



BCM957608

Ethernet Networking Guide for NVIDIA H100 GPU Clusters

Application Note

Copyright © 2024–2025 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to www.broadcom.com. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

Table of Contents

Chapter 1: Introduction	6
1.1 Intended Audience	6
1.2 Data Flow Path with Peer Memory Direct	7
1.3 Host PCIe Topology for Peer Memory Direct	8
Chapter 2: Peer Memory Direct Configuration with BCM957608	9
2.1 NVIDIA H100 GPU	9
2.1.1 NVIDIA GPU Driver Installation	9
2.1.1.1 Example Instructions for Ubuntu 22.04	9
2.1.2 NCCL Library Installation	11
2.1.3 NVIDIA Fabric Manager Installation	11
2.1.4 Installing NVIDIA GPU drivers, NCCL, and Fabric Manager from the Package Manager	11
2.2 Broadcom Ethernet NIC Software Installation with Peer Memory Direct	12
2.2.1 Using the Broadcom Automated Installer to Update Software and Firmware for Peer Memory Direct	13
2.2.1.1 Example On Using the Broadcom Installer	15
2.2.2 Verifying the Correct Driver and Firmware Versions	15
2.2.3 Verifying the Correct RoCE QOS Configuration	16
2.2.4 Using the Broadcom Installer on a Host with Multiple NICs	17
2.2.5 Manually Compiling the Broadcom Host Software from Source Code for Peer Memory Direct	17
2.2.6 Installing the RoCE QOS Configuration (bnxt_re_conf) pkg Manually	18
2.2.7 Configuring the Peer Memory Driver on Kernels that Do Not Have Inbox Peer Memory Driver Support	18
2.2.8 Installing the NIC Firmware Manually	19
2.2.8.1 Verifying the Correct Driver and Firmware Version	19
2.2.8.2 Configuring RoCE Support	19
2.2.8.3 Enable RDMA Option on the NIC	20
2.2.8.4 Enable RoCE Performance Profile on the NIC	20
2.2.8.5 Enable PCIe Relaxed Ordering on the NIC	20
2.2.8.6 Firmware Based DCBx NVM CFG on NIC	22
2.3 ACS and IOMMU Settings	23
2.3.1 Hosts with AMD CPUs	23
2.3.2 Hosts with Intel CPUs	23
2.3.3 Host Memory	23
2.4 Configuring Routing for the BackEnd Network	24
2.4.1 Single Leaf switch Topology with 24-bit subnets	24
2.4.1.1 Host 1 netplan file:/etc/netplan/00-installer-config-24bit-subnet-host1.yaml	25
2.4.1.2 Host 2 netplan file:/etc/netplan/00-installer-config-24bit-subnet-host2.yaml	28
2.4.1.3 Ethernet Leaf Switch Port Configuration for 24-bit Subnet Scheme on Dell Z9664 Switch and Supermicro SSE-T8032 Switch Running SONiC OS	31
2.4.1.4 Ethernet Leaf Switch Port Configuration for 24-bit Subnet Scheme on Juniper QFX5240 Switch	34

2.4.2 Single Leaf Switch Topology with 31-bit Subnets	38
2.4.2.1 Host 1 netplan file:/etc/netplan/00-installer-config-host1.yaml	40
2.4.2.2 Host 2 netplan file:/etc/netplan/00-installer-config-host2.yaml	43
2.4.2.3 Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Dell Z9664 Switch Running SONiC OS.....	47
2.4.2.4 Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Juniper QFX5240 Switch	49
2.4.3 Confirm Routing Between Different NICs Across Different Hosts.....	51
2.5 Ethernet Switch Configuration for QoS and Congestion Control	52
2.5.1 Example: Arista 7060CX (DCQCN-P at 400G) and 31-bit Subnet Scheme	54
2.5.2 Example: Dell Z9664 Switch and Supermicro SSE-T8032 Switch Running SONiC OS and 31-bit Subnet Scheme	55
2.5.3 Example: Juniper QFX5240 Switch and 31-bit Subnet Scheme.....	57
2.6 Final Checks and Settings for Optimal Performance	59
2.7 Installing and Compiling Perfest with NVIDIA GPU Support	61
2.8 Validating Peer Memory Direct Support with Perfest	62
2.8.1 Using an NVIDIA GPU	62
2.8.2 Example – ib_write_bw test using Broadcom NIC with NVIDIA GPU	63
Chapter 3: System BIOS	65
3.1 BIOS Setting Recommendations	65
Chapter 4: Atlas2 PCIe Switch Configuration	66
Chapter 5: Debugging Thor2 NIC	67
5.1 Frequently Asked Questions and Troubleshooting.....	67
5.2 BCM_SOSREPORT	70
Chapter 6: Installing NVIDIA GPU Drivers	71
Chapter 7: Running NCCL Collectives	72
7.1 Setting up the Environmental Variable	72
7.2 Installing UCX for NVIDIA GPUs	73
7.3 Installing Open MPI for NVIDIA GPUs	73
7.4 Compiling NCCL Tests	74
7.5 Single Node NCCL Collectives	74
7.6 MultiNode NCCL Collectives using Open MPI.....	76
Chapter 8: NIC and Ethernet Switch Configuration	78
Appendix A: Compiling Broadcom NIC Software from Source.....	79
A.1 Ubuntu: Install Script for NIC Software (Compiling from Source Code)	79
A.2 RHEL: Install Script for NIC Software (Compiling from Source Code)	81
Appendix B: Script for Disabling ACS.....	84
B.1 Disable PCIe ACS	84
B.2 List all PCIe Devices that Support ACS	85
B.3 List all PCIe Devices with ACS Enabled	85

B.4 List all PCIe Devices with ACS Disabled 85

Appendix C: PCIe Link Speed and Width Related Scripts 86

C.1 Displaying the Link Speed and Link Width of Every PCIe Component 86

C.2 Display Every PCIe Component with Downgraded Speed or Downgraded Width 86

Appendix D: References 87

D.1 Broadcom Ethernet Network Adapter User Guide 87

Appendix E: Terminology 88

Revision History 89

957608-AN304; July 29, 2025 89

957608-AN303; July 9, 2025 89

957608-AN302; April 15, 2025 89

957608-AN301; March 20, 2025 89

957608-AN300; October 22, 2024 89

Chapter 1: Introduction

Broadcom Ethernet network interface controllers (NICs) support Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE). RoCE enables Direct Memory Transfer across GPU or CPU memory on two different hosts, bypassing the CPU on the hosts. RoCE is completely offloaded to the NIC hardware. It, therefore provides high bandwidth, low latency, and low-overhead communication to applications.

RoCE is extensively used in Artificial Intelligence (AI), Machine Learning (ML), and High Performance Computing (HPC) applications. AI/ML and HPC applications follow a distributed and parallel computing paradigm where computations are performed across a large number of GPUs or CPUs spread across a large number of hosts, connected via an Ethernet network. These applications move massive amounts of data across hosts and require high-bandwidth and low-latency transport. RoCE-capable Broadcom NICs provide such a transport, which is completely offloaded to the NIC hardware.

1.1 Intended Audience

This document describes:

- How to use Broadcom NICs and NVIDIA GPUs for [Data Flow Path with Peer Memory Direct](#) on Linux-based hosts.
- How to configure Ethernet switches for RoCE/Peer Memory Direct
- How to run benchmark tests with collective communications libraries such as NCCL.

The intended audience of this document are those looking to deploy AI/ML clusters using Linux-based hosts over an Ethernet network and then run GPU-based collectives and training/inference models on the cluster.

Specifically, this document focuses on the use of Broadcom's 400G BCM957608 NICs and NVIDIA H100 GPUs in a clustered environment over an Ethernet network. The cluster can consist of multiple hosts and multiple Ethernet switches. Each host contains multiple NICs and multiple GPUs.

The document provides details on how to:

- Install and Configure Software and Firmware for Broadcom NICs
- Install GPU Software:
 - CUDA and NCCL for NVIDIA GPUs
- Configure routing on the hosts for RoCE/Peer Mem Direct
- Configure Ethernet switches for RoCE/Peer Mem Direct
- Run [RDMA perftest](#) with CUDA support on NVIDIA GPUs and Broadcom NICs
- Install OpenMPI and UCX with CUDA support
- Run NCCL collectives on NVIDIA GPUs and Broadcom NICs

For more information on CUDA and NCCL, please refer to the links below:

- CUDA: <https://developer.nvidia.com/cuda-downloads>
- NCCL: <https://developer.nvidia.com/nccl/nccl-download>

NVIDIA GPU-related tests mentioned in this document have been verified on:

- Ubuntu 22.04 running 6.5 kernel.

Broadcom's BCM957608 (Thor2) family of NICs support RoCE and Peer Memory Direct. The BCM957608 family supports a max speed of 400Gbps. These NICs are available in the PCIe and the OCP3.0 Form Factor. [Table 1](#) provides a list of Broadcom Thor2 NIC part numbers.

Table 1: Broadcom 400G NIC Part Numbers

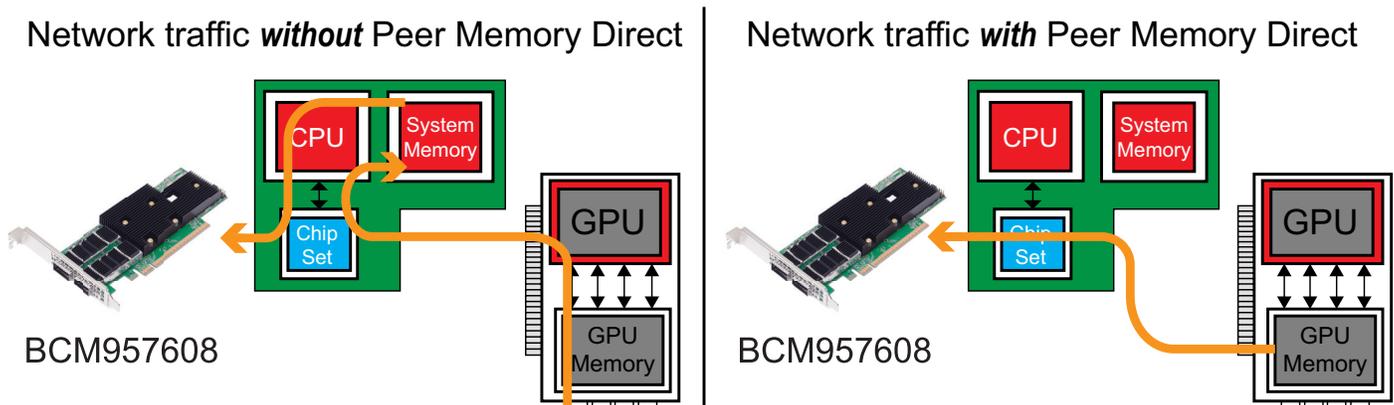
Part Number	Form Factor	Ports	Connector
BCM957608-P1400GD	PCIe	1x 400G	QSFP112-DD
BCM957608-N1400GD	OCP3.0	1x 400G	QSFP112-DD
BCM957608-P2200G	PCIe	2x 200G (default) Can be configured as 1x400G	QSFP112
BCM957608-N2200G	OCP3.0	2x 200G (default) Can be configured as 1x400G	QSFP112

1.2 Data Flow Path with Peer Memory Direct

AI/ML training and inference models require a large number of GPUs for computation and consume massive amounts of data. These GPUs are spread across several hosts in a cluster and connected via Ethernet NICs and Ethernet switches. Using the Peer Memory Direct feature, which is based on RoCE, GPUs on different hosts can exchange data from each other's GPU memory without any CPU involvement.

Without Peer Memory Direct, RoCE can still be used to transfer data across CPU memory on different hosts without any CPU involvement, but the CPU would then have to transfer the data from its memory to the GPU memory. Peer Memory Direct makes use of PCIe peer-to-peer transfers to transfer data between the NIC and the GPU directly, bypassing the CPU.

Figure 1: Data Flow Path with and without Peer Memory Direct



1.3 Host PCIe Topology for Peer Memory Direct

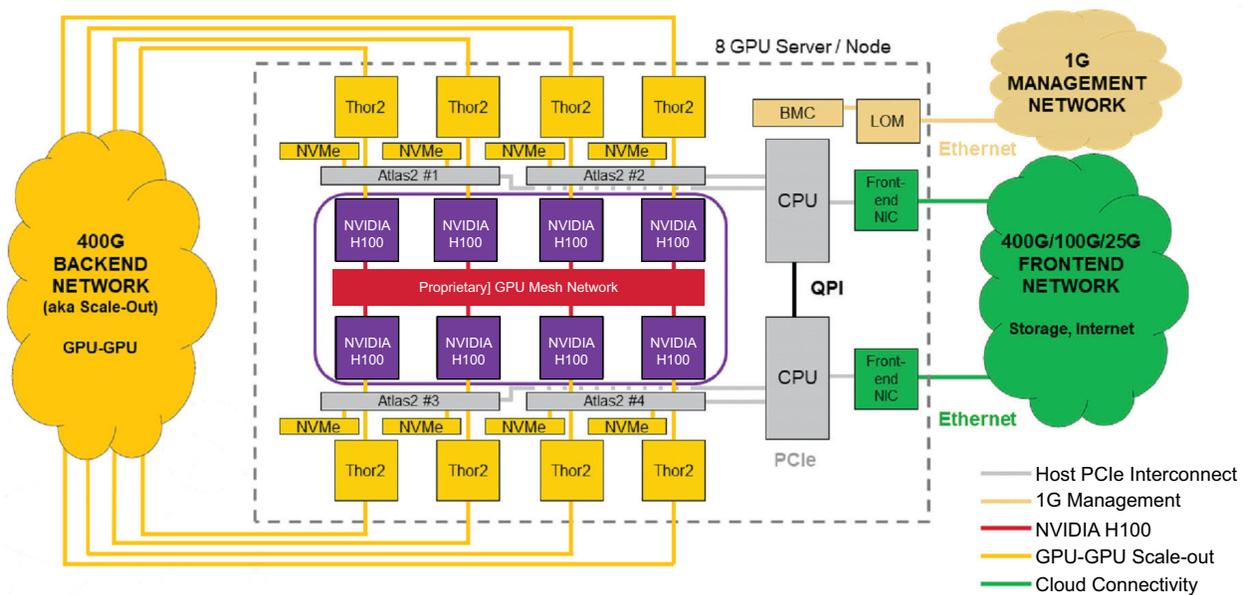
To get the best performance for Peer Memory Direct, the PCIe slot selection for the GPU and the NIC on a host is essential. Peer Memory Direct works via PCIe peer-to-peer transfers, where data is directly transferred between the GPU and the NIC over the PCIe bus.

A typical host used in AI/ML clusters has multiple NICs, multiple GPUs, and multiple PCIe switches per host.

For PCIe peer-to-peer transfers to work, the GPU and the NIC should be connected to the same PCIe switch and the PCIe Access Control Service (ACS) should be disabled on the PCIe switch. If this is not true, then the data is transferred across the NIC and the GPU via the CPU root complex and the benefit of Peer Memory Direct is lost.

A typical NIC, GPU, and PCIe switch configuration inside an AI/ML host is shown below. In this host, there are eight NICs (Thor2), eight GPUs (H100), and four PCIe switches (Atlas2). Each NIC is paired with a GPU, and two NICs and two GPUs share a PCIe switch.

Figure 2: Typical NIC, GPU, and PCIe Switch Configuration Inside a AI/ML Host



Chapter 2: Peer Memory Direct Configuration with BCM957608

2.1 NVIDIA H100 GPU

This section provides information on configuring the BCM957608 with NVIDIA H100.

2.1.1 NVIDIA GPU Driver Installation

See <https://developer.nvidia.com/cuda-downloads> and download the CUDA tool kit installer based on the OS type, architecture, distribution, and installer type.

In the following example, CUDA 12.6, Ubuntu 22.04 x86_64, and runfile (local) installer are used.

2.1.1.1 Example Instructions for Ubuntu 22.04

To install the NVIDIA GPU driver in Ubuntu 22.04, install CUDA using the following commands:

```
wget https://developer.download.nvidia.com/compute/cuda/12.6.0/local_installers/cuda_12.6.0_560.28.03_linux.run
```

```
sudo sh cuda_12.6.0_560.28.03_linux.run
```

```
=====
= Summary =
=====
```

```
Driver:    Installed
Toolkit:   Installed in /usr/local/cuda-12.6/
```

Please make sure that

- PATH includes /usr/local/cuda-12.6/bin
- LD_LIBRARY_PATH includes /usr/local/cuda-12.6/lib64, or, add /usr/local/cuda-12.6/lib64 to /etc/ld.so.conf and run ldconfig as root

To uninstall the CUDA Toolkit, run cuda-uninstaller in /usr/local/cuda-12.6/bin

To uninstall the NVIDIA Driver, run nvidia-uninstall

Logfile is /var/log/cuda-installer.log

```
$ nvidia-smi
```

```
+-----+
| NVIDIA-SMI 560.28.03                Driver Version: 560.28.03          CUDA Version: 12.6          |
+-----+-----+-----+-----+-----+-----+
| GPU  Name      Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M. |
|                                           |              | MIG M. |
+-----+-----+-----+-----+-----+-----+
|   0   NVIDIA H100 80GB HBM3      Off | 00000000:18:00:0 Off |             0 |
| N/A   33C    P0              109W / 700W |  1MiB / 81559MiB |    0%      Default |
|                                           |              | Disabled |
+-----+-----+-----+-----+-----+-----+
|   1   NVIDIA H100 80GB HBM3      Off | 00000000:2B:00:0 Off |             0 |
| N/A   36C    P0              113W / 700W |  1MiB / 81559MiB |    0%      Default |
|                                           |              | Disabled |
+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2  NVIDIA H100 80GB HBM3      Off | 00000000:3A:00.0 Off |           0 |
| N/A  36C   P0                111W / 700W | 1MiB / 81559MiB | 0%   Default |
|                                           |                       |           Disabled |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 3  NVIDIA H100 80GB HBM3      Off | 00000000:5E:00.0 Off |           0 |
| N/A  34C   P0                114W / 700W | 1MiB / 81559MiB | 0%   Default |
|                                           |                       |           Disabled |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 4  NVIDIA H100 80GB HBM3      Off | 00000000:84:00.0 Off |           0 |
| N/A  34C   P0                111W / 700W | 1MiB / 81559MiB | 0%   Default |
|                                           |                       |           Disabled |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 5  NVIDIA H100 80GB HBM3      Off | 00000000:9D:00.0 Off |           0 |
| N/A  36C   P0                115W / 700W | 1MiB / 81559MiB | 0%   Default |
|                                           |                       |           Disabled |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 6  NVIDIA H100 80GB HBM3      Off | 00000000:B0:00.0 Off |           0 |
| N/A  36C   P0                113W / 700W | 1MiB / 81559MiB | 0%   Default |
|                                           |                       |           Disabled |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7  NVIDIA H100 80GB HBM3      Off | 00000000:D7:00.0 Off |           0 |
| N/A  33C   P0                111W / 700W | 1MiB / 81559MiB | 0%   Default |
|                                           |                       |           Disabled |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Processes:
| GPU  GI  CI          PID  Type  Process name                      GPU Memory
|      ID  ID                                     Usage
|=====|
| No running processes found
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

NOTE: If there are issues with installing CUDA, it may be due to a compiler mismatch. Ensure that the appropriate GCC compiler for the correct kernel is installed.

In this example, Ubuntu 22.04 with 6.5 kernel that is built with gcc-12 compiler is used. Run the following to update the compiler:

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-12 60 --slave /usr/bin/g++ g++ /usr/bin/g++-12
```

The NVIDIA GPU driver package provides a kernel module, `nvidia-peermem`, which provides direct peer-to-peer read and write access to the NVIDIA GPU's video memory. It allows GPUDirect RDMA-based applications to use GPU computing power with the RDMA interconnect without needing to copy data to host memory.

