output, it is clear that a threshold occurs, where increasing the packet size offers no additional throughput improvement. A packet between 64K and 128K offers the fastest and most consistent performance rates.

Appendix B – Traditional Ethernet Test Results

To show how interrupt coalescing plays a crucial role in performance, in the following throughput test the application was set to use CPU 1 on both the client and server, while the interrupts were set to use CPU 0.

Ethernet Test 1

```
# ./nuttcp -v -xcs1 -xc1 -T30 10.1.1.2
nuttcp-t: v7.1.6: socket
nuttcp-t: buflen=65536, nstream=1, port=5101 tcp -> 10.1.1.2
nuttcp-t: time limit = 30.00 seconds
nuttcp-t: connect to 10.1.1.2 with mss=8948, RTT=0.305 ms
nuttcp-t: send window size = 524288, receive window size = 524288
nuttcp-t: available send window = 393216, available receive window = 393216
nuttcp-r: v7.1.6: socket
nuttcp-r: buflen=65536, nstream=1, port=5101 tcp
nuttcp-r: interval reporting every 1.00 second
nuttcp-r: accept from 10.1.1.1
nuttcp-r: send window size = 524288, receive window size = 524288
nuttcp-r: available send window = 393216, available receive window = 393216
 1532.6250 MB /   1.00 sec = 12856.4232 Mbps  1857 retrans
 1598.7500 MB /   1.00 sec = 13411.1395 Mbps  1848 retrans
 1600.8125 MB /   1.00 sec = 13428.7900 Mbps  1889 retrans
 1630.9375 MB /   1.00 sec = 13681.0628 Mbps  1940 retrans
 1627.6875 MB /   1.00 sec = 13653.8958 Mbps  1878 retrans
 1633.0000 MB /   1.00 sec = 13699.0078 Mbps  1850 retrans
 1605.8750 MB /   1.00 sec = 13471.0559 Mbps  1908 retrans
 1628.8750 MB /   1.00 sec = 13663.5839 Mbps  1811 retrans
 1634.3750 MB /   1.00 sec = 13710.4877 Mbps  1800 retrans
 1663.3750 MB /   1.00 sec = 13953.3590 Mbps  1910 retrans
 1610.0625 MB /   1.00 sec = 13506.2912 Mbps  1869 retrans
 1608.1875 MB /   1.00 sec = 13490.1308 Mbps  1903 retrans
 1686.0000 MB /   1.00 sec = 14142.9668 Mbps  1860 retrans
 1636.3750 MB /   1.00 sec = 13727.4575 Mbps  1853 retrans
 1666.3750 MB /   1.00 sec = 13978.4548 Mbps  2000 retrans
 1675.3750 MB /   1.00 sec = 14053.7971 Mbps  1913 retrans
 1648.2500 MB /   1.00 sec = 13826.8273 Mbps  1968 retrans
 1659.2500 MB /   1.00 sec = 13918.7143 Mbps  1801 retrans
 1659.2500 MB /   1.00 sec = 13918.8674 Mbps  1939 retrans
 1643.0625 MB /   1.00 sec = 13783.0210 Mbps  1910 retrans
 1675.0625 MB /   1.00 sec = 14051.2741 Mbps  2088 retrans
 1643.1875 MB /   1.00 sec = 13784.1799 Mbps  1938 retrans
 1665.0625 MB /   1.00 sec = 13967.6404 Mbps  2042 retrans
 1634.2500 MB /   1.00 sec = 13708.8770 Mbps  2026 retrans
 1679.3125 MB /   1.00 sec = 14087.1929 Mbps  1933 retrans
 1682.8125 MB /   1.00 sec = 14116.1297 Mbps  2056 retrans
 1653.0000 MB /   1.00 sec = 13866.8128 Mbps  1763 retrans
 1646.6250 MB /   1.00 sec = 13812.6706 Mbps  1959 retrans
 1660.3125 MB /   1.00 sec = 13927.8221 Mbps  1909 retrans
 1660.6250 MB /   1.00 sec = 13930.4436 Mbps  1991 retrans
nuttcp-t: 49248.7500 MB in 30.00 real seconds = 1681019.69 KB/sec = 13770.9133 Mbps
nuttcp-t: retrans = 57412
nuttcp-t: 787980 I/O calls, msec/call = 0.04, calls/sec = 26265.93
nuttcp-t: 0.2user 15.5sys 0:30real 52% 0i+0d 490maxrss 0+3pf 83214+29csw
nuttcp-r: 49248.7500 MB in 30.00 real seconds = 1681020.13 KB/sec = 13770.9169 Mbps
nuttcp-r: 2038427 I/O calls, msec/call = 0.02, calls/sec = 67947.41
nuttcp-r: 0.3user 14.8sys 0:30real 50% 0i+0d 346maxrss 0+21pf 1141404+21csw
```

```
Cpu0  :  0.0%us,  0.0%sy,  0.0%ni, 98.3%id,  0.0%wa,  0.0%hi,  1.7%si,  0.0%st

Cpu1  :  0.0%us, 10.2%sy,  0.0%ni, 89.5%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
```

