



BCM957608

Ethernet Networking Guide for AMD Instinct MI300X 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 AMD Instinct™ MI300 Series Accelerators	9
2.1.1 ROCm Installation	9
2.1.1.1 Example Instructions for Ubuntu 22.04	9
2.2 Broadcom Ethernet NIC Software Installation with Peer Memory Direct	10
2.2.1 Using the Broadcom Automated Installer to Update Software and Firmware for Peer Memory Direct	10
2.2.1.1 Example On Using the Broadcom Installer	12
2.2.2 Verifying the Correct Driver and Firmware Versions	12
2.2.3 Verifying the Correct RoCE QOS Configuration	13
2.2.4 Using the Broadcom Installer on a Host with Multiple NICs	14
2.2.5 Manually Compiling the Broadcom Host Software from Source Code for Peer Memory Direct	14
2.2.6 Installing the RoCE QOS Configuration (bnxt_re_conf) pkg Manually	15
2.2.7 Installing the NIC Firmware Manually	15
2.2.7.1 Verifying the Correct Driver and Firmware Version	16
2.2.7.2 Configuring RoCE Support	16
2.2.7.3 Enable RDMA option on the NIC	16
2.2.7.4 Enable RoCE Performance Profile on the NIC	16
2.2.7.5 Enable PCIe Relaxed Ordering on the NIC	17
2.2.7.6 Firmware Based DCBx NVM CFG on NIC	18
2.3 ACS and IOMMU Settings	19
2.3.1 Hosts with AMD CPUs	19
2.3.2 Hosts with Intel CPUs	19
2.3.3 Host Memory	19
2.4 Configuring Routing for the BackEnd Network	20
2.4.1 Single Leaf switch Topology with 24-bit subnets	20
2.4.1.1 Host 1 netplan file:/etc/netplan/00-installer-config-24bit-subnet-host1.yaml	21
2.4.1.2 Host 2 netplan file:/etc/netplan/00-installer-config-24bit-subnet-host2.yaml	24
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	27
2.4.1.4 Ethernet Leaf Switch Port Configuration for 24-bit Subnet Scheme on Juniper QFX5240 Switch	30
2.4.2 Single Leaf Switch Topology with 31-bit Subnets	34
2.4.2.1 Host 1 netplan file:/etc/netplan/00-installer-config-host1.yaml	36
2.4.2.2 Host 2 netplan file:/etc/netplan/00-installer-config-host2.yaml	39

2.4.2.3 Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Dell Z9664 Switch Running SONiC OS.....	43
2.4.2.4 Ethernet Leaf Switch Port Configuration for 31-bit Subnet Scheme on Juniper QFX5240 Switch	45
2.4.3 Confirm Routing Between Different NICs Across Different Hosts.....	47
2.5 Ethernet Switch Configuration for QoS and Congestion Control	48
2.5.1 Example: Arista 7060CX (DCQCN-P at 400G) and 31-bit Subnet Scheme	50
2.5.2 Example: Dell Z9664 Switch and Supermicro SSE-T8032 Switch Running SONiC OS and 31-bit Subnet Scheme	51
2.5.3 Example: Juniper QFX5240 Switch and 31-bit Subnet Scheme.....	51
2.6 Final Checks and Settings for Optimal Performance	55
2.7 Installing and Compiling PerfTest with AMD GPU Support	56
2.8 Validating Peer Memory Direct Support with PerfTest	57
2.8.1 Using AMD GPU	57
2.8.2 Example – ib_write_bw Test Using Broadcom NIC with AMD GPU	57
Chapter 3: System BIOS	60
3.1 BIOS Setting Recommendations	60
Chapter 4: Atlas2 PCIe Switch Configuration	61
Chapter 5: Debugging Thor2 NIC	62
5.1 Frequently Asked Questions and Troubleshooting.....	62
5.2 BCM_SOSREPORT	65
Chapter 6: Installing AMD GPU Drivers	66
Chapter 7: Debugging AMD Instinct MI300 Series Accelerators	67
Chapter 8: Running RCCL Collectives	68
8.1 Setting up the Environmental Variable	68
8.2 Installing UCX for AMD GPUs	68
8.3 Installing Open MPI for AMD GPUs	68
8.4 Compiling RCCL Tests	69
8.5 Single Node RCCL Collectives	70
8.5.1 Topology and Sample Test Results	70
8.6 Testing Single Node RCCL Collectives Using NICs	71
8.7 MultiNode RCCL Collectives Using Open MPI	72
8.7.1 Prechecks	72
8.7.2 Topology and Sample Test Results	72
8.7.2.1 Test: All-to-All	73
8.7.2.2 Test: All-Reduce	76
8.7.3 Debugging RCCL.....	79
8.7.3.1 RCCL Environment Variable: NCCL_DEBUG	79
8.7.3.2 RCCL Environment Variable: NCCL_SOCKET_IFNAME	79
8.7.3.3 RCCL Environment Variable: NCCL_NET_GDR_LEVEL	79

8.7.3.4 RCCL Environment Variable: NCCL_IB_HCA 79

Chapter 9: NIC and Ethernet Switch Configuration 80

Appendix A: Compiling Broadcom NIC Software from Source 81

 A.1 Ubuntu: Install Script for NIC Software (Compiling from Source Code) 81

 A.2 RHEL: Install Script for NIC Software (Compiling from Source Code) 83

Appendix B: Helpful ROCm Commands 86

 B.1 Checking the Type of GPUs on the Host 86

 B.2 Checking the PCIe BUS ID of Each GPU 87

 B.3 Checking the Processes Running on Each GPU 87

 B.4 Checking the PCIe Bandwidth in Use for Each GPU 89

Appendix C: Script for Disabling ACS 90

 C.1 Disable PCIe ACS 90

 C.2 List all PCIe Devices that Support ACS 91

 C.3 List all PCIe Devices with ACS Enabled 91

 C.4 List all PCIe Devices with ACS Disabled 91

Appendix D: PCIe Link Speed and Width Related Scripts 92

 D.1 Displaying the Link Speed and Link Width of Every PCIe Component 92

 D.2 Display Every PCIe Component with Downgraded Speed or Downgraded Width 92

Appendix E: References 93

 E.1 Broadcom Ethernet Network Adapter User Guide 93

 E.2 ROCm Software installation on Linux 93

Appendix F: Terminology 94

Revision History 95

 957608-AN205; March 20, 2025 95

 957608-AN204; November 5, 2024 95

 957608-AN203; October 28, 2024 95

 957608-AN202; October 22, 2024 95

 957608-AN201; September 23, 2024 95

 957608-AN200; July 31, 2024 95

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 Compute (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 AMD 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 RCCL.

The intended audience of this document is 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 AMD Instinct MI300X 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:
 - ROCm and RCCL for AMD GPUs
- Configure routing on the hosts for RoCE/Peer Mem Direct
- Configure Ethernet switches for RoCE/Peer Mem Direct
- Run [RDMA perftest](#) with ROCm support on AMD GPUs and Broadcom NICs
- Install OpenMPI and UCX with ROCm support
- Run RCCL collectives on AMD GPUs and Broadcom NICs

For more information on AMD ROCm and AMD RCCL, see the following links:

- ROCm: <https://rocm.docs.amd.com/en/latest/rocm.html>
- RCCL: <https://rocm.docs.amd.com/projects/rccl/en/latest/>

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

- Ubuntu 22.04 running 5.15 kernel.

Broadcom's BCM957608 (Thor2) family of NICs supports 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

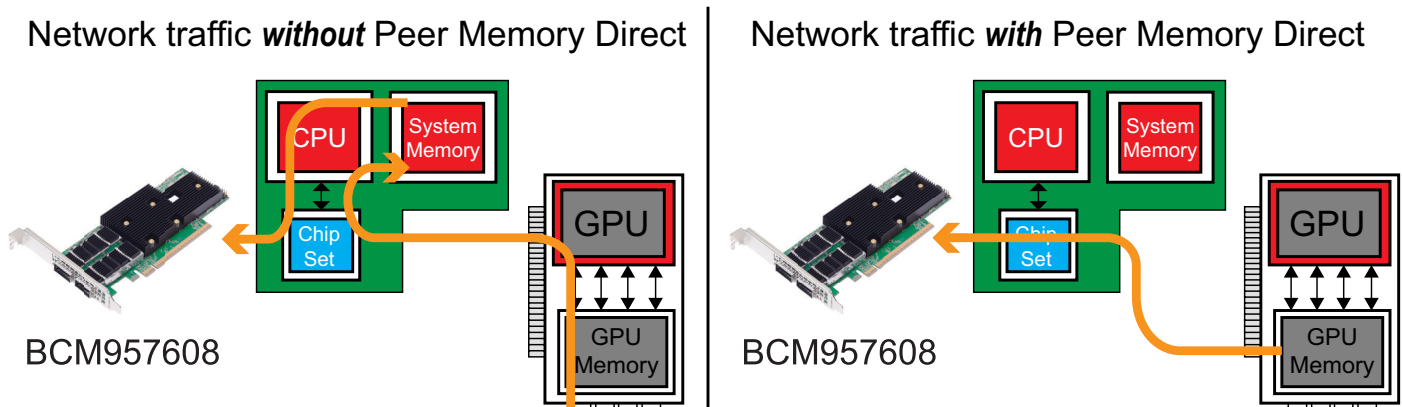
Part Number	Form Factor	Ports	Connector
BCM957608-P1400GD	PCIe	1x 400G	QSFP112-DD
BCM957608-N1400GD	OCP3.0	1x 400G	QSFP112-DD
BCM957608-P2200G	PCIe	2x 200G (default) Can be configured as 1x400G	QSFP112
BCM957608-N2200G	OCP3.0	2x 200G (default) Can be configured as 1x400G	QSFP112