Currently, there is no service to automatically load `nvidia-peermem`. Load the module manually using the following command:

```
sudo modprobe nvidia-peermem
```

2.1.2 NCCL Library Installation

Installation of the NCCL library can be performed by downloading an installation package for your Linux distribution. The library can also be compiled from source. Compiling the library from source is outside of the scope of this document.

Download the [installation package](#). Note that this requires to have an NVIDIA developer account. Use the following instructions:

```
sudo dpkg -i nccl-local-repo-ubuntu2204-2.22.3-cuda12.6_1.0-1_amd64.deb
sudo cp /var/nccl-local-repo-ubuntu2204-2.22.3-cuda12.6/nccl-local-8EF36A6C-keyring.gpg /usr/share/
keyrings /
sudo dpkg -i nccl-local-repo-ubuntu2204-2.22.3-cuda12.6_1.0-1_amd64.deb
sudo apt update
sudo apt install libnccl2 libnccl-i
sudo /usr/local/cuda/bin/nvcc --version
```

2.1.3 NVIDIA Fabric Manager Installation

If the server has an NVswitch that connects multiple NVLink-capable GPUs, the Fabric Manager package must be installed. To enable NVLink peer-to-peer support, the GPUs must register with the NVLink fabric. The Fabric Manager installation package installs the required core components and registers the daemon as a system service called `nvidia-fabricmanager`.

GPU fabric registration status can be found by using the following command:

```
nvidia-smi -q -i 0 | grep -i -A 2 Fabric
    Fabric
        State                : In Progress
        Status                : N/A
```

Download the `nvidia-fabricmanager` pkg from the following link. The Fabric Manager must correspond to the OS and the GPU driver version that is running on the system. See the following link:

<https://developer.download.nvidia.com/compute/cuda/repos/>

```
dpkg -i nvidia-fabricmanager-560_560.28.03-1_amd64.deb
systemctl enable nvidia-fabricmanager
systemctl start nvidia-fabricmanager
```

```
nvidia-smi -q -i 0 | grep -i -A 2 Fabric
    Fabric
        State                : Complete
        Status                : Success
```

2.1.4 Installing NVIDIA GPU drivers, NCCL, and Fabric Manager from the Package Manager

In addition to the previous method for installing NVIDIA GPU drivers, NCCL, and NVIDIA Fabric Manager software. Using the package manager like apt, install NVIDIA GPU drivers, NCCL, and NVIDIA Fabric Manager using the following steps.

The following steps apply to Ubuntu 22.04:

```
sudo apt-get install linux-headers-$(uname -r)
```

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb
sudo apt update
sudo apt install cuda-toolkit-12-6 #or just cuda-toolkit to allow auto upgrades
sudo apt install nvidia-open
sudo apt install nvidia-gds

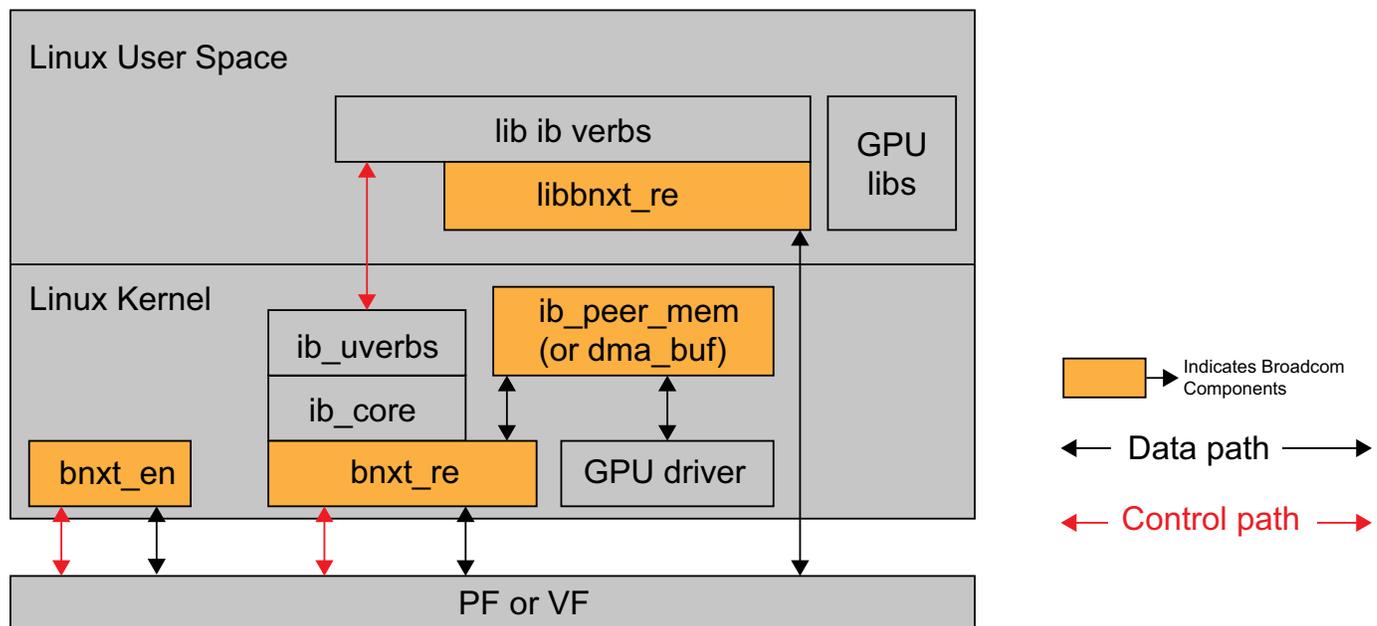
#add the following to .bashrc
export PATH=${PATH:+:${PATH}}:/usr/local/cuda/bin
export LD_LIBRARY_PATH=/usr/local/cuda/lib64/${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
sudo apt install -y libnccl2 libnccl-i
sudo apt install nvidia-fabricmanager-560sudo systemctl enable nvidia-fabricmanagersudo systemctl start nvidia-fabricmanager
```

2.2 Broadcom Ethernet NIC Software Installation with Peer Memory Direct

The host software components required for Broadcom RoCE are as follows:

- Ethernet kernel driver (`bnxt_en`)
- RoCE kernel driver (`bnxt_re`)
- RoCE userspace library (`libbnxt_re`)
- RoCE QOS configuration pkg (`bnxt_re_conf`)

For Peer Memory Direct, an additional kernel driver (`ib_peer_mem`) is required. The `ib_peer_mem` driver interfaces with the GPU driver for Peer Memory Direct.



Broadcom's Ethernet NIC software can be distributed and installed in a variety of approaches listed below:

- Automated installer: Installs all required software and firmware using a single command line
- Dynamic Kernel Module Support (DKMS) format for the kernel drivers

- Source RPM and Binary RPM format for the kernel drivers
- Source Code for the kernel drivers and the RoCE userspace library
- Debian, RPM, and Source Tarball for `bnxt_re_conf` (udev rules and scripts for setting up RoCE QoS parameters – PFC, CC, RoCE and CNP DSCP values, and so forth)

2.2.1 Using the Broadcom Automated Installer to Update Software and Firmware for Peer Memory Direct

Broadcom provides a tarball for every NIC GA release. The release tarball contains all required software and firmware for every NIC part number, and also an automated installer that can be used to install the required software and firmware.

The tarball for the latest GA release is publicly available from the downloads tab of the following links:

<https://www.broadcom.com/products/ethernet-connectivity/network-adapters/p1400g>

or

<https://www.broadcom.com/products/ethernet-connectivity/network-adapters/p2200g>

Besides the public location, customers can contact Broadcom for the GA version of any release not available on the public website.

The installer provides different install options including assigning an IP address, netmask, and MTU size for an Ethernet interface. The installer requires Internet access to download any prerequisite packages from the Linux distribution's package manager for building the required NIC host software.

Installer Options

```
$ sudo bash ./install.sh -h
Usage: install.sh [opts]
-h, --help: Shows this help message and exit
-v, --verbose: Shows all commands run and status information
--uninstall: Uninstalls packages and boot-time configuration of the specified devices (-d)
-i DEVICES [DEVICES ...]: Specifies a network interface. PCI addresses or interface names (
eno1np0,eno2np1 )are allowed.
-A: Applies on all supported network interfaces. All interface names ( eno1np0,eno2np1 )are selected.
-a IP_ADDRESS [IP_ADDRESS ...]: Specifies an IP address for the previous interface
-n NETMASK [NETMASK ...]: Specifies the netmask for the previous interface. Default: 255.255.255.0.
syntax: -a <address> -n <netmask>
-l IP_ADDRESS_PREFIX: Configures interface IP address as <prefix>.(last octet of public interface)
-m MTU, --mtu: MTU Interface MTU. Default: 1500. syntax: -a <address> -m <MTU>
-w, --no-firmware: Do not install firmware
-s RELEASE_SOURCE, --source RELEASE_SOURCE: Path to release files. Default: ../
-o {ECN,PFC,ECNPFC,NOOP}, --cc_mode {ECN,PFC,ECNPFC,NOOP}: RoCE congestion control mode. Default: NOOP
-r ROCE_PRI, --roce_pri ROCE_PRI: RoCE priority. Must set -o to take effect. Default: 3
-d ROCE_DSCP, --roce_dscp ROCE_DSCP: RoCE Packet DSCP value. Must set -o to take effect. Default: 26
-c CNP_PRI, --cnp_pri CNP_PRI: RoCE CNP Packet Priority. Must set -o to take effect. Default: 7
-p CNP_DSCP, --cnp_dscp CNP_DSCP: RoCE CNP DSCP value. Must set -o to take effect. Default: 48
-b ROCE_PCT, --roce_pct ROCE_PCT: RoCE Bandwidth percentage for ETS configuration. Must set -o to take
effect. Default: 90
-2, --l2: Do not install/configure RoCE. Install IP driver only
-q VLAN, --vlan VLAN: Specifies VLAN tag to use instead of DSCP for RoCE
-g, --peermem: Enables peer-mem capable drivers -f, --force Force installation, even if it is a
owngrade, or if versions already match
-t, --nictune: Runs nictune tool to tune the specified interface for optimal throughput
```

`-k, --bcm_sosreport`: Runs `bcm_sosreport` tool to collect system information for support purposes

The installer `README.md` is located inside the release zip file in the directory: `utils/linux_installer`. The `README` provides info on the different installer options.

The following example shows how to use the installer for the NIC 234 GA release.

The main option used by the installer is `-i` which refers to the interface on which the software and firmware must be installed. The interface argument can be specified either as:

- Ethernet interface name
- BDF (Bus, Device, Function) of the PCIe interface for the NIC

The PCIe BDF is useful in cases when no Broadcom Ethernet driver (`bnxt_en`) exists on a host and therefore the NIC interface does not have a name.

The `-g` option of the installer is used to install Peer Memory Direct drivers.

NOTE: If kernel stack already supports peer memory APIs, the installer skips installing the Broadcom peer memory module.

2.2.1.1 Example On Using the Broadcom Installer

```
cd $HOME

# Download the tarball

BCM_TARBALL=bcm_234.1.124.0b.tar.gz
# Highlighted item will change depending on the release

tar -xf ${BCM_TARBALL}

cd $HOME/bcm_234.1.124.0b/utils/linux_installer

sudo bash ./install.sh -v -i 1d:00.0 -f -g
# Highlighted item will change depending on PCIe BDF of the NIC (check with lspci)
```

2.2.2 Verifying the Correct Driver and Firmware Versions

The output of the installer is logged in the file `install.log`. The log file can be checked to see if any error occurred during the installation. After the installer has completed successfully, the version of the drivers loaded in the kernel and the version of the firmware flashed on the NIC should be checked. The best way to check the version of the drivers loaded into the kernel is to check the `dmesg` logs using the following commands:

```
sudo dmesg | grep -i bnx | grep -i ver

modinfo bnxt_en | grep -i ver

modinfo bnxt_re | grep -i ver

modinfo ib_peer_mem | grep -i ver

dkms status
```

The firmware version running on the card can be checked via the Broadcom-provided NICCLI tool.

```
sudo niccli --list_devices
# This command will list available Broadcom NICs and index for each

sudo niccli --dev <index | pci b:d:f | mac | nic interface> show -p
# Highlighted item will change depending on the index or PCIe BDF or MAC

# Example:
sudo niccli --dev 1 show -p
```

In general, reboot the host after the installer has run and re-check the driver version and the FW version. This ensures the correct version of the drivers load after every reboot of the host and the `initramfs/ramdisk` of the host is correctly updated by the installer.

2.2.3 Verifying the Correct RoCE QOS Configuration

The correct RoCE QOS CFG pkg (`bnxt_re_conf`) installation can be verified via the presence of the `/etc/bnxt_re/bnxt_re.conf` file. The file contents should match the configured QOS values. The default values are shown as follows:

```
cat /etc/bnxt_re/bnxt_re.conf
ENABLE_FC=1
FC_MODE=3
ROCE_PRI=3
ROCE_DSCP=26
CNP_PRI=7
CNP_DSCP=48
ROCE_BW=50
UTILITY=4
```

The correct RoCE QOS application on each RoCE NIC can be verified via the following Broadcom-provided NICCLI tool commands:

```
sudo niccli --list_devices
# This command will list available Broadcom NICs and index for each

sudo niccli --dev <index | pci b:d:f | mac> qos --ets --show
# Highlighted item will change depending on the index or PCIE BDF or MAC

# Example:
sudo niccli --dev 1 qos --ets --show

IEEE 8021QAZ ETS Configuration TLV:
    PRIO_MAP: 0:0 1:0 2:0 3:1 4:0 5:0 6:0 7:2
    TC Bandwidth: 50% 50% 0%
    TSA_MAP: 0:ets 1:ets 2:strict
IEEE 8021QAZ PFC TLV:
    PFC enabled: 3
IEEE 8021QAZ APP TLV:
    APP#0:
        Priority: 7
        Sel: 5
        DSCP: 48

    APP#1:
        Priority: 3
        Sel: 5
        DSCP: 26

    APP#2:
        Priority: 3
        Sel: 3
        UDP or DCCP: 4791

TC Rate Limit: 100% 100% 100% 0% 0% 0% 0% 0%
```

In general, reboot the host after the installer has run and re-check the driver version, the firmware version, and the RoCE QOS settings. This ensures the correct version of the drivers load after every reboot of the host and the `initramfs/ramdisk` of the host is correctly updated by the installer.

2.2.4 Using the Broadcom Installer on a Host with Multiple NICs

Most of the hosts used for Peer Memory Direct use multiple NICs and multiple GPUs per host. From the NIC perspective, a single host driver instance is interfacing with the firmware on all the NICs. To flash or upgrade firmware on multiple NICs on the same host, the installer supports specifying multiple interfaces or multiple PCIe Bus, Device, Function. Each interface or PCIe Bus, Device, Function specifies a NIC on which the firmware is to be flashed.

```
cd $HOME

ls bcm_234.1.124.0b.tar.gz
# Highlighted item will change depending on the release

tar xf bcm_234.1.124.0b.tar.gz
# Highlighted item will change depending on the release

cd $HOME/bcm_234.1.124.0b/Utils/linux_installer
# Highlighted item will change depending on the release

sudo bash ./install.sh -v -i 1d:00.0 -i 43:0.0 -i 56:0.0 -i 69:0.0 -i 9f:0.0 -i c3:0.0 -i d5:0.0 -i
e7:0.0 -f -g
# Highlighted items will change depending on the the PCIe BDF of each NIC
```

Reboot the host and check the driver versions and the firmware versions on each NIC as described in [Verifying the Correct Driver and Firmware Versions](#).

2.2.5 Manually Compiling the Broadcom Host Software from Source Code for Peer Memory Direct

The Linux Installer automatically installs the peer mem driver required for the NIC if the driver is not available as part of the kernel. Check [Frequently Asked Questions and Troubleshooting](#) on how to check if the peer mem driver is already part of the kernel. This section outlines the manual compilation of the peer mem driver if a user wants to manually compile the peer mem driver.

The source code required for the kernel drivers is distributed in a tarball with a name that indicates the software release the tarball belongs to. For example, the tarball `netxtreme-peer-mem-234.0.154.0.tar.gz` contains all the kernel driver source code and Makefiles to build the 234.0.154.0 release version of the kernel drivers.

The source code for the `libbnxt_re` library is distributed in a tarball with a name that indicates the software release the tarball belongs to. For example, the `libbnxt_re-234.0.154.0.tar.gz` contains all the software source code and Makefiles to build the 234.0.154.0 release version of the `libbnxt_re` library.

To summarize, the following two source code tarballs are required to be built and installed for the NIC software on any given host

- `netxtreme-peer-mem-a.b.c.d.tar.gz`
- `libbnxt_re-a.b.c.d.tar.gz`

The highlighted items change depending on the release version.

[Appendix A, Compiling Broadcom NIC Software from Source](#) provides Linux shell scripts that can be used to build and install the required software. Two separate scripts are provided for Ubuntu and RHEL-based hosts. The script, along with `netxtreme-peer-mem-a.b.c.d.tar.gz` and `libbnxt_re-a.b.c.d.tar.gz` should be placed in a directory before executing the script. The content of the scripts can be followed to understand the various steps used to build and install the NIC software.

2.2.6 Installing the RoCE QOS Configuration (bnxt_re_conf) pkg Manually

NOTE: If the Broadcom automated installer is used, then the RoCE QOS Configuration pkg (`bnxt_re_conf`) is automatically installed by the installer.

Beginning with the Broadcom 2.31 release, the RoCE QOS CFG package is a standalone pkg. Before the 2.31 release, RoCE QOS CFG was part of the RoCE driver `bnxt_re`. The `bnxt_re_conf` pkg is distributed in a variety of formats (debian, RPM, and source tarball). Depending on the OS distro being used, the appropriate pkg format can be used.

```
cd $HOME

ls bcm_234.1.124.0b.tar.gz
# Highlighted item will change depending on the release

tar xf bcm_234.1.124.0b.tar.gz
# Highlighted item will change depending on the release

cd $HOME/bcm_234.1.124.0b.tar.gz/drivers_linux/bnxt_re/bnxt_re_conf
# Highlighted item will change depending on the release

dpkg -i bnxt_re_conf_234.0.154.0-1_all.deb
# Highlighted item will change depending on the release
or
rpm -Uvh bnxt_re_conf-234.0.154.0-1.noarch.rpm
# Highlighted item will change depending on the release
```

It is recommended to reboot the host if the `bnxt_re_conf` pkg is installed the first time or the contents of the `/etc/bnxt_re/bnxt_re.conf` file are modified. This allows the RoCE QOS settings to be applied correctly to each NIC. After installing the `bnxt_re_conf` pkg, ensure the pkg is correctly installed as shown in [Verifying the Correct RoCE QOS Configuration](#).

2.2.7 Configuring the Peer Memory Driver on Kernels that Do Not Have Inbox Peer Memory Driver Support

Certain Ubuntu kernels have built-in `ib_peer_mem` APIs. These kernels do not require an `ib_peer_mem` module built via the Broadcom Peer Memory Direct Driver pkg. The Driver Pkg Makefile can detect the presence of `ib_peer_mem` APIs in the kernel and skip building the `ib_peer_mem` kernel module. For such kernels, the `nvidia_peermem` module provided via the kernel or the CUDA installation is sufficient. To check if the kernel has inbox peer me support refer to step 3 in [Frequently Asked Questions and Troubleshooting](#).

For the kernels that do not have inbox peer memory driver support, use the following instructions to patch the NVIDIA GPU driver to work with broadcom peer memory direct driver.

1. Retrieve and patch the NVIDIA source `open-gpu-kernel-modules` using the following commands:

```
wget https://github.com/NVIDIA/open-gpu-kernel-modules/archive/refs/tags/$NVIDIA_VERSION.tar.gz
tar xf open-gpu-kernel-modules-$NVIDIA_VERSION.tar.gz
cd open-gpu-kernel-modules-$NVIDIA_VERSION/
```

2. Patch the `kernel-open/confctest.sh` and `nvidia-peermem.Kbuild` file by adding the following line to `compile_test()` in the `kernel-open/confctest.sh` file.