Ethernet Test 2

The same test was run again, this time with both the application and interrupts set to use CPU 0. No packets are dropped due to lack of interrupts, and there is a smooth, lossless data exchange.

```
# ./nuttcp -v -i1 -xc0 -xcs0 -T30 10.1.1.2
nuttcp-t: v7.2.1: socket
nuttcp-t: affinity = CPU 0
nuttcp-t: buflen=65536, nstream=1, port=5101 tcp -> 10.1.1.2
nuttcp-t: time limit = 30.00 seconds
nuttcp-t: connect to 10.1.1.2 with mss=8960, RTT=0.235 ms
nuttcp-t: send window size = 524288, receive window size = 524288
nuttcp-t: available send window = 393216, available receive window = 393216
nuttcp-r: v7.2.1: socket
nuttcp-r: buflen=65536, nstream=1, port=5101 tcp
nuttcp-r: interval reporting every 1.00 second
nuttcp-r: accept from 10.1.1.1
nuttcp-r: send window size = 524288, receive window size = 524288
nuttcp-r: available send window = 393216, available receive window = 393216
 2408.7500 MB /   1.00 sec = 20205.3321 Mbps     0 retrans
 2476.6250 MB /   1.00 sec = 20775.3947 Mbps     0 retrans
 2479.2500 MB /   1.00 sec = 20797.3940 Mbps     0 retrans
 2491.6250 MB /   1.00 sec = 20901.7880 Mbps     0 retrans
 2488.8750 MB /   1.00 sec = 20878.1759 Mbps     0 retrans
 2475.1250 MB /   1.00 sec = 20762.5004 Mbps     0 retrans
 2475.2500 MB /   1.00 sec = 20764.3588 Mbps     0 retrans
 2479.6875 MB /   1.00 sec = 20800.8976 Mbps     0 retrans
 2479.6250 MB /   1.00 sec = 20800.4149 Mbps     0 retrans
 2493.5625 MB /   1.00 sec = 20918.2086 Mbps     0 retrans
 2489.9375 MB /   1.00 sec = 20886.7128 Mbps     0 retrans
 2494.0625 MB /   1.00 sec = 20921.8172 Mbps     0 retrans
 2491.3125 MB /   1.00 sec = 20898.4977 Mbps     0 retrans
 2494.0000 MB /   1.00 sec = 20921.2930 Mbps     0 retrans
 2491.5625 MB /   1.00 sec = 20900.4485 Mbps     0 retrans
 2480.0625 MB /   1.00 sec = 20804.3970 Mbps     0 retrans
 2486.2500 MB /   1.00 sec = 20856.4269 Mbps     0 retrans
 2476.0000 MB /   1.00 sec = 20769.9649 Mbps     0 retrans
 2475.8750 MB /   1.00 sec = 20768.9579 Mbps     0 retrans
 2473.5625 MB /   1.00 sec = 20749.8707 Mbps     0 retrans
 2474.2500 MB /   1.00 sec = 20755.7624 Mbps     0 retrans
 2474.6875 MB /   1.00 sec = 20759.1626 Mbps     0 retrans
 2475.5000 MB /   1.00 sec = 20766.2898 Mbps     0 retrans
 2484.0000 MB /   1.00 sec = 20836.4480 Mbps     0 retrans
 2482.0000 MB /   1.00 sec = 20820.8998 Mbps     0 retrans
 2479.3750 MB /   1.00 sec = 20798.1098 Mbps     0 retrans
 2480.2500 MB /   1.00 sec = 20806.3860 Mbps     0 retrans
 2479.0625 MB /   1.00 sec = 20795.4884 Mbps     0 retrans
 2477.9375 MB /   1.00 sec = 20786.7374 Mbps     0 retrans
nuttcp-t: 74389.9375 MB in 30.00 real seconds = 2539171.96 KB/sec = 20800.8967 Mbps
nuttcp-t: retrans = 0
nuttcp-t: 1190239 I/O calls, msec/call = 0.03, calls/sec = 39674.56
nuttcp-t: 0.3user 29.5sys 0:30real 99% 0i+0d 502maxrss 0+4pf 17+77csw
nuttcp-r: 74389.9375 MB in 30.00 real seconds = 2539182.20 KB/sec = 20800.9806 Mbps
nuttcp-r: 1236588 I/O calls, msec/call = 0.02, calls/sec = 41219.69
nuttcp-r: 0.5user 26.7sys 0:30real 90% 0i+0d 350maxrss 0+21pf 505948+55csw
```

```
Cpu0  :  1.0%us, 46.5%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi, 52.5%si,  0.0%st
```