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

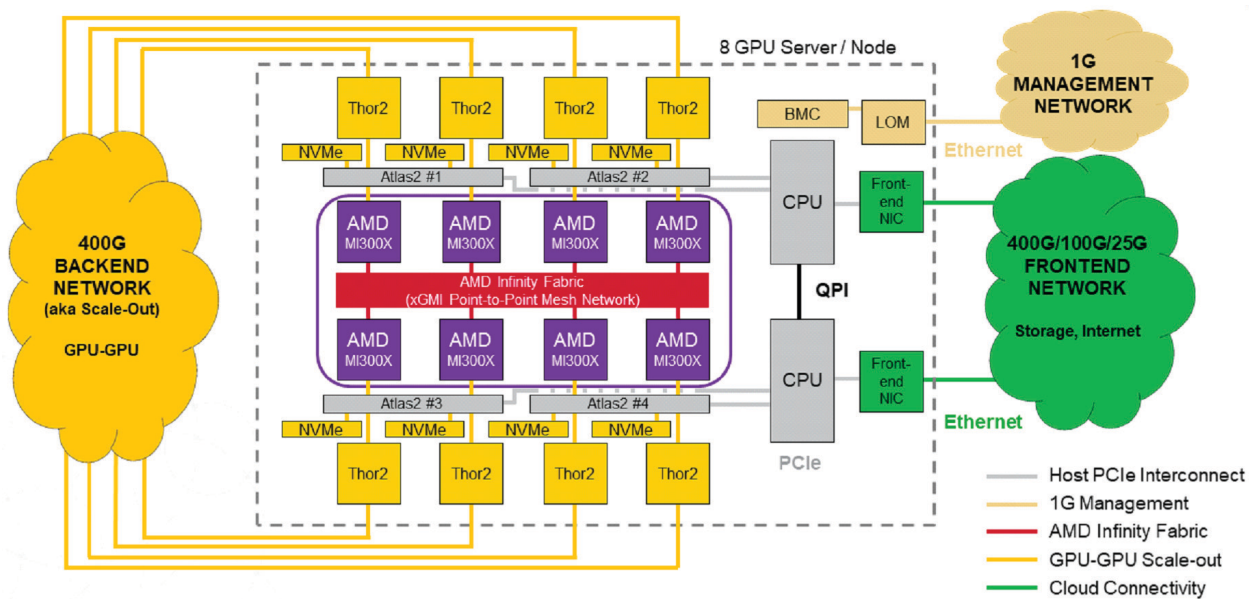
To get the best performance for Peer Memory Direct, the PCIe slot selection for the GPU and the NIC on a host is essential. Peer Memory Direct works via PCIe peer-to-peer transfers where data is directly transferred between the GPU and the NIC over the PCIe bus.

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 (MI300X), 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 AMD Instinct™ MI300 Series Accelerators

This section provides information on configuring the BCM957608 with AMD Instinct MI300 Series Accelerators.

2.1.1 ROCm Installation

Use the instructions provided at: <https://rocm.docs.amd.com/en/latest/> and pick the latest ROCm release.

For the example in this guide, we will be using ROCm v6.1.60100 and the AMD GPU installer method. Instructions are available at: <https://rocm.docs.amd.com/projects/install-on-linux/en/latest/tutorial/quick-start.html>

The installation of the ROCm, as described in this section, installs ROCm, RCCL, and the AMDGPU driver on the host.

2.1.1.1 Example Instructions for Ubuntu 22.04

Enable Video and Render Group

```
sudo usermod -a -G render,video $LOGNAME
```

Install ROCm

```
AMDGPU_DEB=amdgpu-install_6.1.60100-1_all.deb
sudo apt update
wget https://repo.radeon.com/amdgpu-install/6.1/ubuntu/jammy/${AMDGPU_DEB}
sudo apt-get install ./${AMDGPU_DEB}
sudo amdgpu-install --usecase=rocm,hip,dkms
reboot

$ dkms status
amdgpu/6.7.0-1734857.22.04, 5.15.0-101-generic, x86_64: installed

$ dpkg -l | grep rocm-core
ii rocm-core 6.1.0.60100-44~22.04 amd64 Radeon Open Compute (ROCm) Runtime software stack

$ dpkg -l | grep rccl
ii rccl 2.18.6.60100-44~22.04 amd64 ROCm Communication Collectives Library
ii rccl-dev 2.18.6.60100-44~22.04 amd64 ROCm Communication Collectives Library

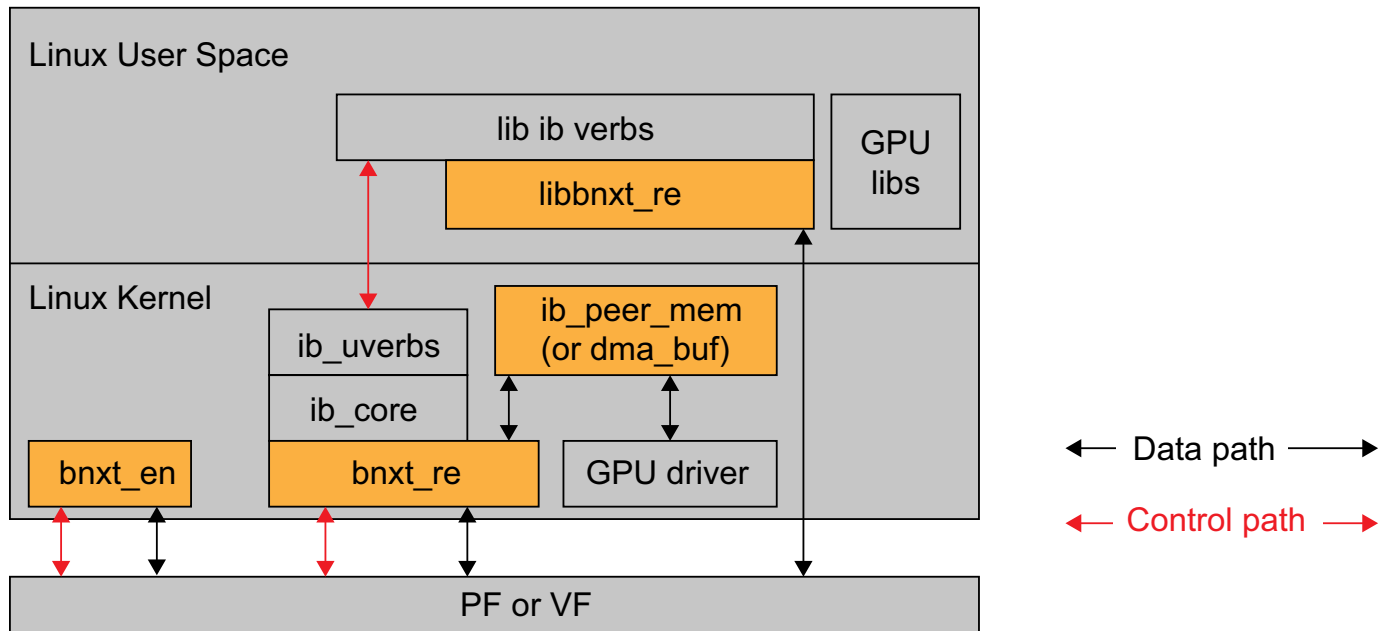
$ rocm-smi
===== ROCm System Management Interface =====
===== Concise Info =====
Device [Model : Revision] Temp Power Power Partitions SCLK MCLK Fan Perf PwrCap VRAM% GPU%
      Name (20 chars) (Junction) (Socket) (Mem, Compute)
=====
0 [0x74a1 : 0x00] 53.0°C 142.0W NPS1, SPX 132Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
1 [0x74a1 : 0x00] 46.0°C 140.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
2 [0x74a1 : 0x00] 48.0°C 147.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
3 [0x74a1 : 0x00] 56.0°C 153.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
4 [0x74a1 : 0x00] 53.0°C 150.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
5 [0x74a1 : 0x00] 45.0°C 136.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
6 [0x74a1 : 0x00] 58.0°C 152.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
7 [0x74a1 : 0x00] 46.0°C 135.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
  AMD Instinct MI300X
=====
===== End of ROCm SMI Log =====
```