```
check_for_ib_peer_memory_symbols "$BNXT_PEER_MEM_INC" || \
```

3. Replace the definition of `MLNX_OFED_KERNEL` in file `nvidia-peermem.Kbuild` with the following:

```
MLNX_OFED_KERNEL := $(shell echo /lib/modules/`uname -r`/build)
```

4. Replace the definition of `KBUILD_EXTRA_SYMBOLS` in file `nvidia-peermem.Kbuild` with the following:

```
KBUILD_EXTRA_SYMBOLS := $(BNXT_PEER_MEM_INC)/Module.symvers
```

5. Recompile and install the `open-gpu-kernel-modules` using the following commands:

```
dkms remove nvidia/$NVIDIA_VERSION
```

```
export $REL_DIR/drivers_linux/peer_mem/netxtreme-peer-mem-x.y.z/peer_mem ##point to the location
of the source of peer_mem driver
```

```
make modules -j
sudo make modules_install
sudo depmod -a
sudo update-initramfs -u
sudo modprobe ib_peer_mem
sudo modprobe nvidia-peermem
```

2.2.8 Installing the NIC Firmware Manually

NOTE: If the Broadcom automated installer is used, then the firmware is automatically installed by the installer.

The Broadcom NIC firmware is provided in a ".pkg" file and a single file contains all the required firmware for a NIC. The firmware pkg file name contains the NIC part number.

For example, the BCM957608-P1400GD card firmware file is named `BCM957608-P1400GDF00.pkg`.

To list the interface name of all the Broadcom NICs available on a host, use the NICCLI tool provided by Broadcom.

```
sudo niccli --list_devices
# List the interfaces where Broadcom Ethernet cards are recognized
```

To install the FW on a Broadcom NIC, use the following NICCLI command:

```
sudo niccli --dev <index | pci b:d:f | mac> fw --update -f <FW.pkg> --yes
# Highlighted item will change depending on the index or PCIe BDF or MAC

# Example:
sudo niccli --dev 1 install BCM957608-P1400GQF00.pkg
```

A reboot is required when a NIC firmware is flashed. The NICCLI tool output indicates a reboot is required.

2.2.8.1 Verifying the Correct Driver and Firmware Version

After manually installing the drivers from source code and/or installing the NIC firmware on a NIC, it's a good idea to check and ensure the proper versions of the driver are loaded into the kernel and the correct firmware version is installed on the NICs. See [Verifying the Correct Driver and Firmware Versions](#) for the required steps.

2.2.8.2 Configuring RoCE Support

The NICs are default configured for RoCE/Peer Memory Direct. The following NVM CFG parameters control RoCE operation on an NIC and can be verified using the NICCLI tool.

2.2.8.3 Enable RDMA Option on the NIC

To check the option value:

```
sudo niccli --dev <index|pci b:d:f> nvm --getoption support_rdma --scope <pf number>
# First highlighted item will change depending on the index or PCIe BDF or MAC
# Second highlighted item will change depending on the PF

# Example:
sudo niccli --dev 1 nvm --getoption support_rdma --scope 0
```

To enable the option:

```
sudo niccli --dev <index|pci b:d:f> nvm --setoption support_rdma --scope <pf number> --value 1
# First highlighted item will change depending on the index or PCIe BDF or MAC
# Second highlighted item will change depending on the PF

# Example:
sudo niccli --dev 1 nvm --setoption support_rdma --scope 0 --value 1
```

2.2.8.4 Enable RoCE Performance Profile on the NIC

The default performance profile on the NICs is non-ROCE. The default profile is optimized for scenarios where the majority of the traffic handled by the NIC is L2. If the NIC needs to handle a traffic mix where the RoCE traffic is greater than 50% of all traffic (as would be the case for Peer Memory Direct), then the performance profile should be changed to RoCE.

To check the option value:

```
sudo niccli --dev <index | pci b:d:f | mac> nvm --getoption performance_profile
# Highlighted item will change depending on the index or PCIe BDF or MAC

# Example:
sudo niccli --dev 1 nvm --getoption performance_profile
```

To enable the option:

```
sudo niccli --dev <index|pci b:d:f> nvm --setoption performance_profile --value 1
# Highlighted item will change depending on the index or PCIe BDF or MAC
#value 0: Default
#value 1: RoCE
# Example:
sudo niccli --dev 1 nvm --setoption performance_profile --value 1
```

2.2.8.5 Enable PCIe Relaxed Ordering on the NIC

PCIe Relaxed ordering allows PCIe transactions to be completed out of order and results in a performance boost for applications when enabled. However, care should be taken before Relaxed ordering is enabled as it can lead to data corruption for some applications.

To check the option value:

```
sudo niccli --dev <index | pci b:d:f | mac> nvm --getoption pcie_relaxed_ordering
# Highlighted item will change depending on the index or PCIe BDF or MAC

# Example:
sudo niccli --dev 1 nvm --getoption pcie_relaxed_ordering
```

To enable the option:

```
sudo niccli --dev <index|pci b:d:f> nvm --setoption pcie_relaxed_ordering --value 1  
# Highlighted item will change depending on the index or PCIe BDF or MAC
```

Example:

```
sudo niccli --dev 1 nvm --setoption pcie_relaxed_ordering --value 1
```

2.2.8.6 Firmware Based DCBx NVM CFG on NIC

The Broadcom RoCE driver (`bnxt_re`) configures the QOS defaults for the NIC upon loading. However, if the firmware-based DCBx or the FW-based LLDP is enabled via NVM CFG, the RoCE driver does not configure the QOS on the NIC interface.

NOTE: Firmware-based DCBx and FW-based LLDP should be disabled if the RoCE driver needs to configure the QOS for the interface.

To check the option values:

```
sudo niccli --dev <index | pci b:d:f | mac> nvm --getoption dcbx_mode --scope <pf number>
# First highlighted item will change depending on the index or PCIe BDF or MAC
# Second highlighted item will change depending on the PF
```

Example:

```
niccli --dev 3 nvm --getoption dcbx_mode --scope 0
```

```
sudo niccli --dev <index | pci b:d:f | mac> nvm --getoption lldp_nearest_bridge --scope <pf number>
# First highlighted item will change depending on the index or PCIe BDF or MAC
# Second highlighted item will change depending on the PF
```

Example:

```
sudo niccli --dev 1 nvm --getoption lldp_nearest_bridge --scope 0
```

```
sudo niccli --dev <index | pci b:d:f | mac> nvm --getoption lldp_nearest_non_tpmr_bridge --scope <pf number>
```

First highlighted item will change depending on the index or PCIe BDF or MAC

Second highlighted item will change depending on the PF

Example:

```
sudo niccli --dev 1 nvm --getoption lldp_nearest_non_tpmr_bridge --scope 0
```

To disable the options:

```
sudo niccli --dev <index | pci b:d:f | mac> nvm --setoption dcbx_mode --scope <pf number> --value 0
# First highlighted item will change depending on the index or PCIe BDF or MAC
# Second highlighted item will change depending on the PF
```

Example:

```
$ sudo niccli --dev 1 nvm --setoption dcbx_mode --scope 0 --value 0
```

```
$ sudo niccli --dev <index | pci b:d:f | mac> nvm --setoption lldp_nearest_bridge --scope <pf number> --value 0
```

First highlighted item will change depending on the index or PCIe BDF or MAC

Second highlighted item will change depending on the PF

Example:

```
$ sudo niccli --dev 1 nvm --getoption lldp_nearest_bridge --scope 0 --value 0
```

```
$ sudo niccli --dev <index | pci b:d:f | mac> nvm --setoption lldp_nearest_non_tpmr_bridge --scope <pf number> --value 0
```

First highlighted item will change depending on the index or PCIe BDF or MAC

Second highlighted item will change depending on the PF

Example:

```
$ sudo niccli --dev 1 nvm --getoption lldp_nearest_non_tpmr_bridge --scope 0 --value 0
```

2.3 ACS and IOMMU Settings

For Peer Memory Direct to work optimally, PCIe Access Control Services (ACS) needs to be disabled. ACS is a PCIe switch setting and needs to be disabled on the PCIe switch connecting the NIC and the GPU.

Additionally, for optimum Peer Memory Direct performance, the IOMMU on the host needs to be disabled or put in the Pass Through (PT) mode.

2.3.1 Hosts with AMD CPUs

The IOMMU should be configured in PT mode via the kernel command line. The kernel parameters to use with AMD CPU-based hosts are `amd_iommu=on iommu=pt`.

Sample kernel command line for hosts with AMD CPU and IOMMU in PT mode:

```
BOOT_IMAGE=/boot/vmlinuz-5.15.0-102-generic root=UUID=8d0ffb16-6f01-44c2-8e16-18bb37d87392 ro
pci=realloc=off amd_iommu=on iommu=pt
```

2.3.2 Hosts with Intel CPUs

The IOMMU can be configured in PT mode via the kernel command line. The kernel parameters to use with Intel CPU-based hosts are `intel_iommu=on iommu=pt`.

Sample kernel command line for hosts with Intel CPU and IOMMU in PT mode:

```
BOOT_IMAGE=/boot/vmlinuz-5.15.0-102-generic root=UUID=8d0ffb16-6f01-44c2-8e16-18bb37d87392 ro
pci=realloc=off intel_iommu=on iommu=pt
```

On some hosts, ACS and IOMMU can be disabled via the BIOS as well. Check with the host/BIOS vendor on how ACS and IOMMU can be configured via the BIOS.

See [Appendix B, Script for Disabling ACS](#) for a bash shell script that can be used to disable ACS on a host.

If the host does not support disabling ACS via the BIOS, then ACS has to be disabled after every host reboot and the shell script in [Appendix B, Script for Disabling ACS](#) can be used.

2.3.3 Host Memory

AI/ML and HPC applications typically require a lot of RAM. 2 Terabytes or more of RAM is typically recommended.

2.4 Configuring Routing for the BackEnd Network

Most hosts used for AI/ML training, inference as well as AI/ML applications have multiple NICs and multiple GPUs per host. GPU collectives such as NCCL require that any NIC on any host should be able to communicate with any other NIC on any host in the cluster.

When a host has multiple NICs in the same IP subnet, an Address Resolution Protocol (ARP) flux problem can happen on Linux-based hosts. Linux may respond to an ARP request on a different interface (IP address) than the IP address carried in the ARP request. This causes the RDMA stack to incorrectly map the remote IP address to the wrong RDMA device.

There are a few solutions to this problem.

Using the `arp_ignore` and `arp_announce` `sysctl` settings below, one can instruct Linux to send ARP replies on the interface targeted in the ARP request.

```
sudo sysctl -w net.ipv4.conf.all.arp_announce=1
sudo sysctl -w net.ipv4.conf.all.arp_ignore=2
```

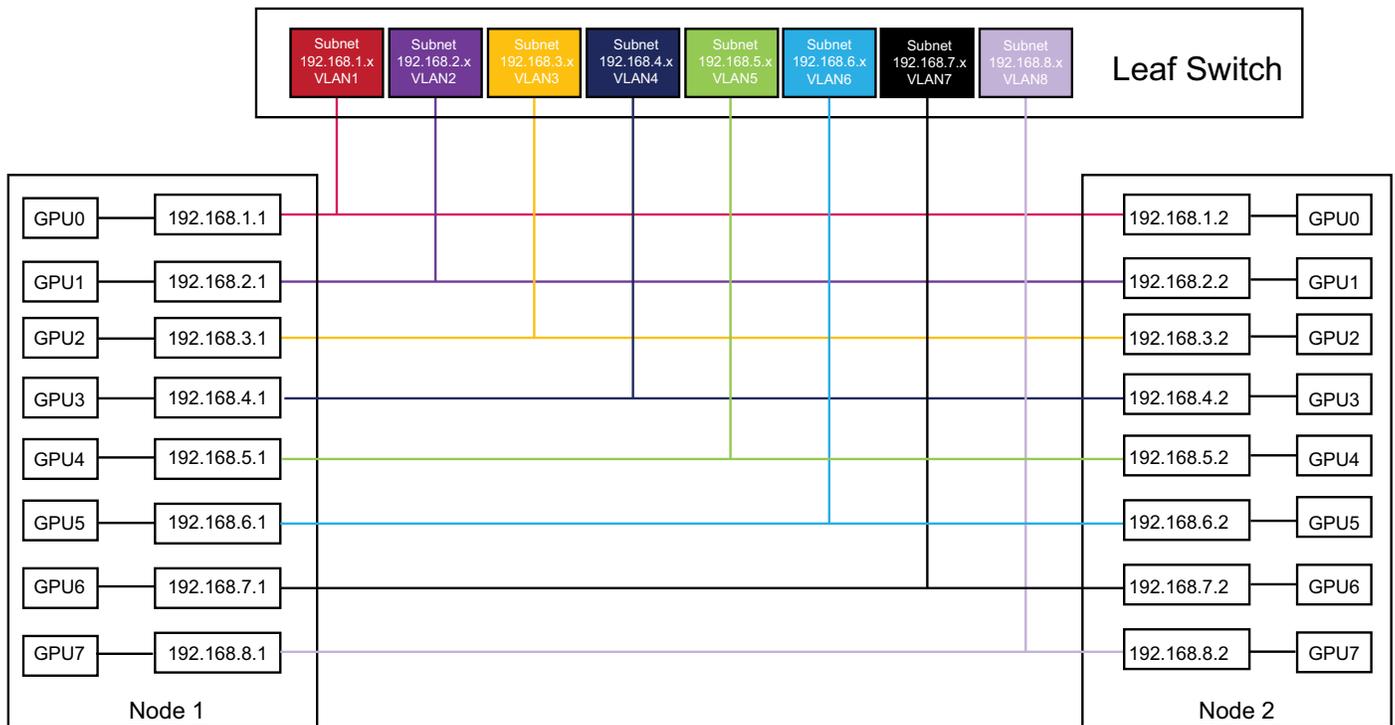
However, a better approach to the ARP flux problem is to configure each NIC on a host in a separate subnet. There are multiple approaches possible here depending on the size of the cluster being configured.

2.4.1 Single Leaf switch Topology with 24-bit subnets

The approach used in the section makes use of the 24-bit IP subnets and a Layer 3 switch. The topology referenced in this section is that of multiple hosts connecting to the same leaf switch.

1. Each host has eight NICs and eight GPUs per host.
2. The leaf switch is a Layer 3 switch and can route packets based on the IPv4 address.
3. A total of 8 IP subnets and eight VLANs are used in the network. The VLANs are only configured on the switch ports. The VLANs are not configured on the NICs.
4. Each VLAN on the switch has an IP address assigned in a unique subnet from the set of eight subnets used in the network.
5. Inter VLAN routing is used on the switch.
6. Each of the eight NICs on a host belongs to a unique subnet from the set of eight subnets used in the network. For symmetry, NIC1 on each host is connected to a switch port in the same VLAN, NIC2 on each host is connected to a switch port in the same VLAN, and so on.
7. Source-Based Routing is applied to each NIC, with the default gateway set to the IP address of the switch port VLAN to which the NIC is connected. A total of eight additional routing tables (101 through 108) are used.

Figure 3: 24-bit Subnet Scheme



The following is a sample netplan file for two of the hosts on the cluster. Other hosts can follow a similar addressing scheme.

The netplan file has 8 Ethernet interfaces. The interfaces are named eth1 through eth8 just for reference and should be replaced with the actual interface Ethernet names on the host. Each interface has an IP address with a 24-bit subnet mask.

2.4.1.1 Host 1 netplan file:/etc/netplan/00-installer-config-24bit-subnet-host1.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets: eth1:
    mtu: 9000
    addresses:
      - 192.168.1.1/24
    routes:
      - to: default
        via: 192.168.1.254
        table: 101
    routing-policy:
      - from: 192.168.1.1
        table: 101
  eth2:
    mtu: 9000
    addresses:
      - 192.168.2.1/24
    routes:
      - to: default
        via: 192.168.2.254
        table: 102
```

```
    routing-policy:
    - from: 192.168.2.1
      table: 102
eth3:
  mtu: 9000
  addresses:
  - 192.168.3.1/24
  routes:
  - to: default
    via: 192.168.3.254
    table: 103
  routing-policy:
  - from: 192.168.3.1
    table: 103
eth4:
  mtu: 9000
  addresses:
  - 192.168.4.1/24
  routes:
  - to: default
    via: 192.168.4.254
    table: 104
  routing-policy:
  - from: 192.168.4.1
    table: 104
eth5:
  mtu: 9000
  addresses:
  - 192.168.5.1/24
  routes:
  - to: default
    via: 192.168.5.254
    table: 105
  routing-policy:
  - from: 192.168.5.1
    table: 105
eth6:
  mtu: 9000
  addresses:
  - 192.168.6.1/24
  routes:
  - to: default
    via: 192.168.6.254
    table: 106
  routing-policy:
  - from: 192.168.6.1
    table: 106
eth7:
  mtu: 9000
  addresses:
  - 192.168.7.1/24
  routes:
  - to: default
    via: 192.168.7.254
    table: 107
  routing-policy:
  - from: 192.168.7.1
    table: 107
```

```
eth8:
  mtu: 9000
  addresses:
  - 192.168.8.1/24
  routes:
  - to: default
    via: 192.168.8.254
    table: 108
  routing-policy:
  - from: 192.168.8.1
    table: 108
```

After the netplan file above is applied, ensure to start and enable networked service. This disables any prior network configuration performed with network manager.

```
sudo systemctl enable systemd-networkd
sudo systemctl start systemd-networkd
```

After the netplan file above is applied, the `ip rule show` command should list eight additional rules and eight additional routing tables should be populated.

```
$ ip rule show
0:      from all lookup local
32758:  from 192.168.7.1 lookup 107 proto static
32759:  from 192.168.5.1 lookup 105 proto static
32760:  from 192.168.6.1 lookup 106 proto static
32761:  from 192.168.8.1 lookup 108 proto static
32762:  from 192.168.1.1 lookup 101 proto static
32763:  from 192.168.2.1 lookup 102 proto static
32764:  from 192.168.3.1 lookup 103 proto static
32765:  from 192.168.4.1 lookup 104 proto static
32766:  from all lookup main
32767:  from all lookup default

$ ip route list table 101
default via 192.168.1.254 dev eth1 proto static

$ ip route list table 102
default via 192.168.2.254 dev eth2 proto static

$ ip route list table 103
default via 192.168.3.254 dev eth3 proto static

$ ip route list table 104
default via 192.168.4.254 dev eth4 proto static

$ ip route list table 105
default via 192.168.5.254 dev eth5 proto static

$ ip route list table 106
default via 192.168.6.254 dev eth6 proto static

$ ip route list table 107
default via 192.168.7.254 dev eth7 proto static

$ ip route list table 108
default via 192.168.8.254 dev eth8 proto static
```

2.4.1.2 Host 2 netplan file:/etc/netplan/00-installer-config-24bit-subnet-host2.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
      mtu: 9000
      addresses:
        - 192.168.1.2/24
      routes:
        - to: default
          via: 192.168.1.254
          table: 101
      routing-policy:
        - from: 192.168.1.2
          table: 101
    eth2:
      mtu: 9000
      addresses:
        - 192.168.2.2/24
      routes:
        - to: default
          via: 192.168.2.254
          table: 102
      routing-policy:
        - from: 192.168.2.2
          table: 102
    eth3:
      mtu: 9000
      addresses:
        - 192.168.3.2/24
      routes:
        - to: default
          via: 192.168.3.254
          table: 103
      routing-policy:
        - from: 192.168.3.2
          table: 103
    eth4:
      mtu: 9000
      addresses:
        - 192.168.4.2/24
      routes:
        - to: default
          via: 192.168.4.254
          table: 104
      routing-policy:
        - from: 192.168.4.2
          table: 104
    eth5:
      mtu: 9000
      addresses:
        - 192.168.5.2/24
      routes:
        - to: default
          via: 192.168.5.254
          table: 105
```

```
    routing-policy:
    - from: 192.168.5.2
      table: 105
eth6:
  mtu: 9000
  addresses:
  - 192.168.6.2/24
  routes:
  - to: default
    via: 192.168.6.254
    table: 106
  routing-policy:
  - from: 192.168.6.2
    table: 106
eth7:
  mtu: 9000
  addresses:
  - 192.168.7.2/24
  routes:
  - to: default
    via: 192.168.7.254
    table: 107
  routing-policy:
  - from: 192.168.7.2
    table: 107
eth8:
  mtu: 9000
  addresses:
  - 192.168.8.2/24
  routes:
  - to: default
    via: 192.168.8.254
    table: 108
  routing-policy:
  - from: 192.168.8.2
    table: 108
```

After the netplan file above is applied, ensure to start and enable networked service. This disables any prior network configuration performed with network manager.