Appendix C – Tuning Commands

Commands used to tune and test the system are listed in this section. Commands must be run as the root user.

NIC Offloading

Toggling TCP Segmentation Offloading (TSO):

```
# ethtool -K eth1 tso [on/off]
```

Toggling Large Receive Offload (LRO):

```
# ethtool -K eth1 lro [on/off]
```

Toggling Rx Checksumming (Rx):
Note: *The GRO setting is also automatically toggled with this.*

```
# ethtool -K eth1 rx [on/off]
```

Toggling Tx Checksumming (Tx):
***Note**: SG and TSO are automatically toggled with this.*

```
# ethtool -K eth1 tx [on/off]
```

Toggling Generic Segmentation Offload (GRO):

```
# ethtool -K eth1 gro [on/off]
```

Toggling Scatter-Gather (SG):
***Note**: TSO is automatically toggled with this.*

```
# ethtool -K eth1 sg [on/off]
```

If you get an "Operation not supported" error when you run one of the above commands, you may need to enable the dependent setting and try the command again. Also, be aware that some drivers do not support these settings, so refer to your vendor documentation for support information. You can validate your changes by running `ethtool -k eth1`.

Kernel Tuning Commands

Changes to kernel-level settings are done through the `sysctl` command. The settings are applied immediately.

```
       # sysctl -w net.ipv4.tcp_low_latency=1
       # sysctl -w net.ipv4.tcp_timestamps=0
       # sysctl -w net.ipv4.tcp_sack=0
       # sysctl -w net.core.netdev_max_backlog=30000
```

CPU Affinity

You can set which IRQ is bound to a specific processor:

```
# echo 01 > /proc/irq/76/smp_affinity
```

In addition, you can define which process ID (PID) is affiliated with each process:

```
# taskset -p 01 1334
```

DMA Buffers

DMA Ring buffers are also set with the ethtool utility. You can set RX, RX mini, RX jumbo, or TX:

```
# ethtool -G eth1 fx 1024
```

NIC Coalescing

Adaptive settings are preferred for coalesce buffers, but not all network cards support this newer technology:

```
# ethtool -C eth1 adaptive-rx on
```

Process Priority

If there is competition with other applications running, it may be useful to increase the process priority of the application needing more network resources:

```
# nice -n 2 nuttcp
```

Interface Tx/Rx Queue

Larger transmit queues are needed for high-speed networks. The size of the queue will vary depending on end-to-end latency.

```
Tx: # ifconfig eth0 txqueuelen 10000
Rx: # sysctl -w  net.core.netdev_max_backlog=30000
```

Pause Frames

Pause frames help reduce congestion and packet loss. Set this to auto-negotiation for both transmitting and receiving:

```
# ethtool -A eth1 augoneg on
```