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]
  -i Specify interface name or device PCI address.
  -a Specify the IP address to assign to the interface
  -n Specify the IP netmask to assign to the interface
  -s Specify source SIT/CD directory or URL. Default:
/home/brcm/bcm_232.1.132.8c/utils/linux_installer/../../
  -v Increase output verbosity. Passed to ansible-playbook. -vvv for full local debug logs
  -l <prefix> Configure interface IP address as <prefix>. (last octet of public interface)
  -o ECN|PFC|ECNPFC|NOOP RoCE congestion control mode. Default: NOOP
  -r [0-7] RoCE Packet Priority. Default: 3. Must set -o to take effect.
  -d VALUE RoCE Packet DSCP Value. Default: 26. Must set -o to take effect.
  -c [0-7] RoCE CNP Packet Priority. Default: 7. Must set -o to take effect.
  -p VALUE RoCE CNP Packet DSCP Value. Default: 48. Must set -o to take effect.
  -b VALUE RoCE Bandwidth percentage for ETS configuration. Default: 50%. Must set -o to
take effect.
  -m VALUE Interface MTU. Default: 1500
  -2 Do not install/configure RoCE. Install IP driver only
  -q Specify VLAN tag to use instead of DSCP for RoCE
  -w Do not upgrade NIC firmware
  -g Enable peer-mem capable drivers
  -f Force installation, even if it is a downgrade, or if versions already match
  -x Ignore failures of component installation, continue to next component
  -u Source directory is unstructured. Install all supplied components. Implies -f, -x
  -t Run nictune tool to tune the specified interface for optimal throughput
  -e Do not install prerequisite operating system packages
```

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 2.32 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.

2.2.1.1 Example On Using the Broadcom Installer

```
cd $HOME

# Download the tarball

BCM_TARBALL=bcm_232.1.132.8c.tar.gz
# Highlighted item will change depending on the release

tar -xf ${BCM_TARBALL}

cd $HOME/bcm_232.1.132.8c/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 --listdev
# This command will list available Broadcom NICs and index for each

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

# Example:
sudo niccli -dev 1 pkgver
```

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 loads 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=3
```

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

```
sudo niccli.x86_64 --listdev
# This command will list available Broadcom NICs and index for each

sudo niccli.x86_64 -dev <index | pci b:d:f | mac> getqos
# Highlighted item will change depending on the index or PCIE BDF or MAC

# Example:
sudo niccli.x86_64 -dev 1 getqos

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 loads 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_232.1.132.8c.tar.gz
# Highlighted item will change depending on the release

tar xf bcm_232.1.132.8c.tar.gz
# Highlighted item will change depending on the release

cd $HOME/bcm_232.1.132.8c/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-232.0.155.5.tar.gz` contains all the kernel driver source code and Makefiles to build the 232.0.155.5 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-232.0.155.5.tar.gz` contains all the software source code and Makefiles to build the 232.0.155.5 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. Prior to 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_232.1.132.8c.tar.gz
# Highlighted item will change depending on the release

tar xf bcm_232.1.132.8c.tar.gz
# Highlighted item will change depending on the release

cd $HOME/bcm_232.1.132.8c.tar.gz/drivers_linux/bnxt_re/bnxt_re_conf
# Highlighted item will change depending on the release

dpkg -i bnxt_re_conf-232.0.155.5-1_all.deb
# Highlighted item will change depending on the release
or
rpm -Uvh bnxt_re_conf-232.0.155.5-1.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 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 --listdev
# 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> install <FW package>
# 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.

2.2.7.1 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.

2.2.7.2 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.

2.2.7.3 Enable RDMA option on the NIC

To check the option value:

```
sudo niccli -dev <index|pci b:d:f> getoption -name 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 getoption -name support_rdma -scope 0
```

To enable the option:

```
sudo niccli -dev <index|pci b:d:f> setoption -name 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 setoption -name support_rdma -scope 0 -value 1
```

2.2.7.4 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> getoption -name performance_profile
# Highlighted item will change depending on the index or PCIe BDF or MAC

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

To enable the option:

```
sudo niccli -dev <index|pci b:d:f> setoption -name 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 setoption -name performance_profile -value 1
```


2.2.7.5 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> getoption -name pcie_relaxed_ordering  
# Highlighted item will change depending on the index or PCIe BDF or MAC
```

Example:

```
sudo niccli -dev 1 getoption -name pcie_relaxed_ordering
```

To enable the option:

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

Example:

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

2.2.7.6 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> getoption -name 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:

```
sudo niccli -dev 1 getoption -name dcbx_mode -scope 0
```

```
sudo niccli -dev <index | pci b:d:f | mac> getoption -name 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 getoption -name lldp_nearest_bridge -scope 0
```

```
sudo niccli -dev <index | pci b:d:f | mac> getoption -name 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 getoption -name lldp_nearest_non_tpmr_bridge -scope 0
```

To disable the options:

```
sudo niccli -dev <index | pci b:d:f | mac> setoption -name 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 setoption -name dcbx_mode -scope 0 -value 0
```

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

```
$ sudo niccli -dev <index | pci b:d:f | mac> setoption -name 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 getoption -name 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 C, 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 C, 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 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 RCCL 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
```

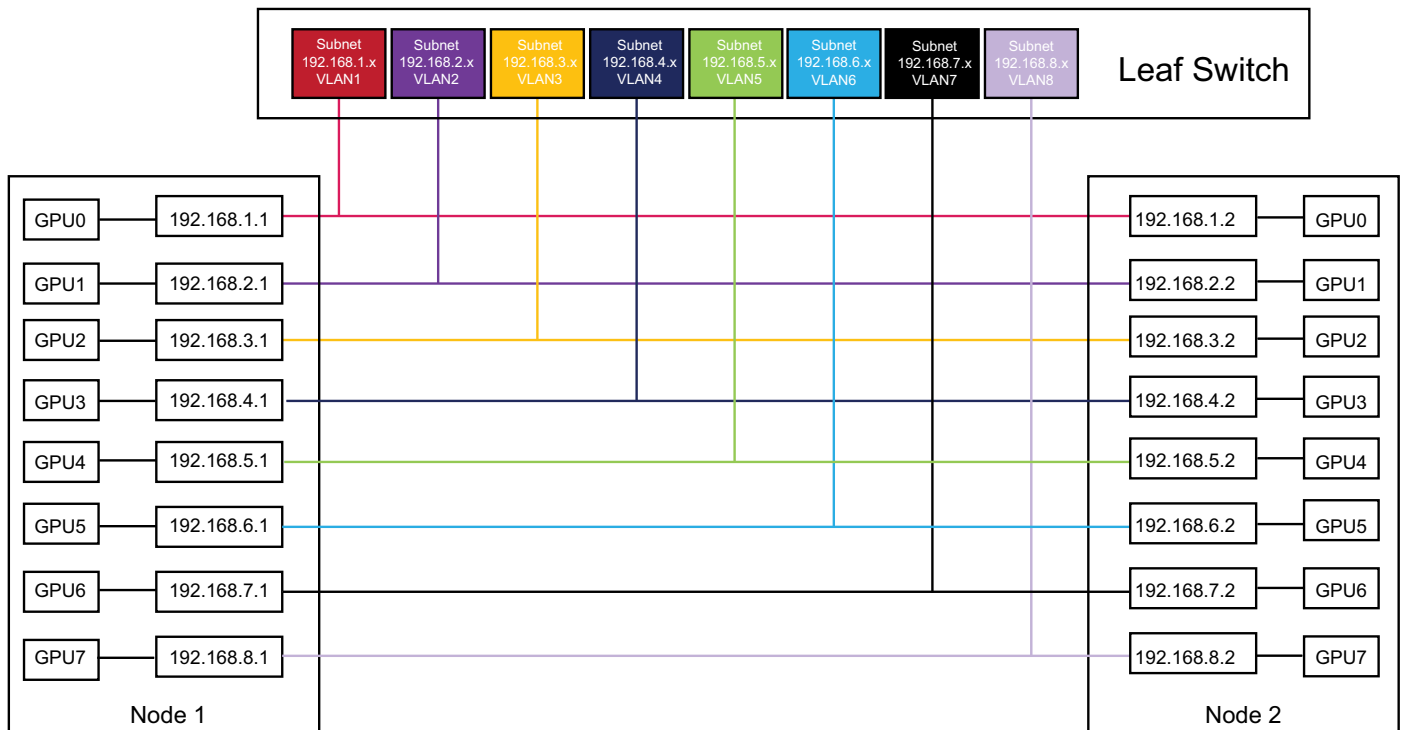
However, a better approach to the ARP flux problem is to configure each NIC on a host in a separate subnet. There are multiple approaches possible here depending on the size of the cluster being configured.