```
sudo systemctl enable systemd-networkd
sudo systemctl start systemd-networkd
```

After the netplan file above is applied, the `ip rule show` command should list eight additional rules and eight additional routing tables should be populated.

```
$ ip rule show
0:      from all lookup local
32758:  from 192.168.7.2 lookup 107 proto static
32759:  from 192.168.5.2 lookup 105 proto static
32760:  from 192.168.6.2 lookup 106 proto static
32761:  from 192.168.8.2 lookup 108 proto static
32762:  from 192.168.1.2 lookup 101 proto static
32763:  from 192.168.2.2 lookup 102 proto static
32764:  from 192.168.3.2 lookup 103 proto static
32765:  from 192.168.4.2 lookup 104 proto static
32766:  from all lookup main
```

```
32767: from all lookup default

$ ip route list table 101
default via 192.168.1.254 dev eth1 proto static

$ ip route list table 102
default via 192.168.2.254 dev eth2 proto static

$ ip route list table 103
default via 192.168.3.254 dev eth3 proto static

$ ip route list table 104
default via 192.168.4.254 dev eth4 proto static

$ ip route list table 105
default via 192.168.5.254 dev eth5 proto static

$ ip route list table 106
default via 192.168.6.254 dev eth6 proto static

$ ip route list table 107
default via 192.168.7.254 dev eth7 proto static

$ ip route list table 108
default via 192.168.8.254 dev eth8 proto static
```

The corresponding sample Ethernet switch configuration with respect to configuring the VLAN, the subnets, and the routing on the switch are shown below. The sample configuration is based on a Dell Z9664 Ethernet switch and a Supermicro SSE-T8032 Ethernet switch running SONiC. Only the configuration pertinent to the routing scheme outlined in this section is presented.

2.4.1.3 Ethernet Leaf Switch Port Configuration for 24-bit Subnet Scheme on Dell Z9664 Switch and Supermicro SSE-T8032 Switch Running SONiC OS

```
interface Vlan1
  description nic1_vlan
  ip address 192.168.1.254/24
!
interface Vlan2
  description nic2_vlan
  ip address 192.168.2.254/24
!
interface Vlan3
  description nic3_vlan
  ip address 192.168.3.254/24
!
interface Vlan4
  description nic4_vlan
  ip address 192.168.4.254/24
!
interface Vlan5
  description nic5_vlan
  ip address 192.168.5.254/24
!
interface Vlan6
  description nic6_vlan
  ip address 192.168.6.254/24
!
interface Vlan7
  description nic7_vlan
  ip address 192.168.7.254/24
!
interface Vlan8
  description nic8_vlan
  ip address 192.168.8.254/24
!
```

```
interface Eth1/1
  description "Node1 NIC1"
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  unreliable-los auto
  no shutdown
  switchport access Vlan 1
!
interface Eth1/2
  description "Node2 NIC1"
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
```

```
unreliable-los auto
no shutdown
switchport access Vlan 1
!
interface Eth1/3
description "Node1 NIC2"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 2
!
interface Eth1/4
description "Node2 NIC2"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 2
!
interface Eth1/5
description "Node1 NIC3"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 3
!
interface Eth1/6
description "Node2 NIC3"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 3
!
interface Eth1/7
description "Node1 NIC4"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 4
!
interface Eth1/8
description "Node2 NIC4"
mtu 9100
speed 400000
```

```
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 4
!
interface Eth1/9
description "Node1 NIC5"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 5
!
interface Eth1/10
description "Node2 NIC5"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 5
!
interface Eth1/11
description "Node1 NIC6"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 6
!
interface Eth1/12
description "Node2 NIC6"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 6
!
interface Eth1/13
description "Node1 NIC7"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 7
!
interface Eth1/14
description "Node2 NIC7"
```

```

mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 7
!
interface Eth1/15
description "Node1 NIC8"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 8
!
interface Eth1/16
description "Node2 NIC8"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 8
!

```

2.4.1.4 Ethernet Leaf Switch Port Configuration for 24-bit Subnet Scheme on Juniper QFX5240 Switch

```

interfaces {
  et-0/0/1 {
    description "Breakout et-0/0/1";
    number-of-sub-ports 2;
    speed 400g;
  }
  et-0/0/1:0 {
    description to.AMD-ML3100-02;
    native-vlan-id 2;
    mtu 9216;
    unit 0 {
      family ethernet-switching {
        interface-mode trunk;
        vlan {
          members vn2;
        }
      }
    }
  }
  et-0/0/1:1 {
    description to.AMD-ML3100-03;
    native-vlan-id 3;
    mtu 9216;
    unit 0 {
      family ethernet-switching {

```

```
        interface-mode trunk;
        vlan {
            members vn3;
        }
    }
}
et-0/0/2 {
    description "Breakout et-0/0/2";
    number-of-sub-ports 2;
    speed 400g;
}
et-0/0/2:0 {
    description to.AMD-ML3100-04;
    native-vlan-id 4;
    mtu 9216;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members vn4;
            }
        }
    }
}
et-0/0/2:1 {
    description to.AMD-ML3100-05;
    native-vlan-id 5;
    mtu 9216;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members vn5;
            }
        }
    }
}
et-0/0/3 {
    description "Breakout et-0/0/3";
    number-of-sub-ports 2;
    speed 400g;
}
et-0/0/3:0 {
    description to.AMD-ML3100-06;
    native-vlan-id 6;
    mtu 9216;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members vn6;
            }
        }
    }
}
et-0/0/3:1 {
    description to.AMD-ML3100-07;
```

```
native-vlan-id 7;
mtu 9216;
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members vn7;
        }
    }
}
}
et-0/0/4 {
    description "Breakout et-0/0/1";
    number-of-sub-ports 2;
    speed 400g;
}
et-0/0/4:0 {
    description to.AMD-ML3100-08;
    native-vlan-id 8;
    mtu 9216;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members vn8;
            }
        }
    }
}
}
et-0/0/4:1 {
    description to.AMD-ML3100-09;
    native-vlan-id 9;
    mtu 9216;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members vn9;
            }
        }
    }
}
}
irb {
    mtu 9216;
    unit 2 {
        family inet {
            mtu 9000;
            address 192.168.2.254/24;
        }
    }
    unit 3 {
        family inet {
            mtu 9000;
            address 192.168.3.254/24;
        }
    }
    unit 4 {
        family inet {
```

```
        mtu 9000;
        address 192.168.4.254/24;
    }
}
unit 5 {
    family inet {
        mtu 9000;
        address 192.168.5.254/24;
    }
}
unit 6 {
    family inet {
        mtu 9000;
        address 192.168.6.254/24;
    }
}
unit 7 {
    family inet {
        mtu 9000;
        address 192.168.7.254/24;
    }
}
unit 8 {
    family inet {
        mtu 9000;
        address 192.168.8.254/24;
    }
}
unit 9 {
    family inet {
        mtu 9000;
        address 192.168.9.254/24;
    }
}
}
}
vlangs {
    vn2 {
        description stripe1_leaf1_vlan2;
        vlan-id 2;
        l3-interface irb.2;
    }
    vn3 {
        description stripe1_leaf1_vlan3;
        vlan-id 3;
        l3-interface irb.3;
    }
    vn4 {
        description stripe1_leaf1_vlan4;
        vlan-id 4;
        l3-interface irb.4;
    }
    vn5 {
        description stripe1_leaf1_vlan5;
        vlan-id 5;
        l3-interface irb.5;
    }
    vn6 {
```

```
        description stripe1_leaf1_vlan6;
        vlan-id 6;
        l3-interface irb.6;
    }
    vn7 {
        description stripe1_leaf1_vlan7;
        vlan-id 7;
        l3-interface irb.7;
    }
    vn8 {
        description stripe1_leaf1_vlan8;
        vlan-id 8;
        l3-interface irb.8;
    }
    vn9 {
        description stripe1_leaf1_vlan9;
        vlan-id 9;
        l3-interface irb.9;
    }
}
```

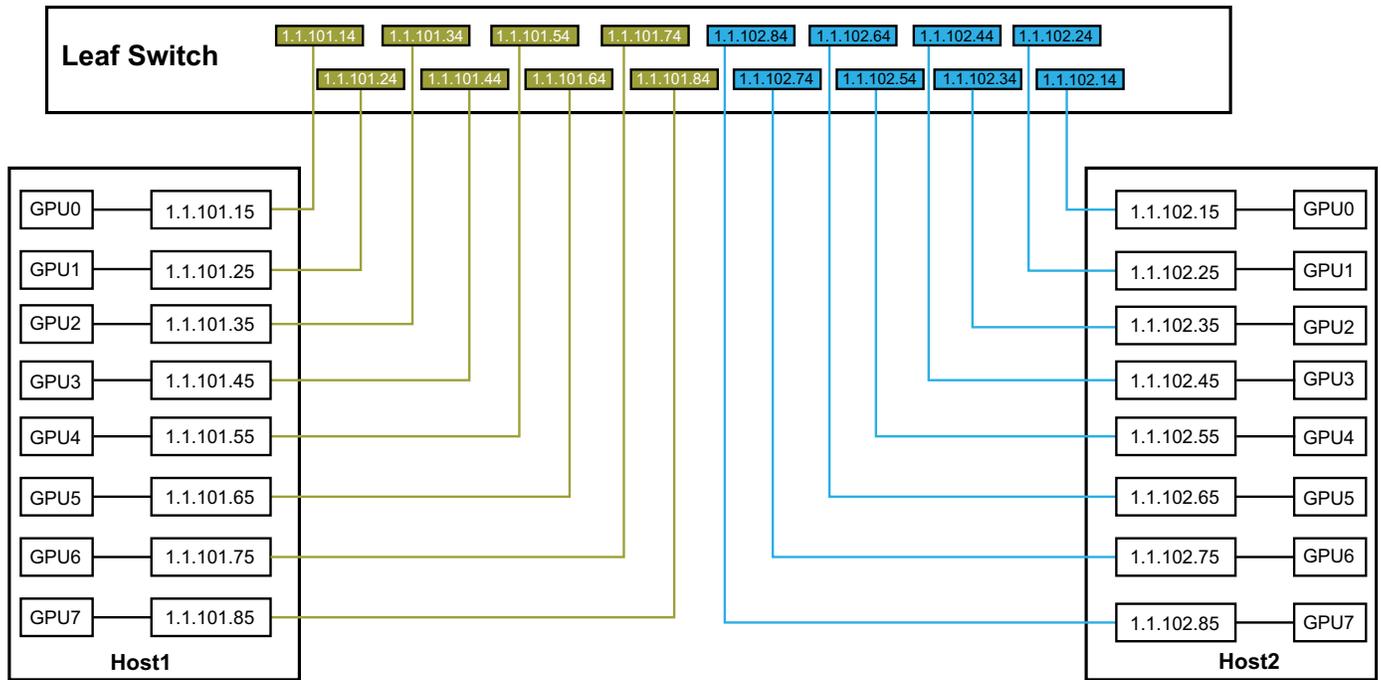
2.4.2 Single Leaf Switch Topology with 31-bit Subnets

The approach used in this section makes use of the 31-bit IP subnets and a Layer 3 switch. The 31-bit subnets enable point-to-point links as the 31-bit subnet allows only 2 addresses in the subnet. These two addresses form each end of the point-to-point link.

The topology referenced in this section is that of multiple hosts connecting to the same leaf switch.

1. Each host has eight NICs and eight GPUs per host.
2. Each NIC in the network is in a unique subnet (31-bit subnet mask). The NIC IP address forms one end of the point-to-point connection. The other IP address of the subnet is assigned to the Ethernet switch port to which the NIC connects. The switch port forms the other end of the point-to-point connection.
3. Source Based Routing is used for each NIC where the default gateway of each NIC is the IP address of the switch port to which it connects. A total of eight additional routing tables (101 through 108) are used.

Figure 4: 31-bit Subnet Scheme



The following is a sample netplan file for two of the hosts on the cluster. Other hosts can follow a similar addressing scheme.

The netplan file has eight Ethernet interfaces. The interfaces are named eth1 through eth8 just for reference and should be replaced with the actual interface Ethernet names on the host. Each interface has an IP address with a 31-bit subnet mask.

2.4.2.1 Host 1 netplan file:/etc/netplan/00-installer-config-host1.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
      mtu: 9000
      addresses:
        - 1.1.101.15/31
      routes:
        - to: default
          via: 1.1.101.14
          table: 101
          scope: global
      routing-policy:
        - from: 1.1.101.14/31
          table: 101
          priority: 0
        - to: 1.1.101.14/16
          table: 101
          priority: 1
    eth2:
      mtu: 9000
      addresses:
        - 1.1.101.25/31
      routes:
        - to: default
          via: 1.1.101.24
          table: 102
          scope: global
      routing-policy:
        - from: 1.1.101.24/31
          table: 102
          priority: 0
        - to: 1.1.101.24/16
          table: 102
          priority: 1
    eth3:
      mtu: 9000
      addresses:
        - 1.1.101.35/31
      routes:
        - to: default
          via: 1.1.101.34
          table: 103
          scope: global
      routing-policy:
        - from: 1.1.101.34/31
          table: 103
          priority: 0
        - to: 1.1.101.34/16
          table: 103
          priority: 1
    eth4:
      mtu: 9000
      addresses:
        - 1.1.101.45/31
```

```
routes:
- to: default
  via: 1.1.101.44
  table: 104
  scope: global
routing-policy:
- from: 1.1.101.44/31
  table: 104
  priority: 0
- to: 1.1.101.44/16
  table: 104
  priority: 1
eth5:
mtu: 9000
addresses:
- 1.1.101.55/31
routes:
- to: default
  via: 1.1.101.54
  table: 105
  scope: global
routing-policy:
- from: 1.1.101.54/31
  table: 105
  priority: 0
- to: 1.1.101.54/16
  table: 105
  priority: 1
eth6:
mtu: 9000
addresses:
- 1.1.101.65/31
routes:
- to: default
  via: 1.1.101.64
  table: 106
  scope: global
routing-policy:
- from: 1.1.101.64/31
  table: 106
  priority: 0
- to: 1.1.101.64/16
  table: 106
  priority: 1
eth7:
mtu: 9000
addresses:
- 1.1.101.75/31
routes:
- to: default
  via: 1.1.101.74
  table: 107
  scope: global
routing-policy:
- from: 1.1.101.74/31
  table: 107
  priority: 0
- to: 1.1.101.74/16
```

```

    table: 107
    priority: 1
eth8:
  mtu: 9000
  addresses:
  - 1.1.101.85/31
  routes:
  - to: default
    via: 1.1.101.84
    table: 108
    scope: global
  routing-policy:
  - from: 1.1.101.84/31
    table: 108
    priority: 0
  - to: 1.1.101.84/16
    table: 108
    priority: 1

```

After the netplan file above is applied, ensure to start and enable networked service. This disables any prior network configuration performed with the network manager.

```

sudo systemctl enable systemd-networkd
sudo systemctl start systemd-networkd

```

After the netplan file above is applied, the `ip rule show` command should list eight additional rules and eight additional routing tables should be populated.

```

$ ip rule show
0:      from all lookup local
0:      from 1.1.101.14/31 lookup 101 proto static
0:      from 1.1.101.24/31 lookup 102 proto static
1:      from all to 1.1.101.14/16 lookup 101 proto static
1:      from all to 1.1.101.24/16 lookup 102 proto static
0:      from 1.1.101.34/31 lookup 103 proto static
0:      from 1.1.101.44/31 lookup 104 proto static
1:      from all to 1.1.101.34/16 lookup 103 proto static
1:      from all to 1.1.101.44/16 lookup 104 proto static
0:      from 1.1.101.54/31 lookup 105 proto static
0:      from 1.1.101.64/31 lookup 106 proto static
1:      from all to 1.1.101.54/16 lookup 105 proto static
1:      from all to 1.1.101.64/16 lookup 106 proto static
0:      from 1.1.101.74/31 lookup 107 proto static
0:      from 1.1.101.84/31 lookup 108 proto static
1:      from all to 1.1.101.74/16 lookup 107 proto static
1:      from all to 1.1.101.84/16 lookup 108 proto static
32766:  from all lookup main
32767:  from all lookup default

```

```

$ ip route show table 101
default via 1.1.101.14 dev eth1 proto static

```

```

$ ip route show table 102
default via 1.1.101.24 dev eth2 proto static

```

```

$ ip route show table 103
default via 1.1.101.34 dev eth3 proto static

```

```
$ ip route show table 104
default via 1.1.101.44 dev eth4 proto static

$ ip route show table 105
default via 1.1.101.54 dev eth5 proto static

$ ip route show table 106
default via 1.1.101.64 dev eth6 proto static

$ ip route show table 107
default via 1.1.101.74 dev eth7 proto static

$ ip route show table 108
default via 1.1.101.84 dev eth8 proto static
```

2.4.2.2 Host 2 netplan file:/etc/netplan/00-installer-config-host2.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
      mtu: 9000
      addresses:
        - 1.1.102.15/31
      routes:
        - to: default
          via: 1.1.102.14
          table: 101
          scope: global
      routing-policy:
        - from: 1.1.102.14/31
          table: 101
          priority: 0
        - to: 1.1.102.14/16
          table: 101
          priority: 1
    eth2:
      mtu: 9000
      addresses:
        - 1.1.102.25/31
      routes:
        - to: default
          via: 1.1.102.24
          table: 102
          scope: global
      routing-policy:
        - from: 1.1.102.24/31
          table: 102
          priority: 0
        - to: 1.1.102.24/16
          table: 102
          priority: 1
    eth3:
      mtu: 9000
      addresses:
```

```
- 1.1.102.35/31
routes:
- to: default
  via: 1.1.102.34
  table: 103
  scope: global
routing-policy:
- from: 1.1.102.34/31
  table: 103
  priority: 0
- to: 1.1.102.34/16
  table: 103
  priority: 1
eth4:
mtu: 9000
addresses:
- 1.1.102.45/31
routes:
- to: default
  via: 1.1.102.44
  table: 104
  scope: global
routing-policy:
- from: 1.1.102.44/31
  table: 104
  priority: 0
- to: 1.1.102.44/16
  table: 104
  priority: 1
eth5:
mtu: 9000
addresses:
- 1.1.102.55/31
routes:
- to: default
  via: 1.1.102.54
  table: 105
  scope: global
routing-policy:
- from: 1.1.102.54/31
  table: 105
  priority: 0
- to: 1.1.102.54/16
  table: 105
  priority: 1
eth6:
mtu: 9000
addresses:
- 1.1.102.65/31
routes:
- to: default
  via: 1.1.102.64
  table: 106
  scope: global
routing-policy:
- from: 1.1.102.64/31
  table: 106
  priority: 0
```

```
- to: 1.1.102.64/16
  table: 106
  priority: 1
eth7:
  mtu: 9000
  addresses:
  - 1.1.102.75/31
  routes:
  - to: default
    via: 1.1.102.74
    table: 107
    scope: global
  routing-policy:
  - from: 1.1.102.74/31
    table: 107
    priority: 0
  - to: 1.1.102.74/16
    table: 107
    priority: 1
eth8:
  mtu: 9000
  addresses:
  - 1.1.102.85/31
  routes:
  - to: default
    via: 1.1.102.84
    table: 108
    scope: global
  routing-policy:
  - from: 1.1.102.84/31
    table: 108
    priority: 0
  - to: 1.1.102.84/16
    table: 108
    priority: 1
```

After the netplan file above is applied, ensure to start and enable networked service. This disables any prior network configuration performed with the network manager.

```
sudo systemctl enable systemd-networkd
sudo systemctl start systemd-networkd
```

After the netplan file above is applied, the `ip rule show` command should list 16 additional rules and 8 additional routing tables should be populated.

```
$ ip rule show
0:    from all lookup local
0:    from 1.1.102.14/31 lookup 101 proto static
0:    from 1.1.102.24/31 lookup 102 proto static
1:    from all to 1.1.102.14/16 lookup 101 proto static
1:    from all to 1.1.102.24/16 lookup 102 proto static
0:    from 1.1.102.34/31 lookup 103 proto static
0:    from 1.1.102.44/31 lookup 104 proto static
1:    from all to 1.1.102.34/16 lookup 103 proto static
1:    from all to 1.1.102.44/16 lookup 104 proto static
0:    from 1.1.102.54/31 lookup 105 proto static
0:    from 1.1.102.64/31 lookup 106 proto static
```

```
1:      from all to 1.1.102.54/16 lookup 105 proto static
1:      from all to 1.1.102.64/16 lookup 106 proto static
0:      from 1.1.102.74/31 lookup 107 proto static
0:      from 1.1.102.84/31 lookup 108 proto static
1:      from all to 1.1.102.74/16 lookup 107 proto static
1:      from all to 1.1.102.84/16 lookup 108 proto static
32766:  from all lookup main
32767:  from all lookup default
```

```
$ ip route show table 101
default via 1.1.102.14 dev eth1 proto static
```

```
$ ip route show table 102
default via 1.1.102.24 dev eth2 proto static
```

```
$ ip route show table 103
default via 1.1.102.34 dev eth3 proto static
```

```
$ ip route show table 104
default via 1.1.102.44 dev eth4 proto static
```

```
$ ip route show table 105
default via 1.1.102.54 dev eth5 proto static
```

```
$ ip route show table 106
default via 1.1.102.64 dev eth6 proto static
```

```
$ ip route show table 107
default via 1.1.102.74 dev eth7 proto static
```

```
$ ip route show table 108
default via 1.1.102.84 dev eth8 proto static
```

The corresponding sample Ethernet switch configuration with respect to configuring the IP addresses and the routing on the switch is shown below. The sample configuration is based on a Dell Z9664 Ethernet switch and a Supermicro SSE-T8032 Ethernet switch running SONiC. Only the configuration pertinent to the routing scheme outlined in this section is presented.

NOTE: The example shown below uses a port scheme where the different NICs of the same host are connected to the consecutive front panel Ethernet switch ports. However, it's also possible to connect NIC1 of each host to the consecutive front panel switch ports and so on.

2.4.2.3 Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Dell Z9664 Switch Running SONiC OS

```
interface Eth1/1
  description node1_eth1
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  no shutdown
  ip address 1.1.101.14/31
!
interface Eth1/2
  description node1_eth2
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  no shutdown
  ip address 1.1.101.24/31
!
interface Eth1/3
  description node1_eth3
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  no shutdown
  ip address 1.1.101.34/31
!
interface Eth1/4
  description node1_eth4
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  no shutdown
  ip address 1.1.101.44/31
!
interface Eth1/5
  description node1_eth5
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  no shutdown
  ip address 1.1.101.54/31
!
interface Eth1/6
  description node1_eth6
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training

  no shutdown
  ip address 1.1.101.64/31
```

```
!  
interface Eth1/7  
  description node1_eth7  
  mtu 9100  
  speed 400000  
  fec RS  
  standalone-link-training  
  no shutdown  
  ip address 1.1.101.74/31  
!  
interface Eth1/8  
  description node1_eth8  
  mtu 9100  
  speed 400000  
  fec RS  
  standalone-link-training  
  no shutdown  
  ip address 1.1.101.84/31  
!  
interface Eth1/9  
  description node2_eth1  
  mtu 9100  
  speed 400000  
  fec RS  
  standalone-link-training  
  no shutdown  
  ip address 1.1.102.14/31  
!  
interface Eth1/10  
  description node2_eth2  
  mtu 9100  
  speed 400000  
  fec RS  
  standalone-link-training  
  no shutdown  
  ip address 1.1.102.24/31  
!  
interface Eth1/11  
  description node2_eth3  
  mtu 9100  
  speed 400000  
  fec RS  
  standalone-link-training  
  no shutdown  
  ip address 1.1.102.34/31  
!  
interface Eth1/12  
  description node2_eth4  
  mtu 9100  
  speed 400000  
  fec RS  
  standalone-link-training  
  no shutdown  
  ip address 1.1.102.44/31  
!  
interface Eth1/13  
  description node2_eth5  
  mtu 9100
```

```
speed 400000
fec RS
standalone-link-training
no shutdown
ip address 1.1.102.54/31
!
interface Eth1/14
description node2_eth6
mtu 9100
speed 400000
fec RS
standalone-link-training
no shutdown
ip address 1.1.102.64/31
!
interface Eth1/15
description node2_eth7
mtu 9100
speed 400000
fec RS
standalone-link-training
no shutdown
ip address 1.1.102.74/31
!
interface Eth1/16
description node2_eth8
mtu 9100
speed 400000
fec RS
standalone-link-training
no shutdown
ip address 1.1.102.84/31
!
```

2.4.2.4 Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Juniper QFX5240 Switch

```
interfaces {
  et-0/0/1 {
    description "Breakout et-0/0/1";
    number-of-sub-ports 2;
    speed 400g;
  }
  et-0/0/1:0 {
    description to.AMD-ML3100-01;
    mtu 9216;
    unit 0 {
      family inet {
        mtu 9170;
        address 192.168.2.1/31;
      }
    }
  }
  et-0/0/1:1 {
    description to.AMD-ML3100-02;
    mtu 9216;
  }
}
```

```
    unit 0 {
        family inet {
            mtu 9170;
            address 192.168.2.3/31;
        }
    }
}
et-0/0/2 {
    description "Breakout et-0/0/2";
    number-of-sub-ports 2;
    speed 400g;
}
et-0/0/2:0 {
    description to.AMD-ML3100-03;
    mtu 9216;
    unit 0 {
        family inet {
            mtu 9170;
            address 192.168.2.5/31;
        }
    }
}
et-0/0/2:1 {
    description to.AMD-ML3100-ML3100-04;
    mtu 9216;
    unit 0 {
        family inet {
            mtu 9170;
            address 192.168.2.7/31;
        }
    }
}
et-0/0/3 {
    description "Breakout et-0/0/3";
    number-of-sub-ports 2;
    speed 400g;
}
et-0/0/3:0 {
    description to.AMD-ML3100-ML3100-05;
    mtu 9216;
    unit 0 {
        family inet {
            mtu 9170;
            address 192.168.2.9/31;
        }
    }
}
et-0/0/3:1 {
    description to.AMD-ML3100-ML3100-06;
    mtu 9216;
    unit 0 {
        family inet {
            mtu 9170;
            address 192.168.2.11/31;
        }
    }
}
et-0/0/4 {
```

```

        description "Breakout et-0/0/4";
        number-of-sub-ports 2;
        speed 400g;
    }
    et-0/0/3:0 {
        description to.AMD-ML3100-ML3100-07;
        mtu 9216;
        unit 0 {
            family inet {
                mtu 9170;
                address 192.168.2.13/31;
            }
        }
    }
    et-0/0/3:1 {
        description to.AMD-ML3100-ML3100-08;
        mtu 9216;
        unit 0 {
            family inet {
                mtu 9170;
                address 192.168.2.15/31;
            }
        }
    }
}

```

2.4.3 Confirm Routing Between Different NICs Across Different Hosts

With the routing configured in the backend network as shown in [Configuring Host Routing for the Backend Network](#), ping and trace route tests can be used to confirm that any NIC on any host can reach any other NIC on any other host.

```

# Example: ping Host1,NIC1 to Host2,NIC7
ping -I 192.168.1.1 192.168.7.2
traceroute -i eth1 192.168.7.2

```

```

# Example: ping Host1,NIC6 to Host2,NIC2
ping -I 192.168.6.1 192.168.2.2
traceroute -i eth6 192.168.2.2

```

All highlighted items above will depend on netplan setup

The following shell script can be used to test the full mesh ping across all the NICs of two given hosts. The script runs ping tests from each NIC of host1 to each NIC of host2.

```

#!/bin/bash

host1_ip_list=(192.168.1.15 192.168.2.15 192.168.3.15 192.168.4.15 192.168.5.15 192.168.6.15
192.168.7.15 192.168.8.15)

host2_ip_list=(192.168.1.14 192.168.2.14 192.168.3.14 192.168.4.14 192.168.5.14 192.168.6.14
192.168.7.14 192.168.8.14)

for i in ${host1_ip_list[@]}
do
    for j in ${host2_ip_list[@]}
    do
        echo -e "\nping -I $i $j\n===== \n"
    done
done

```

```
ping -I $i -c 1 $j > /dev/null
if [ $? == 0 ]; then

    echo "Ping Passed"
else
    echo -e "Ping Failed\n\n\n\n"
    exit 5
fi
done
done
```

```
# All highlighted items above will depend on netplan setup
```

2.5 Ethernet Switch Configuration for QoS and Congestion Control

Every GPU cluster and HPC Cluster has a single or multi-tier Ethernet switch architecture to connect all the NICs on all the hosts in the cluster. With multiple NICs and multiple hosts in the cluster communicating with each other all the time, and flows starting and stopping asynchronously, congestion in the network becomes inevitable. NCCL collectives also involve many-to-many and many-to-one communication patterns leading to congestion on various network paths.

RoCE provides a very high bandwidth and low latency transport, but it is sensitive to packet drops. Any packet drop incurred due to congestion in the switches or the NICs impacts RoCE performance. Therefore, to handle congestion, the NICs and the switches in the cluster need to be configured for Congestion Control.

Broadcom NICs implement the DCQCN-P Congestion Control Algorithm. DCQCN stands for Data Center Quantized Congestion Notification and DCQCN-P stands for DCQCN- Probabilistic. With the NICs using the DCQCN-P Congestion Control Algorithm, the switches in the network need to be configured for DCQCN-P as well.

In simple terms, the DCQCN algorithm relies on the switches marking the ECN field in the IP header of the RoCE v2 packets when the switch buffers rise beyond a configured threshold. The NIC receiving an ECN-marked packet sends a Congestion Notification Packet (CNP) to the NIC that sent the RoCE v2 packet that was ECN marked by the switch. The NIC, upon receiving a CNP regulates its sending rate to alleviate the congestion in the network. An important thing to note is that the Congestion Control Algorithm operates at the RoCE Connection or the RoCE Queue Pair(QP) level, with the RoCE v2 and the CNP packets carrying the QP number in them.

The DCQCN-P algorithm uses the probabilistic ECN marking. In probabilistic marking, minimum and maximum marking thresholds are specified at the switch together with maximum marking probability. When the switch egress port queue crosses the minimum threshold, marking starts at low probability which linearly increases between 0 and max probability in the range between min. and max. marking thresholds. Once the maximum threshold is reached, every packet is marked (for example, 100% marking).

Congestion Control Algorithms help with Congestion in the network but do not make the network lossless. Even with Congestion Control configured in the network, packets can be dropped with flows starting asynchronously or due to an incast. To make the network lossless, Priority Flow Control (PFC) needs to be configured in the network.

When the NIC is configured for RoCE on a host along with the RoCE QOS configuration pkg (`bnxt_re_conf`), the NIC is automatically configured for DCQCN-P along with the following settings.

- RoCE v2 packets are marked with a DSCP value of 26 and use Priority 3 internally
- CNP packets are marked with a DSCP value of 48 and use Priority 7 internally
- PFC is enabled for Priority 3 traffic
- Three Traffic classes are set up, TC0 for non-RoCE traffic, TC1 for RoCE traffic, and TC2 for CNP traffic

- RoCE and non-RoCE traffic share ETS bandwidth of 50% each. The ETS bandwidth share applies only when the actual traffic is available to use the bandwidth share. In the absence of non-RoCE traffic, all the available bandwidth will be used by RoCE and vice-versa.
- CNP traffic is treated as ETS Strict Priority

```
$ niccli -i 1 qos --ets --show
```

```
IEEE 8021QAZ ETS Configuration TLV:
    PRIO_MAP: 0:0 1:0 2:0 3:1 4:0 5:0 6:0 7:2
    TC Bandwidth: 50% 50% 0% 0% 0% 0% 0% 0%
    TSA_MAP: 0:ets 1:ets 2:strict 3:strict 4:strict 5:strict 6:strict 7:strict
IEEE 8021QAZ PFC TLV:
    PFC enabled: 3
IEEE 8021QAZ APP TLV:
    APP#0:
        Priority: 7
        Sel: 5
        DSCP: 48

    APP#1:
        Priority: 3
        Sel: 5
        DSCP: 26

    APP#2:
        Priority: 3
        Sel: 3
        UDP or DCCP: 4791
```

```
TC Rate Limit: 100% 100% 100% 0% 0% 0% 0% 0%
```

```
$ sudo niccli -i 1 qos --listmap --pri2cos
```

```
Base Queue is 0 for port 0.
```

```
-----
Priority   TC HW Queue ID
-----
0         0   4
1         0   4
2         0   4
3         1   0
4         0   4
5         0   4
6         0   4
7         2   5
```

```
$ sudo niccli -i 1 qos --dscp2prio
```

```
dscp2prio mapping:
    priority:7 dscp:48,
    priority:3 dscp:26,
```

Broadcom provides tools NICCLI and bnxsetupcc.sh that allow changing the DSCP values, the Priority values, the ETS settings, and the PFC settings.

NOTE: The important thing to note is that the settings on the NIC and the Ethernet switches match each other.

Most network switches used for RoCE are based on Broadcom's Ethernet switch silicon. The switch silicon provides the ability to configure PFC and Congestion Control settings. Prominent switch vendors such as Dell, Arista, Supermicro, Juniper Networks and so forth, implement their own Software Stacks for Ethernet switches. Each vendor provides their own command line interface (CLI) to configure the switches. The following sections describe how to configure popular Dell, Arista, Juniper Networks, and Supermicro switch models for PFC and DCQCN-P.

The switch configuration elements required are as follows:

- Map DSCP traffic priorities for RoCE and CNP traffic to traffic classes
- Enable PFC
- Enable ECN
- Configure ECN marking algorithm and ECN marking threshold

2.5.1 Example: Arista 7060CX (DCQCN-P at 400G) and 31-bit Subnet Scheme

```
qos map traffic-class 3 to dscp 26
qos map traffic-class 7 to dscp 48
```

```
qos profile QOS_ROCE_DCQCN
  qos trust dscp
  priority-flow-control on
  priority-flow-control priority 3 no-drop
  !
  uc-tx-queue 0
    no priority
  !
  uc-tx-queue 1
    no priority
  !
  uc-tx-queue 3
    no priority
    random-detect ecn minimum-threshold 1000 kbytes maximum-threshold 3000 kbytes max-mark-
probability 20 weight 0
  !
```

```
interface Ethernet1/1
  mtu 9200
  flowcontrol send off
  flowcontrol receive off
  speed 400g-8
  error-correction encoding reed-solomon
  phy link training
  service-profile QOS_ROCE_DCQCN
  ip address 1.1.101.14/31
  !
```

```
interface Ethernet2/1
  mtu 9200
```

```

flowcontrol send off
flowcontrol receive off
speed 400g-8
error-correction encoding reed-solomon
phy link training
service-profile QOS_ROCE_DCQCN
ip address 1.1.101.24/31
!

```

2.5.2 Example: Dell Z9664 Switch and Supermicro SSE-T8032 Switch Running SONiC OS and 31-bit Subnet Scheme

```

qos wred-policy ROCE
green minimum-threshold 1048 maximum-threshold 2097 drop-probability 5
ecn green
!
qos scheduler-policy ROCE
!
queue 0
type dwrr
weight 50
!
queue 3
type dwrr
weight 50
!
queue 4
type dwrr
weight 50
!
queue 6
type strict
!

qos map dscp-tc ROCE
dscp 0-3,5-23,25,27-47,49-63 traffic-class 0
dscp 24,26 traffic-class 3
dscp 4 traffic-class 4
dscp 48 traffic-class 6
!
qos map dot1p-tc ROCE
dot1p 0-2,5-7 traffic-class 0
dot1p 3 traffic-class 3
dot1p 4 traffic-class 4
!
qos map tc-queue ROCE
traffic-class 0 queue 0
traffic-class 1 queue 1
traffic-class 2 queue 2
traffic-class 3 queue 3
traffic-class 4 queue 4
traffic-class 5 queue 5
traffic-class 6 queue 6
traffic-class 7 queue 7
!

```

```
qos map tc-pg ROCE
 traffic-class 3 priority-group 3
 traffic-class 4 priority-group 4
 traffic-class 0-2,5-7 priority-group 7
!
qos map pfc-priority-queue ROCE
 pfc-priority 0 queue 0
 pfc-priority 1 queue 1
 pfc-priority 2 queue 2
 pfc-priority 3 queue 3
 pfc-priority 4 queue 4
 pfc-priority 5 queue 5
 pfc-priority 6 queue 6
 pfc-priority 7 queue 7
!

interface Eth1/1
 description nodel_eth2
 mtu 9100
 speed 400000
 fec RS
 standalone-link-training
 unreliable-los auto
 no shutdown
 ip address 1.1.101.14/31
 queue 3 wred-policy ROCE
 scheduler-policy ROCE
 qos-map dscp-tc ROCE
 qos-map dot1p-tc ROCE
 qos-map tc-queue ROCE
 qos-map tc-pg ROCE
 qos-map pfc-priority-queue ROCE
 priority-flow-control priority 3
 priority-flow-control priority 4
 priority-flow-control watchdog action drop
 priority-flow-control watchdog on detect-time 200
 priority-flow-control watchdog restore-time 400
!
```

```

interface Eth1/2
  description node1_eth2
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  unreliable-los auto
  no shutdown
  ip address 1.1.101.24/31
  queue 3 wred-policy ROCE
  scheduler-policy ROCE
  qos-map dscp-tc ROCE
  qos-map dot1p-tc ROCE
  qos-map tc-queue ROCE
  qos-map tc-pg ROCE
  qos-map pfc-priority-queue ROCE
  priority-flow-control priority 3
  priority-flow-control priority 4
  priority-flow-control watchdog action drop
  priority-flow-control watchdog on detect-time 200
  priority-flow-control watchdog restore-time 400
!
```

2.5.3 Example: Juniper QFX5240 Switch and 31-bit Subnet Scheme