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 used for each NIC where the default gateway of each NIC is the IP address of the switch port VLAN the NIC connects to. 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;
    }
}
```

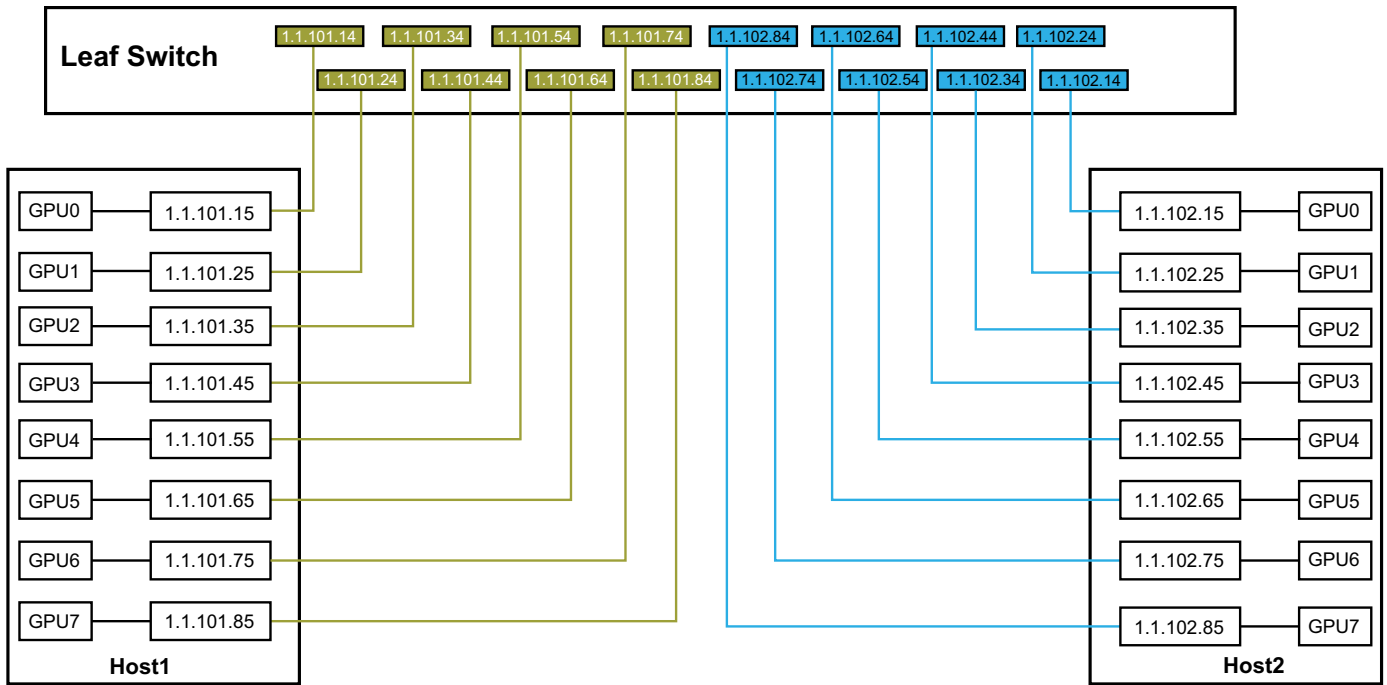
2.4.2 Single Leaf Switch Topology with 31-bit Subnets

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 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. RCCL 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

```
$ sudo niccli -dev 1 get_qos
```

```
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 -dev 1 dump pri2cos
```

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

```
-----
Priority  TC  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 -dev 1 get_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
!

```

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 sm1;
        unit * {
            classifiers {
                dscp mydscp;
            }
        }
    }
}
scheduler-maps {
    sm1 {
        forwarding-class CNP scheduler s2-cnp;
        forwarding-class NO-LOSS scheduler s1;
    }
}
schedulers {
    s1 {
        drop-profile-map loss-priority any protocol any drop-profile dp1;
        explicit-congestion-notification;
    }
    s2-cnp {
        transmit-rate percent 5;
        priority strict-high;
    }
}
}
```

```
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 dotlp-tc ROCE
  dotlp 0-2,5-7 traffic-class 0
  dotlp 3 traffic-class 3
  dotlp 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 dotlp-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 nodel_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
!

```

2.6 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 getqos` command:

```
niccli <index | pci b:d:f | mac> getqos
```

3. The AMD GPU driver `amdgpu.ko` is loaded.
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 RCCL testing, NUMA balancing is disabled on each host participating in the RCCL 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 AMD GPU Support

NOTE: The following instructions are intended for recompiled perfctest utility that can be run directly from the home directory. The instructions assume ROCm is already installed. See [ROCm Installation](#) for details.

```
sudo apt install libibumad-dev
sudo apt install pciutils
sudo apt install libpci*
sudo apt install automake autoconf libtool libibverbs-dev ibverbs-utils infiniband-diags ethtool
librdmacm-dev
```

```
cd $HOME
git clone https://github.com/linux-rdma/perftest.git
cd perftest
./autogen.sh
./configure --prefix=`pwd` --enable-rocm --with-rocm=/opt/rocm
make
```

```
./ib_write_bw -h | grep -i rocm
--use_rocm=<rocm device id> Use selected ROCm device for GPUDirect RDMA testing
```


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 AMD GPU

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

2.8.2 Example – ib_write_bw Test Using Broadcom NIC with AMD GPU

Following is an example of running `ib_write_bw` test using Broadcom NIC with AMDGPU across Host1, NIC0, and Host2, NIC4. For the best `peer_memory` direct performance, the GPUs used for this test should be the ones closest to the NIC on the PCIe Bus.

```
# Host1,NIC0

$ rdma link show | grep -i bnxt_re0
link bnxt_re0/1 state ACTIVE physical_state LINK_UP netdev enp29s0np0

$ ethtool -i enp29s0np0 | grep -i bus
bus-info: 0000:1d:00.0

$ ip a | grep -i enp29s0np0 | grep -i "inet "
inet 1.1.101.15/31 scope global enp29s0np0

$ rocm-smi --showbus
GPU[0]      : PCI Bus: 0000:1C:00.0
GPU[1]      : PCI Bus: 0000:42:00.0
GPU[2]      : PCI Bus: 0000:55:00.0
GPU[3]      : PCI Bus: 0000:68:00.0
GPU[4]      : PCI Bus: 0000:9E:00.0
GPU[5]      : PCI Bus: 0000:C2:00.0
GPU[6]      : PCI Bus: 0000:D4:00.0
GPU[7]      : PCI Bus: 0000:E6:00.0

# Host2,NIC4

$ rdma link show | grep -i bnxt_re4
link bnxt_re4/1 state ACTIVE physical_state LINK_UP netdev enp159s0np0

$ ethtool -i enp159s0np0 | grep -i bus
bus-info: 0000:9f:00.0

$ ip a | grep -i enp159s0np0 | grep -i "inet "
inet 1.1.102.45/31 scope global enp159s0np0

$ rocm-smi --showbus
GPU[0]      : PCI Bus: 0000:1C:00.0
GPU[1]      : PCI Bus: 0000:42:00.0
GPU[2]      : PCI Bus: 0000:55:00.0
GPU[3]      : PCI Bus: 0000:68:00.0
GPU[4]      : PCI Bus: 0000:9E:00.0
```

```
GPU[5]      : PCI Bus: 0000:C2:00.0
GPU[6]      : PCI Bus: 0000:D4:00.0
GPU[7]      : PCI Bus: 0000:E6:00.0
```

From the NIC PCIe B:D:F and the GPU PCIe B:D:F, we can infer that the GPU0 is nearest to NIC0 on host 1 and GPU4 is nearest to NIC4 on host 2.

```
# Start Server on Host1
```

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

```
# Start Client on Host2
```

```
$ $HOME/perftest/bin/ib_write_bw -d bnxt_re4 --use_rocm=4 -a -F -x 3 -q 4 --report_gbits -b --
bind_source_ip 1.1.102.45 1.1.101.15
```

```
Using ROCm Device with ID: 4, Name: AMD Instinct MI300X, PCI Bus ID: 0x9e, GCN Arch:
gfx942:sramecc+:xnack-
allocated 67108864 bytes of GPU buffer at 0x7f3e83e00000
```

```
-----
RDMA_Write Bidirectional BW Test
Dual-port      : OFF          Device      : bnxt_re4
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
Use ROCm memory : ON
Data ex. method : Ethernet
-----
local address: LID 0000 QPN 0x2c06 PSN 0x914f93 RKey 0x2002a13 VAddr 0x007f3e85e00000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:03
local address: LID 0000 QPN 0x2c66 PSN 0xa85d81 RKey 0x2002a13 VAddr 0x007f3e86600000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:03
local address: LID 0000 QPN 0x2c15 PSN 0xf3737 RKey 0x2002a13 VAddr 0x007f3e86e00000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:03
local address: LID 0000 QPN 0x2c33 PSN 0xf08bba RKey 0x2002a13 VAddr 0x007f3e87600000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:03
remote address: LID 0000 QPN 0x2c58 PSN 0xe38cfd RKey 0x200e115 VAddr 0x007f0862e2a000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:01
remote address: LID 0000 QPN 0x2ca8 PSN 0x91bfa3 RKey 0x200e115 VAddr 0x007f086362a000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:01
remote address: LID 0000 QPN 0x2c66 PSN 0xf08bb1 RKey 0x200e115 VAddr 0x007f0863e2a000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:01
remote address: LID 0000 QPN 0x2c43 PSN 0x3f1fac RKey 0x200e115 VAddr 0x007f086462a000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:01
-----
#bytes      #iterations      BW peak[Gb/sec]      BW average[Gb/sec]      MsgRate[Mpps]
2           20000             0.119718             0.108526                 6.782848
4           20000             0.215444             0.163003                 5.093837
8           20000             0.42                 0.42                     6.491752
```

16	20000	0.85	0.84	6.551117
32	20000	1.63	1.61	6.306775
64	20000	3.33	3.33	6.497098
128	20000	8.23	8.22	8.030138
256	20000	16.62	16.61	8.110011
512	20000	32.98	32.91	8.034420
1024	20000	66.29	66.19	8.080304
2048	20000	133.10	132.91	8.112182
4096	20000	246.94	198.41	6.055130
8192	20000	473.26	463.26	7.068743
16384	20000	701.70	593.80	4.530332
32768	20000	748.95	677.73	2.585332
65536	20000	757.18	444.06	0.846972
131072	20000	771.17	732.12	0.698202
262144	20000	773.98	763.28	0.363961
524288	20000	774.33	770.98	0.183815
1048576	20000	772.44	765.68	0.091276
2097152	20000	774.92	774.92	0.046189
4194304	20000	772.83	772.64	0.023026
8388608	20000	775.17	774.99	0.011548

deallocating GPU buffer 0x7f3e83e00000

NOTE: The GPU buffers allocated prior to start of the test and deallocated after test completion. This confirms that perftest 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

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 D, 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. How to check the ROCm and RCCL versions installed on the host.