```

forwarding-options {
  hash-key {
    family inet {
      layer-3;
      layer-4;
    }
  }
  enhanced-hash-key {
    ecmp-dlb {
      flowlet {
        inactivity-interval 256;
        flowset-table-size 2048;
        reassignment {
          prob-threshold 3;
          quality-delta 6;
        }
      }
      ether-type {
        ipv4;
      }
      sampling-rate 1000000;
    }
  }
}
class-of-service {
  classifiers {
    dscp mydscp {
      forwarding-class CNP {
        loss-priority low code-points 110000;
      }
      forwarding-class NO-LOSS {

```

```
        loss-priority low code-points 011010;
    }
}
drop-profiles {
    dp1 {
        interpolate {
            fill-level [ 55 90 ];
            drop-probability [ 0 100 ];
        }
    }
}
shared-buffer {
    ingress {
        buffer-partition lossless {
            percent 80;
        }
        buffer-partition lossless-headroom {
            percent 10;
        }
        buffer-partition lossy {
            percent 10;
        }
    }
    egress {
        buffer-partition lossless {
            percent 80;
        }
        buffer-partition lossy {
            percent 10;
        }
    }
}
forwarding-classes {
    class CNP queue-num 3;
    class NO-LOSS queue-num 4 no-loss pfc-priority 3;
}
congestion-notification-profile {
    cnp {
        input {
            dscp {
                code-point 011010 {
                    pfc;
                }
            }
        }
        output {
            ieee-802.1 {
                code-point 011 {
                    flow-control-queue 4;
                }
            }
        }
    }
}
interfaces {
    et-* {
        congestion-notification-profile cnp;
    }
}
```

```

        scheduler-map sml;
        unit * {
            classifiers {
                dscp mydscp;
            }
        }
    }
}
scheduler-maps {
    sml {
        forwarding-class CNP scheduler s2-cnp;
        forwarding-class NO-LOSS scheduler s1;
    }
}
schedulers {
    s1 {
        drop-profile-map loss-priority any protocol any drop-profile dpl;
        explicit-congestion-notification;
    }
    s2-cnp {
        transmit-rate percent 5;
        priority strict-high;
    }
}
}

```

2.6 Final Checks and Settings for Optimal Performance

The following final checks can be made to ensure that the software, the firmware, the tools, and other settings are configured correctly for optimal Peer Memory Direct performance.

1. Ensure the `bnxt_en.ko`, `bnxt_re.ko`, and `ib_peer_mem.ko` kernel modules are loaded and are the correct version.

NOTE: Certain Ubuntu kernels have built-in `ib_peer_mem` support; these kernels do not require `ib_peer_mem` to be built and loaded from the Broadcom-provided release. The kernel driver Makefile can detect if `ib_peer_mem` is required to be built or not and act accordingly.

2. Ensure that the file `/etc/bnxt_re/bnxt_re.conf` has the correct RoCE QOS values and each NIC has the correct QOS settings. The QOS settings on each NIC can be confirmed via the following `niccli qos --ets --show` command:

```
niccli --dev <index | pci b:d:f | mac> qos --ets --show
```

3. Make sure `nvidia-peermem` module is loaded when using NVIDIA GPUs.

```
lsmod | grep peer
nvidia_peermem      16384  0
nvidia              9744384  3 nvidia_uvm,nvidia_peermem,nvidia_modeset
ib_uverbs           196608  3 nvidia_peermem,bnxt_re,rdma_ucm
```

4. PCIe Access Control Service (ACS) is disabled on the PCIe switch connecting the NIC and the GPU to allow PCIe Peer to Peer Transactions between the GPU and the NIC. If ACS is enabled, performance will degrade.
5. IOMMU is disabled or is in Pass Through (PT) mode.
6. Standard InfiniBand Commands listed below work correctly. These commands are part of the `infiniband-diags` package which can be downloaded by the OS distro's package manager:

- `ibstatus`
 - `ibv_devinfo -vvv`
 - `ibdev2netdev`
7. NIC NVM Configuration (to enable RDMA, performance profile, and PCIe Relaxed Ordering) is set to enabled.
8. The NIC interface shows the NIC link is up and linked at the correct speed. This can be verified using one of the following commands.
- `ibstatus`
 - `ethtool <ifname>`
9. An IP address is assigned to the NIC interface and the IP address is visible as GID 3 for IPv4 address or IPv6 address in `ibv_devinfo -vvv` command below:
- `rdma link show`
 - `ibv_devinfo -vvv`
 - `ibv_devinfo -vvv -d <roce_interface_name>`
(example, `ibv_devinfo -vvv -d bnxt_re0`)
10. Interface MTU size is set to 9000 bytes on the host for maximum throughput.
11. The Ethernet switch port to which the NIC connects has its MTU set to 9000.
12. The PCIe slot for the NIC shows correct PCIe GEN speed and width.
- `lspci -vvv -s <B:D:F>`
13. Firewall is disabled on the communicating hosts, in case it prevents RDMA connections from being set up.
14. There are no NIC and GPU-related errors in the Linux dmesg logs.
15. For NCCL testing, NUMA balancing is disabled on each host participating in the NCCL tests.
- ```
echo 0 > /proc/sys/kernel/numa_balancing or
sysctl -w kernel.numa_balancing=0
```
- Another option is to add the following entry to file `/etc/sysctl.d/99-sysctl.conf` so that the setting takes effect automatically after a reboot.
- ```
kernel.numa_balancing=0
```

2.7 Installing and Compiling Perfctest with NVIDIA GPU Support

NOTE: The following instructions are intended for recompiled perfctest utility that can be run directly from the home directory. The instructions assume CUDA toolkit is already installed.

```
sudo apt install libibumad-i
sudo apt install pciutils
sudo apt install libpci*
sudo apt install automake autoconf libtool libibverbs-i ibverbs-utils infiniband-diags ethtool
librdmacm-i
git clone https://github.com/linux-rdma/perfctest.git
cd perfctest/
./autogen.sh
./configure CUDA_H_PATH=/usr/local/cuda/include/cuda.h
make

./ib_write_bw -h | grep -i cuda
--use_cuda=<cuda device id> Use CUDA specific device for GPUDirect RDMA testing
--use_cuda_bus_id=<cuda full BUS id> Use CUDA specific device, based on its full PCIe address
```

2.8 Validating Peer Memory Direct Support with Perfctest

This section provides information on validating Peer Memory Direct Support with Perfctest:

2.8.1 Using an NVIDIA GPU

```
$HOME/perfctest/ib_write_bw -d <roce-interface-name> --use_cuda=<gpu-id> -a -F -x 3 --report_gbits -q  
2 -b
```

With the routing configured in the backend network as shown in [Configuring Routing for the BackEnd Network](#), perfctest can be used for RDMA tests across different NICs on different hosts.

2.8.2 Example – `ib_write_bw` test using Broadcom NIC with NVIDIA GPU

The following example of running `ib_write_bw` test using a Broadcom NIC with an NVIDIA GPU across Host1, NIC0, and Host2, NIC0. For the best `peer_memory` direct performance, the GPUs used for this test should be the one closest to the NIC on the PCIe Bus.

```
# Start Server on Host1

$ $HOME/perftest/bin/ib_write_bw -d bnxt_re0 --use_cuda=0 -a -F -x 3 -q 4 --report_gbits

# Start Client on Host2

$ $HOME/perftest/bin/ib_write_bw -d bnxt_re0 --use_cuda=0 -a -F -x 3 -q 4 --report_gbits 1.1.102.15
```

```
Perftest doesn't supports CUDA tests with inline messages: inline size set to 0
initializing CUDA
```

```
Listing all CUDA devices in system:
CUDA device 0: PCIe address is 18:00
CUDA device 1: PCIe address is 2B:00
CUDA device 2: PCIe address is 3A:00
CUDA device 3: PCIe address is 5E:00
CUDA device 4: PCIe address is 84:00
CUDA device 5: PCIe address is 9D:00
CUDA device 6: PCIe address is B0:00
CUDA device 7: PCIe address is D7:00
```

```
Picking device No. 0
[pid = 7710, dev = 0] device name = [NVIDIA H100 80GB HBM3]
creating CUDA Ctx
making it the current CUDA Ctx
CUDA device integrated: 0
cuMemAlloc() of a 67108864 bytes GPU buffer
allocated GPU buffer address at 000074ddf0000000 pointer=0x74ddf0000000
```

```
-----
RDMA_Write BW Test
Dual-port      : OFF          Device       : bnxt_re0
Number of qps  : 4           Transport type : IB
Connection type : RC         Using SRQ     : OFF
PCIe relax order: ON
ibv_wr* API    : OFF
TX depth       : 128
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : Ethernet
GID index      : 3
Max inline data : 0[B]
rdma_cm QPs    : OFF
Data ex. method : Ethernet
-----
```

```
local address: LID 0000 QPN 0x2c01 PSN 0xee4161 RKey 0x2000001 VAddr 0x0074ddf2000000
GID: 00:00:00:00:00:00:00:00:00:00:00:00:00:255:255:01:01:101:15
local address: LID 0000 QPN 0x2c02 PSN 0x626637 RKey 0x2000001 VAddr 0x0074ddf2800000
GID: 00:00:00:00:00:00:00:00:00:00:00:00:00:255:255:01:01:101:15
local address: LID 0000 QPN 0x2c03 PSN 0xe070b5 RKey 0x2000001 VAddr 0x0074ddf3000000
GID: 00:00:00:00:00:00:00:00:00:00:00:00:00:255:255:01:01:101:15
local address: LID 0000 QPN 0x2c04 PSN 0xd53a60 RKey 0x2000001 VAddr 0x0074ddf3800000
GID: 00:00:00:00:00:00:00:00:00:00:00:00:00:255:255:01:01:101:15
```

```

remote address: LID 0000 QPN 0x2c01 PSN 0xb642de RKey 0x2000001 VAddr 0x00739cf6000000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:01:01:102:15
remote address: LID 0000 QPN 0x2c02 PSN 0x93c9a7 RKey 0x2000001 VAddr 0x00739cf6800000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:01:01:102:15
remote address: LID 0000 QPN 0x2c03 PSN 0x7f42f RKey 0x2000001 VAddr 0x00739cf7000000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:01:01:102:15
remote address: LID 0000 QPN 0x2c04 PSN 0xc95e1b RKey 0x2000001 VAddr 0x00739cf7800000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:01:01:102:15
    
```

#bytes	#iterations	BW peak[Gb/sec]	BW average[Gb/sec]	MsgRate[Mpps]
2	20000	0.071589	0.071102	4.443871
4	20000	0.13	0.13	3.955896
8	20000	0.28	0.28	4.302460
16	20000	0.52	0.52	4.040328
32	20000	0.95	0.95	3.704634
64	20000	1.73	1.73	3.370019
128	20000	3.81	3.71	3.621056
256	20000	6.91	6.91	3.371592
512	20000	14.42	13.92	3.399008
1024	20000	27.63	27.57	3.365319
2048	20000	55.26	55.16	3.366869
4096	20000	111.27	111.15	3.392172
8192	20000	222.53	221.91	3.386064
16384	20000	387.79	387.59	2.957057
32768	20000	390.10	389.81	1.487013
65536	20000	391.12	390.98	0.745735
131072	20000	391.63	391.61	0.373472
262144	20000	391.89	391.85	0.186850
524288	20000	392.01	392.00	0.093461
1048576	20000	392.08	392.07	0.046739
2097152	20000	392.11	392.11	0.023371
4194304	20000	392.12	392.12	0.011686
8388608	20000	392.13	392.13	0.005843

```

deallocating GPU buffer 000074ddf0000000
destroying current CUDA Ctx
    
```

NOTE: The GPU buffers are allocated before the start of the test and deallocated after test completion. This confirms that perfest is running with Peer Mem.

Chapter 3: System BIOS

3.1 BIOS Setting Recommendations

The following BIOS settings (see [Table 2](#)) are recommended by Dell for their XE9680 AI//ML server. The BIOS settings disable IOMMU and ACS on the host as well.

Table 2: BIOS Settings Recommendations

UEFI/BIOS Area	Value
BIOS -> Processor Settings	Logical Processor = Disable
	Virtualization Technology = Disable
	SubNumaCluster = Disable
	MADt Core cluster = Linear
BIOS -> Integrated Devices	Global SRIOV = Disable
BIOS -> System Profile Setting	Server System Profile = Performance
	Workload = Not Configured
BIOS -> System Security	AC Recovery Delay = Random (highly recommended)

Chapter 4: Atlas2 PCIe Switch Configuration

There are four switches in the system. Each switch is partitioned into two virtual switches. Each VS has 4 to 5 downstream ports. Station 0 is configured as 4 x 4 and other stations are all x16.

See [Appendix C, PCIe Link Speed and Width Related Scripts](#) for a bash shell script that can be used to disable ACS on a host.

Chapter 5: Debugging Thor2 NIC

5.1 Frequently Asked Questions and Troubleshooting

1. RoCE interface names on a host do not have names like `bnx_re0`, `bnxt_re1`, and so forth.

This is due to the setting `NAME_FALLBACK` in file `/usr/lib/udev/rules.d/60-rdma-persistent-naming.rules` as follows:

```
ACTION=="add", SUBSYSTEM=="infiniband", PROGRAM="rdma_rename %k NAME_FALLBACK"
```

With `NAME_FALLBACK`, the RoCE interfaces are named based on the PCIe ID of the PCIe slot used for the NIC.

Replacing `NAME_FALLBACK` with `NAME_KERNEL` will rename the RoCE interfaces to the `bnxt_reX` format.

2. RoCE perfests (`ib_write_bw`, `ib_read_bw`, `ib_send_bw`) fail with status 12 as shown in the following example:

```
ib_write_bw -d roceo3811 -F -x 3 192.168.1.11
```

```
-----
RDMA_Write BW Test
Dual-port      : OFF          Device      : roceo3811
Number of qps  : 1           Transport type : IB
Connection type : RC         Using SRQ    : OFF
PCIe relax order: ON
ibv_wr* API    : OFF
TX depth       : 128
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : Ethernet
GID index      : 3
Max inline data : 0[B]
rdma_cm QPs    : OFF
Data ex. method : Ethernet
-----
local address: LID 0000 QPN 0x2c02 PSN 0xab5dfe RKey 0x2000007 VAddr 0x007ff32cb4a000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:13
remote address: LID 0000 QPN 0x2c02 PSN 0x987205 RKey 0x2000207 VAddr 0x007fee3ba00000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:11
-----
Completion with error at client
Failed status 12: wr_id 0 syndrom 0xa
scnt=128, ccnt=0
Failed to complete run_iter_bw function successfully
```

Failed Status 12 means that the RoCE packets cannot be transferred across the connection and point to a networking error. Often (not always) this is due to the switch MTU not being set to 9000. On some switches, if the switch port belongs to a VLAN, the VLAN MTU size must be set to 9000 as well. By default, perfest uses a 64 KB msg size which requires multiple 4096 byte MTUs to transfer. Therefore, if the switch MTU is not configured to 9000, the switch drops 4096 byte RoCE packets.

3. How to check if `ib_peer_mem` is part of the kernel or if it must be loaded as `ib_peer_mem.ko` kernel module. Use the following command:

```
cat /proc/kallsyms | grep -i ib_register_peer_memory_client
```

If the `ib_register_peer_memory_client` symbol is provided by `ib_uverbs`, then `ib_peer_mem` is part of the kernel. Installing the Broadcom software stack will not install and load the `ib_peer_mem.ko` kernel module. If the output of the `cat` command shows that the `ib_register_peer_memory_client` symbol is provided by `ib_peer_mem`, it means that the Broadcom `ib_peer_mem` kernel module is loaded into the kernel. If the `cat` command returns empty, the `ib_peer_mem` is not yet loaded into the kernel. Installing the Broadcom stack should load the module.

```
$ cat /proc/kallsyms | grep -i ib_register_peer_memory_client
0000000000000000 r __kstrtab_ib_register_peer_memory_client [ib_uverbs]
0000000000000000 r __kstrtabns_ib_register_peer_memory_client [ib_uverbs]
0000000000000000 r __ksymtab_ib_register_peer_memory_client [ib_uverbs]
0000000000000000 r __crc_ib_register_peer_memory_client [ib_uverbs]
0000000000000000 r __export_symbol_ib_register_peer_memory_client [ib_uverbs]
0000000000000000 t __pfx_ib_register_peer_memory_client [ib_uverbs]
0000000000000000 T ib_register_peer_memory_client [ib_uverbs]
```

4. Make sure `nvidia-peermem` module is loaded when using NVIDIA GPUs.

```
lsmod | grep peer
nvidia_peermem      16384  0
nvidia              9744384  3 nvidia_uvm,nvidia_peermem,nvidia_modeset
ib_uverbs           196608  3 nvidia_peermem,bnxt_re,rdma_ucm
```

5. QOS (PFC, DSCP, ETS) is not configured after RoCE driver load or host reboot.

If the `NICCLI getqos` command shows an output similar to the following, the QOS for RoCE is not configured.

```
$ sudo niccli -i <index | pci b:d:f | mac> qos --ets --show
IEEE 8021QAZ ETS Configuration TLV:
    Prio_MAP: 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
    TC Bandwidth: 0% 0% 0% 0% 0% 0% 0% 0%
    TSA_MAP: 0:strict 1:strict 2:strict 3:strict 4:strict 5:strict 6:strict 7:strict
IEEE 8021QAZ PFC TLV:
    PFC enabled: none
```

This generally means that firmware-based DCBX or firmware-based LLDP is enabled and the RoCE driver (`bnxt_re`) did not configure the QOS for the given NIC interface. The `dmesg` logs have errors similar to the following:

```
infiniband bnxt_re2: Fail to setets rc:-22
infiniband bnxt_re2: Fail to initialize Flow control
```

Ensure that the following NVM CFGs are disabled. See [Firmware Based DCBX NVM CFG on NIC](#).

- a. `dcbx_mode`
- b. `lldp_nearest_bridge`
- c. `lldp_nearest_non_tpmr_bridge`

6. The following error is seen when running RDMA-related commands such as:

```
$ ibv_devinfo -vvv
libibverbs: Warning: Driver bnxt_re does not support the kernel ABI of 6 (supports 1 to 1) for device /sys/class/infiniband/bnxt_re0
libibverbs: Warning: Driver bnxt_re does not support the kernel ABI of 6 (supports 1 to 1) for device /sys/class/infiniband/bnxt_re0
```

This means the in-box `libbnxt_re` library is being used. We need to use the out of box `libbnxt_re` library. This can happen if the out of box library has not been installed or the `rdma-core` package on the host has been updated. Use the command `strace ibv_devinfo 2>&1 | grep libbnxt_re | grep -v 'No such file'` identify the path of the current `libbnxt_re` library and rename or delete it.