On Ubuntu-based hosts, use the command `dpkg -l | grep -i rocm-core`. On RHEL based hosts, use the `rpm -qa | grep -i rocm-core` command. When ROCm is installed on the host, RCCL is also installed by the same pkg. The ROCm and RCCL versions usually match, unless RCCL is downloaded and installed separately, for example from GitHub. Passing the value **version** to the `NCCL_DEBUG` environment variable during `rccl-tests` prints the rccl version in use. See [MultiNode RCCL Collectives Using Open MPI](#).

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

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

```
$ sudo niccli -dev <index | pci b:d:f | mac> get_qos
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 inbox `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.x86_64 -dev <index | pci b:d:f | mac> getoption -name enable_sriov
# The highlighted item will change depending on the index or PCIE BDF or MAC
```

```
# Example:
sudo niccli.x86_64 -dev 1 getoption -name enable_sriov
```

```
sudo niccli.x86_64 -dev <index | pci b:d:f | mac> getoption -name 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.x86_64 -dev 1 getoption -name default_evb_mode -scope 0
```

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


```
sudo niccli.x86_64 -dev <index | pci b:d:f | mac> setoption -name enable_sriov -value 0
# The highlighted item will change depending on the index or PCIE BDF or MAC
```

```
# Example:
```

```
$ sudo niccli.x86_64 -dev 1 setoption -name enable_sriov -value 0
```

```
$ sudo niccli.x86_64 -dev <index | pci b:d:f | mac> setoption -name 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.x86_64 -dev 1 setoption -name 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
```

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 AMD GPU Drivers

The installing of the ROCm software stack as described in section ROCm Installation installs the AMD GPU driver `amdgpu` as well. However, there may be changes across ROCm releases, therefore, see the ROCm official documentation at <https://rocm.docs.amd.com/projects/install-on-linux/en/latest/>.

Chapter 7: Debugging AMD Instinct MI300 Series Accelerators

If the `rocm-smi` command does not display any GPU in the system, it indicates that the AMD GPU driver `amdgpu` is not loaded. Use the following commands to check the status of the `amdgpu` driver:

1. `smode | grep -i amdgpu`
2. `modinfo amdgpu`
3. `dkms status`
4. `dmesg | grep -i amdgpu`

See the following AMD instructions to collect debug info to troubleshoot problems related to the GPU: <https://github.com/amddcgpuce/rocmtechsupport>

Chapter 8: Running RCCL Collectives

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

To run RCCL across multiple nodes in a cluster, Open MPI installation is required. Open MPI is only used to launch the RCCL tests/processes across multiple nodes. Additionally, UCX can be used along with Open MPI to launch the RCCL tests/processes across multiple nodes. RCCL itself does not use openMPI or UCX for its operation. Broadcom uses OpenMPI version 4.1.6 and UCX version 1.15 for RCCL testing. Other later versions of OpenMPI and UCX should work as well, but Broadcom testing has been done using OpenMPI version 4.1.6 and UCX version 1.15. It is recommended to install and execute Open MPI, UCX, and rccl-tests as a non-root user. Passwordless SSH should be set up for the non-root user as well.

8.1 Setting up the Environmental Variable

To set up environment variables, use the following commands:

```
export UCX_VER=v1.15.0
export OMPI_VER=v4.1.6
```

8.2 Installing UCX for AMD GPUs

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

```
# The UCX build needs to point to the ROCm installation /opt/rocm
# as shown in the steps below.

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
../contrib/configure-release --disable-debug --disable-assertions --disable-params-check --with-rocm=/opt/rocm --with-rc --with-ud --with-dc --with-dm --with-ib-hw-tm --prefix=$HOME/ucx/ucx_install --disable-log

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

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

8.3 Installing Open MPI for AMD GPUs

To install Open MPI for AMD 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/mpi.git
cd mpi

./autogen.pl
mkdir build
mkdir ompi_install

cd build

../configure --prefix=$HOME/mpi/mpi_install --with-ucx=$HOME/ucx/ucx_install --enable-mca-no-
build=bt1-uct

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

# Verify by running
$HOME/mpi/mpi_install/bin/mpi_info | grep Configure
```

Open MPI BTL openib requires Broadcom NIC PCIe vendor_part_id 0x1760 to be added to the file \$HOME/mpi/mpi_install/share/openmpi/mca-btl-openib-device-params.ini, under:

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

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

8.4 Compiling RCCL Tests

NOTE: The location of the RCCL library (`/opt/rocm/lib/`) may change with ROCm versions.

```
# On Ubuntu install libstdc++-12-dev
sudo apt install libstdc++-12-dev
cd $HOME
git clone https://github.com/ROCmSoftwarePlatform/rccl-tests.git
cd rccl-tests/

MPI=1 MPI_HOME=$HOME/mpi/mpi_install/ RCCL_HOME=/opt/rocm/lib make -j $(nproc)
```

8.5 Single Node RCCL Collectives

This section provides information single-node RCCL collectives.

8.5.1 Topology and Sample Test Results

This section provides the topology used and sample test results.

```
$ rocm-smi --showtopotype
```

```
===== ROCm System Management Interface =====
===== Link Type between two GPUs =====
      GPU0      GPU1      GPU2      GPU3      GPU4      GPU5      GPU6      GPU7
GPU0    0      XGMI      XGMI      XGMI      XGMI      XGMI      XGMI      XGMI
GPU1  XGMI      0      XGMI      XGMI      XGMI      XGMI      XGMI      XGMI
GPU2  XGMI      XGMI      0      XGMI      XGMI      XGMI      XGMI      XGMI
GPU3  XGMI      XGMI      XGMI      0      XGMI      XGMI      XGMI      XGMI
GPU4  XGMI      XGMI      XGMI      XGMI      0      XGMI      XGMI      XGMI
GPU5  XGMI      XGMI      XGMI      XGMI      XGMI      0      XGMI      XGMI
GPU6  XGMI      XGMI      XGMI      XGMI      XGMI      XGMI      0      XGMI
GPU7  XGMI      XGMI      XGMI      XGMI      XGMI      XGMI      XGMI      0
===== End of ROCm SMI Log =====
```

Single Node RCCL collectives do not use RoCE by default and can be exercised with the Broadcom `bnxt_en` and `bnxt_re` drivers removed as well. Single Node tests make use of the XGMI links for inter-GPU communication on the host.