```
$ strace ibv_devinfo 2>&1 | grep libbnxt_re | grep -v 'No such file'
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libibverbs/libbnxt_re-rdmav34.so",
O_RDONLY|O_CLOEXEC) = 3
```

7. Broadcom software installation fails due to errors similar to:

```
bnxt_re: disagrees about version of symbol ib_umem_release
bnxt_re: Unknown symbol ib_umem_release (err -22)
bnxt_re: disagrees about version of symbol ib_modify_qp_is_ok
bnxt_re: Unknown symbol ib_modify_qp_is_ok (err -22)
```

Check if the MLNX OFED is installed on the host. MLNX OFED install removes the standard RDMA/ib_core kernel modules and rdma-core user space libraries from the host and installs MLNX OFED specific variants. After that, the install of the Broadcom RoCE software stack fails. Check for the presence of /usr/sbin/ofed_uninstall.sh shell script. If the script exists, it indicates MLNX OFED is installed on the host. Execute /usr/sbin/ofed_uninstall.sh which uninstalls MLNX OFED from the host, then reboot the host. After reboot, the /usr/sbin/ofed_uninstall.sh script no longer exists and the Broadcom stack installs.

8. RoCE (non-GPU) performance using perftest (ib_write_bw) is low (~235 Gbps) but GPU based perftest (ib_write_bw) performance is at line rate (~320 Gbps).

One of the reasons for low RoCE (non-GPU) performance, but good GPU based RoCE performance is the **Memory Interleaving** BIOS setting on some hosts. The **Memory Interleaving** option should not be disabled and set to **Auto**. If disabled, RoCE (non-GPU) performance suffers due to Host Memory BW limitation.

9. How to capture RoCE packets on a host using tcpdump.

For debug purposes, RoCE packets can be captured on a host via tcpdump. RoCE packet capture is not a production feature as it has its side effects, but can be very useful in debugging certain problems. The following two NVM CFG options must be modified to enable RoCE packet capture.

- a. enable_sriov should be set to value 0 (Disabled)
- b. default_evb_modes should be set to value 3 (none)

To check the option values, use the following commands:

```
sudo niccli -i <index | pci b:d:f | mac> nvm --getoption enable_sriov
# The highlighted item will change depending on the index or PCIE BDF or MAC
```

```
# Example:
sudo niccli -i 1 nvm --getoption enable_sriov
```

```
sudo niccli -i <index | pci b:d:f | mac> nvm --getoption default_evb_mode --scope <pf number>
# First highlighted item will change depending on the index or PCIE BDF or MAC
# Second highlighted item will change depending on the PF
```

```
# Example:
sudo niccli -i 1 nvm --getoption default_evb_mode --scope 0
```

To set the value of the options, use the following commands:

```
sudo niccli -i <index | pci b:d:f | mac> nvm --setoption enable_sriov --value 0
# The highlighted item will change depending on the index or PCIE BDF or MAC
```

```
# Example:
$ sudo niccli -i 1 nvm --setoption enable_sriov --value 0

$ sudo niccli -i <index | pci b:d:f | mac> nvm --setoption default_evb_mode --scope <pf number> -
-value 3
# First highlighted item will change depending on the index or PCIE BDF or MAC
# Second highlighted item will change depending on the PF
```

```
# Example:
$ sudo niccli -i 1 nvm --setoption default_evb_mode --scope 0 --value 3
```

A reboot is required for the modified NVM CFGs to take effect. After the reboot, tcpdump must be run on the Ethernet interface name (not the RoCE interface name). See the following example:

```
$ rdma link show
link bnxt_re0/1 state ACTIVE physical_state LINK_UP netdev enp30s0np0
link bnxt_re1/1 state ACTIVE physical_state LINK_UP netdev enp67s0np0
link bnxt_re2/1 state ACTIVE physical_state LINK_UP netdev enp86s0np0
link bnxt_re3/1 state ACTIVE physical_state LINK_UP netdev enp105s0np0
link bnxt_re4/1 state ACTIVE physical_state LINK_UP netdev enp160s0np0
link bnxt_re5/1 state ACTIVE physical_state LINK_UP netdev enp195s0np0
link bnxt_re6/1 state ACTIVE physical_state LINK_UP netdev enp213s0np0
link bnxt_re7/1 state ACTIVE physical_state LINK_UP netdev enp231s0np0
```

```
$ tcpdump -i enp30s0np0 udp
```

Starting with the 2.34 release, roce pkt capture via tcpdump requires tcpdump to be executed on the RoCE interface name instead of the ethernet interface name.

```
$ tcpdump -i bnxt_re0 udp
```

5.2 BCM_SOSREPORT

The Broadcom SOS reporting tool builds on the open source `sosreport` tool to collect system information for support purposes. The following sections describe how to create the `bcm_sosreport` package and how to install and run the tool. The report that is generated can be sent to a Broadcom support representative for analysis.

On a Ubuntu host, the tool can be installed and executed as follows:

```
To install the tool on a Ubuntu Host
$ dpkg -i bcm_sosreport_<version>.deb
```

```
To execute the tool
$ bcm_sosreport
```

See the [following link](#) for additional details on `bcm_sosreport`.

Chapter 6: Installing NVIDIA GPU Drivers

The NVIDIA GPU driver installation process, outlined in the [NVIDIA GPU Driver Installation](#) section, sets up the NVIDIA GPU driver.

Chapter 7: Running NCCL Collectives

Install NCCL as described in [Peer Memory Direct Configuration with BCM957608](#).

To run NCCL across multiple nodes in a cluster, Open MPI installation is required. Open MPI is only used to launch the NCCL tests/processes across multiple nodes. Additionally, UCX can be used along with Open MPI to launch the NCCL tests/processes across multiple nodes. NCCL itself does not use openMPI or UCX for its operation. See [Table 3](#) for the software component versions used for AI/ML/HPC applications. It is recommended to install and execute Open MPI, UCX, and nccl-tests as a non-root user. Passwordless SSH should be set up for the non-root user as well.

Table 3: NVIDIA GPU

Component Name	Component Version
Cuda version	12.8
UCX	1.18
OpenMPI	5.0.6
OSU	7.5.1
NCCL version	2.25.1
GDRCOPY git clone	https://github.com/NVIDIA/gdrCOPY.git (current version GDR-Copy 2.4.4)

7.1 Setting up the Environmental Variable

To set up environment variables, use the following commands:

```
export UCX_VER=vXXX
export OMPI_VER=vXXX
```

NOTE: See [Table 3](#) for the correct version numbers.

7.2 Installing UCX for NVIDIA GPUs

To install UCX for NVIDIA GPUs, use the following examples:

```
cd $HOME
git clone --recursive -b ${UCX_VER} https://github.com/openucx/ucx.git
cd ucx
./autogen.sh
mkdir ucx_install
mkdir build
cd build
../configure --with-cuda=/usr/local/cuda --with-verbs --with-go=no --enable-debug --prefix=$HOME/ucx/ucx_install
make -j $(nproc)
make -j $(nproc) install

# Verify by running
$HOME/ucx/ucx_install/bin/ucx_info -d
```

7.3 Installing Open MPI for NVIDIA GPUs

To install Open MPI for NVIDIA GPUs, use the following commands:

```
# The Open MPI build needs to point to the UCX installation $HOME/ucx/ucx_install
# as shown in the steps below.

# On Ubuntu please install flex
sudo apt install flex

cd $HOME
git clone --recursive -b $OMPI_VER https://github.com/open-mpi/ompi.git
cd ompi

./autogen.pl
mkdir build
mkdir ompi_install

cd build

../configure --prefix=$HOME/ompi/ompi_install --with-cuda=/usr/local/cuda --with-cuda-libdir=/usr/local/cuda/targets/x86_64-linux/lib/stubs --with-verbs --with-ucx=$HOME/ucx/ucx_install --enable-debug

make -j $(nproc)
make -j $(nproc) install

# Verify by running
$HOME/ompi/ompi_install/bin/ompi_info --parsable --all | grep mpi_built_with_cuda_support:value
$HOME/ompi/ompi_install/bin/ompi_info | grep "MPI extensions"
```

Open MPI BTL openib requires Broadcom NIC PCIE vendor_part_id 0x1760 to be added to the file \$HOME/ompi/ompi_install/share/openmpi/mca-btl-openib-device-params.ini, under section

```
[Broadcom BCM57XXX]
vendor_id = 0x14e4
```

The file `mca-btl-openib-device-params.ini` is installed as part of the Open MPI installation.

7.4 Compiling NCCL Tests

The NCCL tests are located on [GitHub](#). Clone the repo and build as follows. Note there is no need to install the binaries as they can be run directly from the home directory. If CUDA is not installed in `/usr/local/cuda`, you may specify `CUDA_HOME`. Similarly, if NCCL is not installed in `/usr`, you may specify `NCCL_HOME` while trying to do make.

In this example, the CUDA and NCCL libraries are in the appropriate folders as the NCCL test expects.

```
git clone https://github.com/NVIDIA/nccl-tests.git
cd nccl-tests/
make MPI=1 MPI_HOME=$HOME/mpi/mpi_install
```

With tests compiled, the following collectives can be tested:

- `all_gather`
- `all_reduce`
- `all_to_all`
- `all_to_allv`
- `broadcast`
- `gather`
- `reduce`
- `reduce_scatter`
- `scatter`
- `send_recv`

7.5 Single Node NCCL Collectives

Single Node NCCL collectives do not use RoCE by default and can be exercised with the Broadcom `bnxt_en` and `bnxt_re` drivers removed as well.

The `LD_LIBRARY_PATH` should be set correctly for a single node run though.

```
export LD_LIBRARY_PATH=$HOME/mpi/mpi_install/lib:$HOME/ucx/ucx_install/lib:$LD_LIBRARY_PATH
$ $HOME/nccl-tests/build/all_reduce_perf -b 8 -e 16g -f 2 -g 8
-----
WARNING: There was an error initializing an OpenFabrics device.

Local host:  COS-AI-R1C2S1
Local device: bnxt_re0
-----
# nThread 1 nGpus 8 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 1 graph: 0
#
# Using devices
# Rank 0 Group 0 Pid 234319 on COS-AI-R1C2S1 device 0 [0x18] NVIDIA H100 80GB HBM3
# Rank 1 Group 0 Pid 234319 on COS-AI-R1C2S1 device 1 [0x2b] NVIDIA H100 80GB HBM3
# Rank 2 Group 0 Pid 234319 on COS-AI-R1C2S1 device 2 [0x3a] NVIDIA H100 80GB HBM3
# Rank 3 Group 0 Pid 234319 on COS-AI-R1C2S1 device 3 [0x5e] NVIDIA H100 80GB HBM3
# Rank 4 Group 0 Pid 234319 on COS-AI-R1C2S1 device 4 [0x84] NVIDIA H100 80GB HBM3
# Rank 5 Group 0 Pid 234319 on COS-AI-R1C2S1 device 5 [0x9b] NVIDIA H100 80GB HBM3
# Rank 6 Group 0 Pid 234319 on COS-AI-R1C2S1 device 6 [0xae] NVIDIA H100 80GB HBM3
# Rank 7 Group 0 Pid 234319 on COS-AI-R1C2S1 device 7 [0xd7] NVIDIA H100 80GB HBM3
```

#	size (B)	count (elements)	type	redop	root	time (us)	out-of-place			in-place			
							algbw (GB/s)	busbw (GB/s)	#wrong	time (us)	algbw (GB/s)	busbw (GB/s)	#wrong
#	8	2	float	sum	-1	65.51	0.00	0.00	0	51.92	0.00	0.00	0
#	16	4	float	sum	-1	50.13	0.00	0.00	0	49.43	0.00	0.00	0
#	32	8	float	sum	-1	70.35	0.00	0.00	0	51.07	0.00	0.00	0
#	64	16	float	sum	-1	53.06	0.00	0.00	0	52.94	0.00	0.00	0
#	128	32	float	sum	-1	53.20	0.00	0.00	0	54.17	0.00	0.00	0
#	256	64	float	sum	-1	56.01	0.00	0.01	0	56.37	0.00	0.01	0
#	512	128	float	sum	-1	57.17	0.01	0.02	0	59.80	0.01	0.01	0
#	1024	256	float	sum	-1	60.11	0.02	0.03	0	59.23	0.02	0.03	0
#	2048	512	float	sum	-1	59.78	0.03	0.06	0	60.17	0.03	0.06	0
#	4096	1024	float	sum	-1	60.10	0.07	0.12	0	60.19	0.07	0.12	0
#	8192	2048	float	sum	-1	60.66	0.14	0.24	0	60.04	0.14	0.24	0
#	16384	4096	float	sum	-1	60.24	0.27	0.48	0	61.71	0.27	0.46	0
#	32768	8192	float	sum	-1	60.81	0.54	0.94	0	61.19	0.54	0.94	0
#	65536	16384	float	sum	-1	56.92	1.15	2.01	0	53.13	1.23	2.16	0
#	131072	32768	float	sum	-1	51.56	2.54	4.45	0	52.34	2.50	4.38	0
#	262144	65536	float	sum	-1	64.98	4.03	7.06	0	65.13	4.02	7.04	0
#	524288	131072	float	sum	-1	67.22	7.80	13.65	0	67.69	7.74	13.55	0
#	1048576	262144	float	sum	-1	70.47	14.88	26.04	0	69.98	14.98	26.22	0
#	2097152	524288	float	sum	-1	68.44	30.64	53.62	0	69.10	30.35	53.11	0
#	4194304	1048576	float	sum	-1	68.42	61.31	107.29	0	68.78	60.98	106.72	0
#	8388608	2097152	float	sum	-1	85.71	97.87	171.27	0	84.71	99.02	173.29	0
#	16777216	4194304	float	sum	-1	127.5	131.54	230.19	0	125.9	133.22	233.14	0
#	33554432	8388608	float	sum	-1	200.1	167.65	293.40	0	200.2	167.60	293.31	0
#	67108864	16777216	float	sum	-1	326.3	205.69	359.95	0	325.8	205.99	360.49	0
#	134217728	33554432	float	sum	-1	591.6	226.86	397.00	0	592.0	226.74	396.79	0
#	268435456	67108864	float	sum	-1	1120.0	239.68	419.43	0	1120.1	239.65	419.39	0
#	536870912	134217728	float	sum	-1	2161.9	248.33	434.57	0	2159.8	248.58	435.01	0
#	1073741824	268435456	float	sum	-1	4031.4	266.34	466.10	0	4032.3	266.29	466.00	0
#	2147483648	536870912	float	sum	-1	7974.0	269.31	471.30	0	7978.1	269.17	471.05	0
#	4294967296	1073741824	float	sum	-1	15822	271.46	475.05	0	15836	271.22	474.63	0
#	8589934592	2147483648	float	sum	-1	31436	273.25	478.18	0	31421	273.38	478.42	0
#	17179869184	4294967296	float	sum	-1	62563	274.60	480.55	0	62579	274.53	480.43	0
#	Out of bounds values : 0 OK												
#	Avg bus bandwidth : 152.97												
#													

7.6 MultiNode NCCL Collectives using Open MPI

Ensure the following prechecks are complete:

- NIC IPs and routings are configured
- Passwordless ssh has been setup between the hosts
- NUMA balancing is disabled on every host. This can be done at runtime using `echo 0 > /proc/sys/kernel/numa_balancing` or via `sysctl -w kernel.numa_balancing=0`. Another option is to add the following entry to the file `/etc/sysctl.d/99-sysctl.conf` so that the setting takes effect automatically after a reboot
- Each node used for the test has the `PATH` and `LD_LIBRARY_PATH` set correctly via `.bashrc` file as follows:

```
export LD_LIBRARY_PATH=$HOME/mpi/mpi_install/lib:$HOME/ucx/ucx_install/lib:$LD_LIBRARY_PATH
export PATH=$HOME/mpi/mpi_install/bin:$HOME/ucx/ucx_install/bin:$PATH
```

In the NCCL test below, two nodes are used to run the `all_reduce` collective. Each node has eight GPUs and eight NICs. Passwordless ssh is set up for this interface. Both the Hosts connect to the same leaf switch.

```
$ $HOME/mpi/mpi_install/bin/mpirun --np 16 -H COS-AI-R1C1S1:8,COS-AI-R1C2S1:8 \
-x NCCL_DEBUG=VERSION --mca pml ucx --mca btl ^openib \
-x NCCL_SOCKET_IFNAME=enp25s0np0 -x NCCL_IB_DISABLE=0 \
-x NCCL_IB_HCA=bnxt_re0:1,bnxt_re1:1,bnxt_re2:1,bnxt_re3:1,bnxt_re4:1,bnxt_re5:1,bnxt_re6:1,bnxt_re7:1 \
-x NCCL_IB_GID_INDEX=3 \
-x NCCL_P2P_NET_CHUNKSIZE=262144 \
$HOME/nccl-tests/build/all_reduce_perf -b 8 -e 16g -f 2 -g 1
# nThread 1 nGpus 1 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 1 graph: 0
#
# Using devices
# Rank 0 Group 0 Pid 220894 on COS-AI-R1C2S1 device 0 [0x18] NVIDIA H100 80GB HBM3
# Rank 1 Group 0 Pid 220895 on COS-AI-R1C2S1 device 1 [0x2b] NVIDIA H100 80GB HBM3
# Rank 2 Group 0 Pid 220896 on COS-AI-R1C2S1 device 2 [0x3a] NVIDIA H100 80GB HBM3
# Rank 3 Group 0 Pid 220898 on COS-AI-R1C2S1 device 3 [0x5e] NVIDIA H100 80GB HBM3
# Rank 4 Group 0 Pid 220901 on COS-AI-R1C2S1 device 4 [0x84] NVIDIA H100 80GB HBM3
# Rank 5 Group 0 Pid 220904 on COS-AI-R1C2S1 device 5 [0x9b] NVIDIA H100 80GB HBM3
# Rank 6 Group 0 Pid 220906 on COS-AI-R1C2S1 device 6 [0xae] NVIDIA H100 80GB HBM3
# Rank 7 Group 0 Pid 220909 on COS-AI-R1C2S1 device 7 [0xd7] NVIDIA H100 80GB HBM3
# Rank 8 Group 0 Pid 249511 on COS-AI-R1C1S1 device 0 [0x18] NVIDIA H100 80GB HBM3
# Rank 9 Group 0 Pid 249512 on COS-AI-R1C1S1 device 1 [0x2b] NVIDIA H100 80GB HBM3
# Rank 10 Group 0 Pid 249513 on COS-AI-R1C1S1 device 2 [0x3a] NVIDIA H100 80GB HBM3
# Rank 11 Group 0 Pid 249514 on COS-AI-R1C1S1 device 3 [0x5e] NVIDIA H100 80GB HBM3
# Rank 12 Group 0 Pid 249515 on COS-AI-R1C1S1 device 4 [0x84] NVIDIA H100 80GB HBM3
# Rank 13 Group 0 Pid 249518 on COS-AI-R1C1S1 device 5 [0x9d] NVIDIA H100 80GB HBM3
# Rank 14 Group 0 Pid 249521 on COS-AI-R1C1S1 device 6 [0xb0] NVIDIA H100 80GB HBM3
# Rank 15 Group 0 Pid 249524 on COS-AI-R1C1S1 device 7 [0xd7] NVIDIA H100 80GB HBM3
NCCL version 2.22.3+cuda12.6
#
#
#
# out-of-place in-place
# size count type redop root time algbw busbw #wrong time algbw busbw #wrong
# (B) (elements) (us) (GB/s) (GB/s) (us) (GB/s) (GB/s)
# 8 2 float sum -1 98.30 0.00 0.00 0 38.03 0.00 0.00 0
# 16 4 float sum -1 40.04 0.00 0.00 0 38.92 0.00 0.00 0
# 32 8 float sum -1 41.82 0.00 0.00 0 42.26 0.00 0.00 0
# 64 16 float sum -1 42.65 0.00 0.00 0 41.01 0.00 0.00 0
# 128 32 float sum -1 44.18 0.00 0.01 0 40.38 0.00 0.01 0
# 256 64 float sum -1 99.02 0.00 0.00 0 42.38 0.01 0.01 0
# 512 128 float sum -1 45.28 0.01 0.02 0 33.07 0.02 0.03 0
# 1024 256 float sum -1 46.54 0.02 0.04 0 42.05 0.02 0.05 0
# 2048 512 float sum -1 46.79 0.04 0.08 0 43.89 0.05 0.09 0
# 4096 1024 float sum -1 50.99 0.08 0.15 0 46.98 0.09 0.16 0
# 8192 2048 float sum -1 50.92 0.16 0.30 0 51.34 0.16 0.30 0
# 16384 4096 float sum -1 53.48 0.31 0.57 0 44.46 0.37 0.69 0
# 32768 8192 float sum -1 58.73 0.56 1.05 0 54.03 0.61 1.14 0
# 65536 16384 float sum -1 57.14 1.15 2.15 0 55.40 1.18 2.22 0
# 131072 32768 float sum -1 62.14 2.11 3.95 0 66.29 1.98 3.71 0
# 262144 65536 float sum -1 70.03 3.74 7.02 0 68.48 3.83 7.18 0
# 524288 131072 float sum -1 165.6 3.17 5.94 0 139.4 3.76 7.05 0
# 1048576 262144 float sum -1 141.7 7.40 13.87 0 151.7 6.91 12.96 0
# 2097152 524288 float sum -1 153.8 13.63 25.56 0 153.6 13.65 25.59 0
# 4194304 1048576 float sum -1 180.3 23.26 43.62 0 192.8 21.75 40.79 0
# 8388608 2097152 float sum -1 203.5 41.22 77.30 0 230.4 36.41 68.26 0
# 16777216 4194304 float sum -1 281.3 59.64 111.82 0 279.8 59.97 112.44 0
# 33554432 8388608 float sum -1 410.6 81.71 153.21 0 411.1 81.63 153.05 0
```

67108864	16777216	float	sum	-1	528.4	127.01	238.15	0	530.1	126.60	237.38	0
134217728	33554432	float	sum	-1	872.1	153.90	288.57	0	852.6	157.43	295.18	0
268435456	67108864	float	sum	-1	1429.1	187.84	352.19	0	1433.1	187.32	351.22	0
536870912	134217728	float	sum	-1	2539.5	211.41	396.40	0	2416.4	222.18	416.58	0
1073741824	268435456	float	sum	-1	4493.3	238.96	448.06	0	4478.9	239.73	449.50	0
2147483648	536870912	float	sum	-1	8745.0	245.57	460.44	0	8747.1	245.51	460.33	0
4294967296	1073741824	float	sum	-1	17241	249.11	467.09	0	17256	248.90	466.69	0
8589934592	2147483648	float	sum	-1	34313	250.34	469.38	0	34337	250.16	469.06	0
17179869184	4294967296	float	sum	-1	68552	250.61	469.89	0	68595	250.45	469.60	0

Out of bounds values : 0 OK
 # Avg bus bandwidth : 126.377

Chapter 8: NIC and Ethernet Switch Configuration

Broadcom publishes an Ethernet NIC user guide that is publicly available. This guide contains detailed information on how to configure and use Broadcom NICs, including how to use the NIC for RoCE and PeerMem Direct.

- [Broadcom Ethernet Network Adapter User Guide](#)
- [RDMA over Converged Ethernet \(RoCE\)](#)

Broadcom and Arista jointly publish a Broadcom RoCE Deployment guide that has more details on how to configure Arista switches for RoCE and RoCE congestion Control.

- [Lossless Network for AI/ML/Storage/HPC with RDMA](#)

Appendix A: Compiling Broadcom NIC Software from Source

This appendix shows example Linux shell scripts that can be used to compile and install Broadcom RoCE Kernel drivers and the user-space library `libbnxt_re` from source code. The script examples use the 232 release and the 232 release pkg file names as an example. Depending on the actual release and release pkg names being used, the script can be updated. Assuming the scripts below are placed in a file named `brcm_sw_compile_install.sh`, execute the script as follows:

```
chmod 777 brcm_sw_compile_install.sh

sudo ./brcm_sw_compile_install.sh | tee build_log.txt
# or
sudo bash ./brcm_sw_compile_install.sh | tee build_log.txt
```

A.1 Ubuntu: Install Script for NIC Software (Compiling from Source Code)

Install the NIC software using the following script:

```
#!/bin/bash

echo -e "\n\n====Installing required pkgs====\n\n"
sudo apt install linux-headers-"$(uname -r)" libelf-i
sudo apt install gcc make libtool autoconf librdmacm-dev rdmacm-utils infiniband-diags ibverbs-utils
perftest ethtool libibverbs-dev rdma-core strace

echo -e "\n\n====Compiling and installing L2 and RoCE kernel drivers====\n\n"
# Highlighted item will change depending on the release
tar -xf netxtreme-peer-mem-234.0.154.0.tar.gz
cd netxtreme-peer-mem-234.0.154.0
make
sudo make install
sudo depmod -a
cd ..
echo -e "\n\n====L2 and RoCE Kernel Driver Compile Complete====\n\n"

echo -e "\n\n====Loading built modules into the kernel====\n\n"

#Unload the current version of the loaded drivers incase they are loaded.
sudo rmdir /sys/kernel/config/bnxt_re/* 2> /dev/null
sudo modprobe -r bnxt_re
sudo modprobe -r ib_peer_mem
sudo modprobe ib_peer_mem
#Make sure the 2 commands below, seperated by ";" are executed together in a single line
sudo rmmod bnxt_en; sudo modprobe bnxt_en
sudo modprobe bnxt_re

sudo update-initramfs -u -k `uname -r`
```

```

echo -e "\n\n=====  

file=====  

\n\n"

if [[ $(grep '^* soft memlock unlimited$' /etc/security/limits.conf) ]]; then
    echo "Soft MemLock ok"
else
    echo "Adding soft memlock unlimited to /etc/security/limits.conf"
    sudo sh -c "echo '* soft memlock unlimited' >> /etc/security/limits.conf"
fi

if [[ $(grep '^* hard memlock unlimited$' /etc/security/limits.conf) ]]; then
    echo "Hard MemLock ok"
else
    echo "Adding hard memlock unlimited to /etc/security/limits.conf"
    sudo sh -c "echo '* hard memlock unlimited' >> /etc/security/limits.conf"
fi

echo -e "\n\n=====  

# Highlighted item will change depending on the release  

tar -xf libbnxt_re-234.0.154.0.tar.gz  

cd libbnxt_re-234.0.154.0  

sh autogen.sh  

./configure  

make  

find /usr/lib64/ /usr/lib -name "libbnxt_re-rdmav*.so" -exec mv {} {}.inbox \;  

sudo make install all  

sudo sh -c "echo /usr/local/lib >> /etc/ld.so.conf"  

sudo ldconfig  

sudo cp -f bnxt_re.driver /etc/libibverbs.d/

find . -name "*.so" -exec md5sum {} \;  

BUILT_MD5SUM=$(find . -name "libbnxt_re-rdmav*.so" -exec md5sum {} \; | cut -d " " -f 1)  

echo -e "\n\nmd5sum of the built libbnxt_re is $BUILT_MD5SUM"

echo -e "\n\n=====  

cd ..  

echo -e "\nRunning strace"  

strace ibv_devinfo 2>&1 | grep libbnxt_re | grep -v 'No such file'  

INSTALLED_LIB_PATH=$(strace ibv_devinfo 2>&1 | grep libbnxt_re | grep -v 'No such file' | cut -d ","  

-f 2 | tr -d "\")  

echo -e "\n\nInstalled libbnxt_re is at path $INSTALLED_LIB_PATH\n"

if [[ -z "$INSTALLED_LIB_PATH" ]]; then
    echo -e "Failed to find location of installed libbnxt_re, exiting...\n\n\n"  

    exit 4
fi

md5sum $INSTALLED_LIB_PATH  

INSTALLED_MD5SUM=$(md5sum $INSTALLED_LIB_PATH | cut -d " " -f 1)  

echo -e "md5sum of the installed library is $INSTALLED_MD5SUM"

```

```

echo -e "\n\nlibbnext_re BUILT_MD5SUM=$BUILT_MD5SUM, INSTALLED_MD5SUM=$INSTALLED_MD5SUM \n\n"

if [[ -z "$BUILT_MD5SUM" ]]; then
    echo -e "Failed to get the md5sum of the built libbnext_re lib\n\n\n"
    exit 1
elif [[ -z "$INSTALLED_MD5SUM" ]]; then
    echo -e "Failed to get the md5sum of the installed libbnext_re lib\n\n\n"
    exit 2
elif [[ "$BUILT_MD5SUM" = "$INSTALLED_MD5SUM" ]]; then
    echo -e "MD5Sum of the built and installed libbnext_re match"
else
    echo -e "MD5Sum of the built and installed libbnext_re do not match \n\n\n"
    exit 3
fi

echo -e "\n\n\n"

```

A.2 RHEL: Install Script for NIC Software (Compiling from Source Code)

Install the NIC software using the following script:

```

#!/bin/bash

echo -e "\n\n====Installing required pkgs====\n\n"
sudo yum install -y "kernel-devel-uname-r == $(uname -r)" elfutils-libelf-devel
sudo yum install -y libibverbs-devel qperf perftest infiniband-diags make gcc kernel-devel autoconf
libtool libibverbs-utils rdma-core-devel librdmacm-utils strace

echo -e "\n\n====Compiling and installing L2 and RoCE kernel drivers====\n\n"
# Highlighted item will change depending on the release
tar -xf netxtreme-peer-mem-234.0.154.0.tar.gz
cd netxtreme-peer-mem-234.0.154.0
make
sudo make install
sudo depmod -a
cd ..
echo -e "\n\n====L2 and RoCE Kernel Driver Compile Complete====\n\n"

echo -e "\n\n====Loading built modules into the kernel====\n\n"
#Unload the current version of the loaded drivers incase they are loaded.
sudo rmdir /sys/kernel/config/bnxt_re/* 2> /dev/null
sudo modprobe -r bnxt_re
sudo modprobe -r ib_peer_mem
sudo modprobe ib_peer_mem
#Make sure the 2 commands below, seperated by ";" are executed together in a single line
sudo rmmod bnxt_en; sudo modprobe bnxt_en
sudo modprobe bnxt_re

sudo dracut -f

echo -e "\n\n==== Checking and updating /etc/security/limit.conf
file==== \n\n"

if [[ $(grep '^* soft memlock unlimited$' /etc/security/limits.conf) ]]; then
    echo "Soft MemLock ok"

```

```

else
    echo "Adding soft memlock unlimited to /etc/security/limits.conf"
    sudo sh -c "echo '* soft memlock unlimited' >> /etc/security/limits.conf"
fi

if [[ $(grep '^* hard memlock unlimited$' /etc/security/limits.conf) ]]; then
    echo "Hard MemLock ok"
else
    echo "Adding hard memlock unlimited to /etc/security/limits.conf"
    sudo sh -c "echo '* hard memlock unlimited' >> /etc/security/limits.conf"
fi

echo -e "\n\n=====Compiling RoCE Lib now===== \n\n"
# Highlighted item will change depending on the release
tar -xf libbnxt_re-234.0.154.0.tar.gz
cd libbnxt_re-234.0.154.0
sh autogen.sh
./configure
make
find /usr/lib64/ /usr/lib -name "libbnxt_re-rdmav*.so" -exec mv {} {}.inbox \;
make install all
sudo sh -c "echo /usr/local/lib >> /etc/ld.so.conf"
sudo ldconfig
cp -f bnxt_re.driver /etc/libibverbs.d/

find . -name "*.so" -exec md5sum {} \;
BUILT_MD5SUM=$(find . -name "libbnxt_re-rdmav*.so" -exec md5sum {} \; | cut -d " " -f 1)
echo -e "\n\nmd5sum of the built libbnxt_re is $BUILT_MD5SUM"

echo -e "\n\n=====RoCE userlib compile complete===== \n\n"
cd ..
echo -e "\nRunning strace"
strace ibv_devinfo 2>&1 | grep libbnxt_re | grep -v 'No such file'
INSTALLED_LIB_PATH=$(strace ibv_devinfo 2>&1 | grep libbnxt_re | grep -v 'No such file' | cut -d "," -f 2 | tr -d "\")
echo -e "\n\nInstalled libbnxt_re is at path $INSTALLED_LIB_PATH\n"

if [[ -z "$INSTALLED_LIB_PATH" ]]; then
    echo -e "Failed to find location of installed libbnxt_re, exiting...\n\n"
    exit 4
fi

md5sum $INSTALLED_LIB_PATH
INSTALLED_MD5SUM=$(md5sum $INSTALLED_LIB_PATH | cut -d " " -f 1)

echo -e "md5sum of the installed library is $INSTALLED_MD5SUM"

echo -e "\n\nlibbnxt_re BUILT_MD5SUM=$BUILT_MD5SUM, INSTALLED_MD5SUM=$INSTALLED_MD5SUM \n\n"

if [[ -z "$BUILT_MD5SUM" ]]; then
    echo -e "Failed to get the md5sum of the built libbnxt_re lib\n\n"
    exit 1
elif [[ -z "$INSTALLED_MD5SUM" ]]; then
    echo "Failed to get the md5sum of the installed libbnxt_re lib\n\n"

```

```
    exit 2
elif [[ "$BUILT_MD5SUM" = "$INSTALLED_MD5SUM" ]]; then
    echo -e "MD5Sum of the built and installed libbnxt_re match"
else
    echo -e "MD5Sum of the built and installed libbnxt_re do not match \n\n\n"
    exit 3
fi

echo -e "\n\n\n"
```

Appendix B: Script for Disabling ACS

The following shell script can be used to disable ACS on a host that does not disable ACS by default via the BIOS. The script disables ACS on all ports that support ACS.

B.1 Disable PCIe ACS

To disable PCIe ACS, use the following commands:

```
#!/bin/bash
#
# Disable ACS on every device that supports it
#
PLATFORM=$(dmidecode --string system-product-name)
logger "PLATFORM=${PLATFORM}"
# Enforce platform check here.
#case "${PLATFORM}" in
#    "OAM"*)
#        #logger "INFO: Disabling ACS is no longer necessary for ${PLATFORM}"
#        #exit 0
#        #;;
#*)
#    #;;
#esac
# must be root to access extended PCI config space
if [ "$EUID" -ne 0 ]; then
    echo "ERROR: $0 must be run as root"
    exit 1
fi
for BDF in `lspci -d "*:*:*" | awk '{print $1}'`; do
    # skip if it doesn't support ACS
    setpci -v -s ${BDF} ECAP_ACS+0x6.w > /dev/null 2>&1
    if [ $? -ne 0 ]; then
        #echo "${BDF} does not support ACS, skipping"
        continue
    fi
    logger "Disabling ACS on `lspci -s ${BDF}`"
    setpci -v -s ${BDF} ECAP_ACS+0x6.w=0000
    if [ $? -ne 0 ]; then
        logger "Error enabling directTrans ACS on ${BDF}"
        continue
    fi
    NEW_VAL=`setpci -v -s ${BDF} ECAP_ACS+0x6.w | awk '{print $NF}'`
    if [ "${NEW_VAL}" != "0000" ]; then
        logger "Failed to enabling directTrans ACS on ${BDF}"
        continue
    fi
done
exit 0
```

B.2 List all PCIe Devices that Support ACS

To list all PCIe devices, use the following commands:

```
#!/bin/bash
#
for BDF in `lspci -n | awk '{print $1}'`; do
    if lspci -vvv -s "$BDF" | grep -qi ACSctl; then
        lspci -vvv -s "$BDF" | head -1
        lspci -vvv -s "$BDF" | grep -i ACSctl
    fi
done
```

B.3 List all PCIe Devices with ACS Enabled

To list all PCIe devices with ACS enabled, use the following commands:

```
#!/bin/bash
#
for BDF in `lspci -n | awk '{print $1}'`; do
    if lspci -vvv -s "$BDF" | grep -i ACSctl | grep -qi SrcValid+; then
        lspci -vvv -s "$BDF" | head -1
        lspci -vvv -s "$BDF" | grep -i ACSctl
    fi
done
```

B.4 List all PCIe Devices with ACS Disabled

To list all PCIe devices with ACS disabled, use the following commands:

```
#!/bin/bash
#
for BDF in `lspci -n | awk '{print $1}'`; do
    if lspci -vvv -s "$BDF" | grep -i ACSctl | grep -qi SrcValid-; then
        lspci -vvv -s "$BDF" | head -1
        lspci -vvv -s "$BDF" | grep -i ACSctl
    fi
done
```

Appendix C: PCIe Link Speed and Width Related Scripts

This section provides scripts to get the PCIe link speed and width scripts.

C.1 Displaying the Link Speed and Link Width of Every PCIe Component

To display the link speed and link width, use the following commands:

```
#!/bin/bash
#
for BDF in `lspci -d "*:*:*" | awk '{print $1}'`; do
    if lspci -vvv -s "$BDF" | grep -q LnkSta; then
        lspci -vvv -s "$BDF" | head -1
        lspci -vvv -s "$BDF" | grep LnkSta:
    fi
done
```

C.2 Display Every PCIe Component with Downgraded Speed or Downgraded Width

To display every PCIe component with downgraded speed and width, use the following commands:

```
#!/bin/bash
#
for BDF in `lspci -d "*:*:*" | awk '{print $1}'`; do
    if lspci -vvv -s "$BDF" | grep -q downgraded; then
        lspci -vvv -s "$BDF" | head -1
        lspci -vvv -s "$BDF" | grep -i downgraded
    fi
done
```

Appendix D: References

D.1 Broadcom Ethernet Network Adapter User Guide

<https://techdocs.broadcom.com/us/en/storage-and-ethernet-connectivity/ethernet-nic-controllers/bcm957xxx/adapters.html>

Appendix E: Terminology

This section provides terminology definitions for terms used in this document.

Table 4: Terminology

Acronym	Meaning
ACS	PCIe Access Control Service
ARP	Address Resolution Protocol
BTL	Byte Transfer Layer
CNP	Congestion Notification Packet
DCBX	Data Center Bridging Exchange protocol
DCQCN	Data Centre quantized Congestion Notification
DCQCN-P	Data Centre quantized Congestion Notification-Probabilistic
DCQCN-D	Data Centre quantized Congestion Notification-Deterministic
DKMS	Dynamic Kernel Module Support
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification
ETS	Enhanced Transmission Selection
IOMMU	Input Output Memory Management Unit
L2	Ethernet as Layer 2 protocol in the OSI model
LLDP	Link Layer Discovery Protocol
MLNX_OFED	Mellanox OFED driver pkg
MTU	Maximum Transmission Unit
NUMA	Non Uniform Memory Access
NCCL	NVIDIA Communications Collective Library
OCP	Open Compute Project
Open MPI	Open Message Passing Interface
PFC	Priority Flow Control
RDMA	Remote Direct Memory Access
NIC	Network Interface Card
RoCE	RDMA over Converged Ethernet
SONiC	Software for Open Networking in the cloud
UCX	Unified Communication X
VLAN	Virtual Local Area Network
XGMI	AMD Infinity Fabric

Revision History

957608-AN304; July 29, 2025

Updated:

- Updated formatting issue.

957608-AN303; July 9, 2025

Updated:

- Updated for 234 version of software.

957608-AN302; April 15, 2025

Updated:

- Replaced NICCLI getoption -name commands with nvm-getoption commands throughout.

957608-AN301; March 20, 2025

Updated:

- Broadcom Ethernet NIC Software Installation with Peer Memory Direct
- Debugging Thor2 NIC

Added:

- Verifying the Correct RoCE QOS Configuration
- Ethernet Leaf Switch Port Configuration for 24-bit Subnet Scheme on Juniper QFX5240 Switch
- Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Juniper QFX5240 Switch
- Example: Juniper QFX5240 Switch and 31-bit Subnet Scheme

957608-AN300; October 22, 2024

Initial release.