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

```
export LD_LIBRARY_PATH=$HOME/ompi/ompi_install/lib:$HOME/ucx/ucx_install/lib:$LD_LIBRARY_PATH
```

```
$ $HOME/rccl-tests/build/all_reduce_perf -b 8 -e 16g -f 2 -g 8
```

```
-----
WARNING: There was an error initializing an OpenFabrics device.
```

```
Local host:  brcm-cos-1
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
#
rccl-tests: Version develop:990f88c
# Using devices
# Rank 0 Pid 1630024 on brcm-cos-1 device 0 [0000:1c:00.0] AMD Instinct MI300X
# Rank 1 Pid 1630024 on brcm-cos-1 device 1 [0000:42:00.0] AMD Instinct MI300X
# Rank 2 Pid 1630024 on brcm-cos-1 device 2 [0000:55:00.0] AMD Instinct MI300X
# Rank 3 Pid 1630024 on brcm-cos-1 device 3 [0000:68:00.0] AMD Instinct MI300X
# Rank 4 Pid 1630024 on brcm-cos-1 device 4 [0000:9e:00.0] AMD Instinct MI300X
# Rank 5 Pid 1630024 on brcm-cos-1 device 5 [0000:c2:00.0] AMD Instinct MI300X
# Rank 6 Pid 1630024 on brcm-cos-1 device 6 [0000:d4:00.0] AMD Instinct MI300X
# Rank 7 Pid 1630024 on brcm-cos-1 device 7 [0000:e6:00.0] AMD Instinct MI300X
#
#
#          size          count      type  redop  root    time      time      time      time      time      time      time
#          (B)      (elements)          type          root    (us)      (GB/s)  (GB/s)  #wrong    (us)      (GB/s)  (GB/s)  #wrong
#          8          2      float    sum    -1    34.34    0.00    0.00    0    33.76    0.00    0.00    0
#          16         4      float    sum    -1    33.71    0.00    0.00    0    33.45    0.00    0.00    0
#          32         8      float    sum    -1    32.95    0.00    0.00    0    33.89    0.00    0.00    0
#          64        16      float    sum    -1    33.75    0.00    0.00    0    34.75    0.00    0.00    0
#          128       32      float    sum    -1    33.24    0.00    0.01    0    33.31    0.00    0.01    0
#          256       64      float    sum    -1    33.78    0.01    0.01    0    33.54    0.01    0.01    0
#          512      128      float    sum    -1    33.55    0.02    0.03    0    34.59    0.01    0.03    0
#          1024     256      float    sum    -1    35.96    0.03    0.05    0    34.06    0.03    0.05    0
#          2048     512      float    sum    -1    34.92    0.06    0.10    0    33.28    0.06    0.11    0
#          4096    1024      float    sum    -1    35.48    0.12    0.20    0    33.63    0.12    0.21    0
#          8192    2048      float    sum    -1    34.41    0.24    0.42    0    34.19    0.24    0.42    0
#          16384   4096      float    sum    -1    35.61    0.46    0.81    0    34.77    0.47    0.82    0
```

```

32768      8192      float      sum      -1      36.27      0.90      1.58      0      36.20      0.91      1.58      0
65536      16384     float      sum      -1      36.04      1.82      3.18      0      35.82      1.83      3.20      0
131072     32768     float      sum      -1      39.76      3.30      5.77      0      36.79      3.56      6.23      0
262144     65536     float      sum      -1      41.12      6.38      11.16     0      41.02      6.39      11.18     0
524288     131072    float      sum      -1      48.45      10.82     18.94     0      47.95      10.93     19.13     0
1048576    262144    float      sum      -1      61.50      17.05     29.84     0      61.35      17.09     29.91     0
2097152    524288    float      sum      -1      62.25      33.69     58.95     0      62.33      33.65     58.88     0
4194304    1048576   float      sum      -1      84.09      49.88     87.29     0      87.42      47.98     83.96     0
8388608    2097152   float      sum      -1      104.5      80.30     140.52    0      107.4      78.13     136.72    0
16777216   4194304   float      sum      -1      161.8      103.67    181.43    0      171.2      97.99     171.48    0
33554432   8388608   float      sum      -1      242.8      138.21    241.87    0      256.5      130.79    228.89    0
67108864   16777216  float      sum      -1      421.5      159.21    278.62    0      431.6      155.49    272.10    0
134217728  33554432  float      sum      -1      779.2      172.26    301.45    0      788.0      170.33    298.08    0
268435456  67108864  float      sum      -1      1989.7     134.91    236.10    0      1514.2     177.28    310.24    0
536870912  134217728 float      sum      -1      3403.1     157.76    276.08    0      2957.8     181.51    317.64    0
1073741824 268435456 float      sum      -1      5808.6     184.85    323.49    0      5817.4     184.57    323.00    0
2147483648 536870912 float      sum      -1      11975      179.33    313.84    0      12357      173.79    304.13    0
4294967296 1073741824 float      sum      -1      23577      182.17    318.79    0      23740      180.92    316.61    0
8589934592 2147483648 float      sum      -1      47380      181.30    317.27    0      46880      183.23    320.66    0
17179869184 4294967296 float      sum      -1      94981      180.88    316.53    0      94503      181.79    318.14    0
# Errors with asterisks indicate errors that have exceeded the maximum threshold.
# Out of bounds values : 0 OK
# Avg bus bandwidth   : 109.341

```

8.6 Testing Single Node RCCL Collectives Using NICs

It is possible to use the NICs with RCCL tests on a single node using the RCCL environment variable `RCCL_ENABLE_INTRANET=1`. With this setting, RCCL uses the NICs together with other GPU interconnects (XGMI or PCIe) when running on a single node. The advantage of using this environment variable is that enables NIC testing on a single node itself.

8.7 MultiNode RCCL Collectives Using Open MPI

This section provides information MultiNode RCCL Collectives using Open MPI.

8.7.1 Prechecks

This section provides information on the required prechecks:

- NIC IPs and their 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 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=$HOME/ompi/ompi_install/lib:$HOME/ucx/ucx_install/lib:$LD_LIBRARY_PATH
export PATH=$HOME/ompi/ompi_install/bin:$HOME/ucx/ucx_install/bin:$PATH
```

8.7.2 Topology and Sample Test Results

This section provides the topology used and sample test results.

```
$ rocm-smi --showtopotype
```

```
===== ROCm System Management Interface =====
===== Link Type between two GPUs =====
      GPU0      GPU1      GPU2      GPU3      GPU4      GPU5      GPU6      GPU7
GPU0  0         XGMI     XGMI     XGMI     XGMI     XGMI     XGMI     XGMI
GPU1  XGMI      0        XGMI     XGMI     XGMI     XGMI     XGMI     XGMI
GPU2  XGMI      XGMI     0        XGMI     XGMI     XGMI     XGMI     XGMI
GPU3  XGMI      XGMI     XGMI     0        XGMI     XGMI     XGMI     XGMI
GPU4  XGMI      XGMI     XGMI     XGMI     0        XGMI     XGMI     XGMI
GPU5  XGMI      XGMI     XGMI     XGMI     XGMI     0        XGMI     XGMI
GPU6  XGMI      XGMI     XGMI     XGMI     XGMI     XGMI     0        XGMI
GPU7  XGMI      XGMI     XGMI     XGMI     XGMI     XGMI     XGMI     0
===== End of ROCm SMI Log =====
```

```
$ rocm-smi --showbus
```

```
===== ROCm System Management Interface =====
===== PCI Bus ID =====
GPU[0]      : PCI Bus: 0000:1B:00.0
GPU[1]      : PCI Bus: 0000:3D:00.0
GPU[2]      : PCI Bus: 0000:4E:00.0
GPU[3]      : PCI Bus: 0000:5F:00.0
GPU[4]      : PCI Bus: 0000:9D:00.0
GPU[5]      : PCI Bus: 0000:BD:00.0
GPU[6]      : PCI Bus: 0000:CD:00.0
GPU[7]      : PCI Bus: 0000:DD:00.0
=====
===== End of ROCm SMI Log =====
```



```
$ rdma link show
link bnxt_re0/1 state ACTIVE physical_state LINK_UP netdev enp28s0np0
link bnxt_re1/1 state ACTIVE physical_state LINK_UP netdev enp62s0np0
link bnxt_re2/1 state ACTIVE physical_state LINK_UP netdev enp79s0np0
link bnxt_re3/1 state ACTIVE physical_state LINK_UP netdev enp96s0np0
link bnxt_re4/1 state ACTIVE physical_state LINK_UP netdev enp158s0np0
link bnxt_re5/1 state ACTIVE physical_state LINK_UP netdev enp190s0np0
link bnxt_re6/1 state ACTIVE physical_state LINK_UP netdev enp206s0np0
link bnxt_re7/1 state ACTIVE physical_state LINK_UP netdev enp222s0np0
```

In the RCCL tests below, four nodes are used to run the `all_reduce` and the `alltoall` collective. Each node has eight GPUs and eight NICs. All four Hosts connect to the same leaf switch.

8.7.2.1 Test: All-to-All

Both commands execute the same collective (all-to-all):

- The first command runs 32 processes (one process on each GPU on each of the four nodes).
- The second command runs 4 processes (one process on eight GPUs on each of the four nodes).

NOTE: Multi-line commands are shown in the following sections.

8.7.2.1.1 Test: All-to-All (4 Nodes, 32 Processes)

```
/opt/AMD/install/mpi/bin/mpirun --allow-run-as-root --bind-to none --hostfile hostfile
-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_IB_DISABLE=0 \
-x NCCL_NET_GDR_LEVEL=SYS \
-x NCCL_NET_GDR_READ=1 \
-x NCCL_P2P_LEVEL=SYS \
-x NCCL_SHM_DISABLE=1 \
-x NCCL_IB_PCI_RELAXED_ORDERING=1 \
-x HSA_FORCE_FINE_GRAIN_PCIE=1 \
-x NCCL_DMABUF_ENABLE=0 \
-x LD_LIBRARY_PATH=/opt/AMD/install/mpi/lib/ \
-x NCCL_MIN_NCHANNELS=32 --mca pml ucx --mca osc ucx --mca spml ucx --mca btl ^vader,tcp,openib,uct \
/home/test/rccl-tests/build/alltoall_perf -b 8 -e 16G -f 2 -g 1 -c 0

# nThread 1 nGpus 1 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 0 graph: 0
#
rccl-tests: Version develop:77ae744
# Using devices
# Rank 0 Pid 12833 on irvine-waco1 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 1 Pid 12834 on irvine-waco1 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 2 Pid 12835 on irvine-waco1 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 3 Pid 12836 on irvine-waco1 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 4 Pid 12837 on irvine-waco1 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 5 Pid 12838 on irvine-waco1 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 6 Pid 12839 on irvine-waco1 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 7 Pid 12840 on irvine-waco1 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 8 Pid 12882 on irvine-waco2 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 9 Pid 12883 on irvine-waco2 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 10 Pid 12884 on irvine-waco2 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 11 Pid 12885 on irvine-waco2 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 12 Pid 12886 on irvine-waco2 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 13 Pid 12887 on irvine-waco2 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 14 Pid 12888 on irvine-waco2 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 15 Pid 12889 on irvine-waco2 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 16 Pid 13852 on irvine-waco3 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 17 Pid 13853 on irvine-waco3 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 18 Pid 13854 on irvine-waco3 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 19 Pid 13856 on irvine-waco3 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 20 Pid 13855 on irvine-waco3 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 21 Pid 13857 on irvine-waco3 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 22 Pid 13858 on irvine-waco3 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 23 Pid 13859 on irvine-waco3 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 24 Pid 20255 on irvine-waco4 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 25 Pid 20256 on irvine-waco4 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 26 Pid 20258 on irvine-waco4 device 2 [0000:4e:00.0] AMD Instinct MI300X
```

```
# Rank 27 Pid 20259 on irvine-waco4 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 28 Pid 20257 on irvine-waco4 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 29 Pid 20260 on irvine-waco4 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 30 Pid 20261 on irvine-waco4 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 31 Pid 20262 on irvine-waco4 device 7 [0000:dd:00.0] AMD Instinct MI300X
#
#
# 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)
# 0 0 float none -1 0.16 0.00 0.00 N/A 0.05 0.00 0.00 N/A
# 0 0 float none -1 0.05 0.00 0.00 N/A 0.05 0.00 0.00 N/A
# 0 0 float none -1 0.05 0.00 0.00 N/A 0.05 0.00 0.00 N/A
# 0 0 float none -1 0.05 0.00 0.00 N/A 0.05 0.00 0.00 N/A
# 128 1 float none -1 97.07 0.00 0.00 N/A 92.84 0.00 0.00 N/A
# 256 2 float none -1 92.84 0.00 0.00 N/A 91.35 0.00 0.00 N/A
# 512 4 float none -1 91.20 0.01 0.01 N/A 91.69 0.01 0.01 N/A
# 1024 8 float none -1 91.12 0.01 0.01 N/A 91.46 0.01 0.01 N/A
# 2048 16 float none -1 92.21 0.02 0.02 N/A 92.25 0.02 0.02 N/A
# 4096 32 float none -1 91.06 0.04 0.04 N/A 91.43 0.04 0.04 N/A
# 8192 64 float none -1 91.96 0.09 0.09 N/A 90.82 0.09 0.09 N/A
# 16384 128 float none -1 91.51 0.18 0.17 N/A 91.28 0.18 0.17 N/A
# 32768 256 float none -1 94.69 0.35 0.34 N/A 91.25 0.36 0.35 N/A
# 65536 512 float none -1 93.75 0.70 0.68 N/A 92.41 0.71 0.69 N/A
# 131072 1024 float none -1 96.64 1.36 1.31 N/A 96.78 1.35 1.31 N/A
# 262144 2048 float none -1 97.30 2.69 2.61 N/A 96.76 2.71 2.62 N/A
# 524288 4096 float none -1 104.5 5.02 4.86 N/A 104.2 5.03 4.87 N/A
# 1048576 8192 float none -1 110.0 9.54 9.24 N/A 110.1 9.53 9.23 N/A
# 2097152 16384 float none -1 139.8 15.00 14.53 N/A 139.9 14.99 14.52 N/A
# 4194304 32768 float none -1 242.6 17.29 16.75 N/A 187.8 22.33 21.64 N/A
# 8388608 65536 float none -1 234.4 35.79 34.68 N/A 231.1 36.30 35.16 N/A
# 16777216 131072 float none -1 465.2 36.07 34.94 N/A 582.2 28.82 27.92 N/A
# 33554432 262144 float none -1 1123.6 29.86 28.93 N/A 938.9 35.74 34.62 N/A
# 67108864 524288 float none -1 1571.7 42.70 41.36 N/A 1440.3 46.59 45.14 N/A
# 134217728 1048576 float none -1 4145.6 32.38 31.36 N/A 3985.6 33.68 32.62 N/A
# 268435456 2097152 float none -1 8727.9 30.76 29.79 N/A 8604.3 31.20 30.22 N/A
# 536870912 4194304 float none -1 14931 35.96 34.83 N/A 15078 35.61 34.49 N/A
# 1073741824 8388608 float none -1 27590 38.92 37.70 N/A 29025 36.99 35.84 N/A
# 2147483648 16777216 float none -1 49630 43.27 41.92 N/A 49465 43.41 42.06 N/A
# 4294967296 33554432 float none -1 90275 47.58 46.09 N/A 89518 47.98 46.48 N/A
# 8589934592 67108864 float none -1 172027 49.93 48.37 N/A 170271 50.45 48.87 N/A
# 17179869184 134217728 float none -1 333586 51.50 49.89 N/A 332577 51.66 50.04 N/A
# Out of bounds values : 0 OK
# Avg bus bandwidth : 16.0871
#
$ cat hostfile
host1 slots=8
host2 slots=8
host3 slots=8
host4 slots=8
```

8.7.2.1.2 Test: All-to-All (4 Nodes, 4 Processes)

```
/opt/AMD/install/mpi/bin/mpirun --allow-run-as-root --bind-to none --hostfile hostfile \
-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_IB_DISABLE=0 \
-x NCCL_NET_GDR_LEVEL=SYS \
-x NCCL_NET_GDR_READ=1 \
-x NCCL_P2P_LEVEL=SYS \
-x NCCL_SHM_DISABLE=1 \
-x NCCL_IB_PCI_RELAXED_ORDERING=1 \
-x HSA_FORCE_FINE_GRAIN_PCIE=1 \
-x NCCL_DMABUF_ENABLE=0 -x LD_LIBRARY_PATH=/opt/AMD/install/mpi/lib/ -x NCCL_MIN_NCHANNELS=32 --mca pml ucx --mca osc ucx --mca
spml ucx --mca btl ^vader,tcp,openib,uct /home/test/rccl-tests/build/alltoall_perf -b 8 -e 16G -f 2 -g 8 -c 0
# nThread 1 nGpus 8 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 0 graph: 0
#
rccl-tests: Version develop:77ae744
# Using devices
# Rank 0 Pid 11838 on irvine-waco1 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 1 Pid 11838 on irvine-waco1 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 2 Pid 11838 on irvine-waco1 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 3 Pid 11838 on irvine-waco1 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 4 Pid 11838 on irvine-waco1 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 5 Pid 11838 on irvine-waco1 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 6 Pid 11838 on irvine-waco1 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 7 Pid 11838 on irvine-waco1 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 8 Pid 11927 on irvine-waco2 device 0 [0000:1b:00.0] AMD Instinct MI300X
```

```
# Rank 9 Pid 11927 on irvine-waco2 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 10 Pid 11927 on irvine-waco2 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 11 Pid 11927 on irvine-waco2 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 12 Pid 11927 on irvine-waco2 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 13 Pid 11927 on irvine-waco2 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 14 Pid 11927 on irvine-waco2 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 15 Pid 11927 on irvine-waco2 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 16 Pid 12703 on irvine-waco3 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 17 Pid 12703 on irvine-waco3 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 18 Pid 12703 on irvine-waco3 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 19 Pid 12703 on irvine-waco3 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 20 Pid 12703 on irvine-waco3 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 21 Pid 12703 on irvine-waco3 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 22 Pid 12703 on irvine-waco3 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 23 Pid 12703 on irvine-waco3 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 24 Pid 19105 on irvine-waco4 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 25 Pid 19105 on irvine-waco4 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 26 Pid 19105 on irvine-waco4 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 27 Pid 19105 on irvine-waco4 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 28 Pid 19105 on irvine-waco4 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 29 Pid 19105 on irvine-waco4 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 30 Pid 19105 on irvine-waco4 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 31 Pid 19105 on irvine-waco4 device 7 [0000:dd:00.0] AMD Instinct MI300X
```

#	size (B)	count (elements)	type	redop	root	out-of-place				in-place			
						time (us)	algbw (GB/s)	busbw (GB/s)	#wrong	time (us)	algbw (GB/s)	busbw (GB/s)	#wrong
#	0	0	float	none	-1	0.35	0.00	0.00	N/A	0.31	0.00	0.00	N/A
#	0	0	float	none	-1	0.32	0.00	0.00	N/A	0.31	0.00	0.00	N/A
#	0	0	float	none	-1	0.31	0.00	0.00	N/A	0.31	0.00	0.00	N/A
#	0	0	float	none	-1	0.31	0.00	0.00	N/A	0.32	0.00	0.00	N/A
#	128	1	float	none	-1	190.1	0.00	0.00	N/A	191.2	0.00	0.00	N/A
#	256	2	float	none	-1	189.6	0.00	0.00	N/A	187.4	0.00	0.00	N/A
#	512	4	float	none	-1	187.0	0.00	0.00	N/A	187.7	0.00	0.00	N/A
#	1024	8	float	none	-1	187.0	0.01	0.01	N/A	201.7	0.01	0.00	N/A
#	2048	16	float	none	-1	186.3	0.01	0.01	N/A	188.0	0.01	0.01	N/A
#	4096	32	float	none	-1	185.8	0.02	0.02	N/A	187.9	0.02	0.02	N/A
#	8192	64	float	none	-1	183.7	0.04	0.04	N/A	186.6	0.04	0.04	N/A
#	16384	128	float	none	-1	184.7	0.09	0.09	N/A	186.4	0.09	0.09	N/A
#	32768	256	float	none	-1	196.4	0.17	0.16	N/A	187.0	0.18	0.17	N/A
#	65536	512	float	none	-1	339.5	0.19	0.19	N/A	186.4	0.35	0.34	N/A
#	131072	1024	float	none	-1	184.5	0.71	0.69	N/A	186.5	0.70	0.68	N/A
#	262144	2048	float	none	-1	188.7	1.39	1.35	N/A	190.3	1.38	1.33	N/A
#	524288	4096	float	none	-1	200.6	2.61	2.53	N/A	190.3	2.75	2.67	N/A
#	1048576	8192	float	none	-1	188.0	5.58	5.40	N/A	190.0	5.52	5.35	N/A
#	2097152	16384	float	none	-1	181.1	11.58	11.22	N/A	180.6	11.61	11.25	N/A
#	4194304	32768	float	none	-1	241.2	17.39	16.85	N/A	251.6	16.67	16.15	N/A
#	8388608	65536	float	none	-1	260.6	32.20	31.19	N/A	245.9	34.11	33.05	N/A
#	16777216	131072	float	none	-1	770.9	21.76	21.08	N/A	535.5	31.33	30.35	N/A
#	33554432	262144	float	none	-1	1175.6	28.54	27.65	N/A	917.4	36.58	35.43	N/A
#	67108864	524288	float	none	-1	1840.0	36.47	35.33	N/A	1696.8	39.55	38.31	N/A
#	134217728	1048576	float	none	-1	4261.2	31.50	30.51	N/A	4289.1	31.29	30.32	N/A
#	268435456	2097152	float	none	-1	8479.3	31.66	30.67	N/A	8707.3	30.83	29.87	N/A
#	536870912	4194304	float	none	-1	15369	34.93	33.84	N/A	15530	34.57	33.49	N/A
#	1073741824	8388608	float	none	-1	27005	39.76	38.52	N/A	27382	39.21	37.99	N/A
#	2147483648	16777216	float	none	-1	48739	44.06	42.68	N/A	48361	44.41	43.02	N/A
#	4294967296	33554432	float	none	-1	88526	48.52	47.00	N/A	89129	48.19	46.68	N/A
#	8589934592	67108864	float	none	-1	170385	50.41	48.84	N/A	170906	50.26	48.69	N/A
#	17179869184	134217728	float	none	-1	333902	51.45	49.84	N/A	335275	51.24	49.64	N/A

```
# Out of bounds values : 0 OK
# Avg bus bandwidth : 15.1666
#
```

```
$ cat hostfile
host1 slots=1
host2 slots=1
host3 slots=1
host4 slots=1
```

8.7.2.2 Test: All-Reduce

Both commands execute the same collective (all-reduce):

- The first command runs 32 processes (one process on each GPU on each of the four nodes).
- The second command runs 4 processes (one process on eight GPUs on each of the four nodes).

NOTE: Multi-line commands are shown in the following sections.

8.7.2.2.1 Test: All-Reduce (4 Nodes, 32 Processes)

```

/opt/AMD/install/mpi/bin/mpirun --allow-run-as-root --bind-to none --hostfile hostfile \
-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_IB_DISABLE=0 \
-x NCCL_NET_GDR_LEVEL=SYS \
-x NCCL_NET_GDR_READ=1 \
-x NCCL_P2P_LEVEL=SYS \
-x NCCL_SHM_DISABLE=1 \
-x NCCL_IB_PCI_RELAXED_ORDERING=1 \
-x HSA_FORCE_FINE_GRAIN_PCIE=1 \
-x NCCL_DMABUF_ENABLE=0 \
-x LD_LIBRARY_PATH=/opt/AMD/install/mpi/lib/ \
-x NCCL_MIN_NCHANNELS=32 --mca pml ucx --mca osc ucx --mca spml ucx --mca btl ^vader,tcp,openib,uct \
/home/test/rccl-tests/build/all_reduce_perf -b 8 -e 16G -f 2 -g 1 -c 0

# nThread 1 nGpus 1 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 0 graph: 0
#
rccl-tests: Version develop:77ae744
# Using devices
# Rank 0 Pid 14767 on irvine-waco1 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 1 Pid 14768 on irvine-waco1 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 2 Pid 14769 on irvine-waco1 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 3 Pid 14770 on irvine-waco1 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 4 Pid 14771 on irvine-waco1 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 5 Pid 14772 on irvine-waco1 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 6 Pid 14773 on irvine-waco1 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 7 Pid 14774 on irvine-waco1 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 8 Pid 15112 on irvine-waco2 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 9 Pid 15111 on irvine-waco2 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 10 Pid 15113 on irvine-waco2 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 11 Pid 15115 on irvine-waco2 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 12 Pid 15114 on irvine-waco2 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 13 Pid 15118 on irvine-waco2 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 14 Pid 15117 on irvine-waco2 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 15 Pid 15116 on irvine-waco2 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 16 Pid 15929 on irvine-waco3 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 17 Pid 15930 on irvine-waco3 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 18 Pid 15931 on irvine-waco3 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 19 Pid 15936 on irvine-waco3 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 20 Pid 15933 on irvine-waco3 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 21 Pid 15932 on irvine-waco3 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 22 Pid 15934 on irvine-waco3 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 23 Pid 15935 on irvine-waco3 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 24 Pid 22314 on irvine-waco4 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 25 Pid 22315 on irvine-waco4 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 26 Pid 22319 on irvine-waco4 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 27 Pid 22317 on irvine-waco4 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 28 Pid 22316 on irvine-waco4 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 29 Pid 22318 on irvine-waco4 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 30 Pid 22320 on irvine-waco4 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 31 Pid 22321 on irvine-waco4 device 7 [0000:dd:00.0] AMD Instinct MI300X
#
#
# 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 54.62 0.00 0.00 N/A 52.74 0.00 0.00 N/A
# 16 4 float sum -1 573.5 0.00 0.00 N/A 421.4 0.00 0.00 N/A
# 32 8 float sum -1 1032.6 0.00 0.00 N/A 52.10 0.00 0.00 N/A
# 64 16 float sum -1 51.46 0.00 0.00 N/A 51.52 0.00 0.00 N/A
# 128 32 float sum -1 52.27 0.00 0.00 N/A 52.10 0.00 0.00 N/A
# 256 64 float sum -1 51.87 0.00 0.01 N/A 52.22 0.00 0.01 N/A
# 512 128 float sum -1 52.55 0.01 0.02 N/A 52.57 0.01 0.02 N/A
# 1024 256 float sum -1 54.89 0.02 0.04 N/A 53.99 0.02 0.04 N/A
# 2048 512 float sum -1 55.57 0.04 0.07 N/A 55.91 0.04 0.07 N/A
# 4096 1024 float sum -1 58.84 0.07 0.13 N/A 57.90 0.07 0.14 N/A

```

```

8192      2048      float      sum      -1      298.6      0.03      0.05      N/A      267.1      0.03      0.06      N/A
16384     4096     float      sum      -1      63.26      0.26      0.50      N/A      58.67      0.28      0.54      N/A
32768     8192     float      sum      -1      61.00      0.54      1.04      N/A      59.31      0.55      1.07      N/A
65536    16384     float      sum      -1      64.52      1.02      1.97      N/A      62.93      1.04      2.02      N/A
131072   32768     float      sum      -1      234.2      0.56      1.08      N/A      66.39      1.97      3.82      N/A
262144   65536     float      sum      -1      79.04      3.32      6.43      N/A      77.57      3.38      6.55      N/A
524288   131072    float      sum      -1      100.6      5.21      10.10     N/A      100.4      5.22      10.12     N/A
1048576  262144    float      sum      -1      146.5      7.16      13.87     N/A      145.1      7.23      14.00     N/A
2097152  524288    float      sum      -1      189.9      11.04     21.39     N/A      189.1      11.09     21.48     N/A
4194304  1048576   float      sum      -1      210.3      19.94     38.64     N/A      210.9      19.89     38.53     N/A
8388608  2097152   float      sum      -1      251.1      33.41     64.73     N/A      252.0      33.28     64.49     N/A
16777216 4194304   float      sum      -1      1162.5     14.43     27.96     N/A      1196.3     14.02     27.17     N/A
33554432 8388608   float      sum      -1      1346.4     24.92     48.28     N/A      566.0      59.28     114.85    N/A
67108864 16777216  float      sum      -1      729.8      91.95     178.16    N/A      729.7      91.97     178.20    N/A
134217728 33554432 float      sum      -1      1448.4     92.67     179.54    N/A      3954.9     33.94     65.75     N/A
268435456 67108864 float      sum      -1      2190.3     122.56    237.45    N/A      1737.6     154.49    299.33    N/A
536870912 134217728 float      sum      -1      5146.8     104.31    202.11    N/A      3301.6     162.61    315.05    N/A
1073741824 268435456 float      sum      -1      6239.7     172.08    333.41    N/A      9101.3     117.98    228.58    N/A
2147483648 536870912 float      sum      -1      14843     144.68    280.32    N/A      14279     150.40    291.39    N/A
4294967296 1073741824 float      sum      -1      29792     144.17    279.32    N/A      29392     146.13    283.12    N/A
8589934592 2147483648 float      sum      -1      55181     155.67    301.61    N/A      52800     162.69    315.21    N/A
17179869184 4294967296 float      sum      -1      101984     168.46    326.38    N/A      101464     169.32    328.06    N/A
# Out of bounds values : 0 OK
# Avg bus bandwidth : 80.6921
#
$ cat hostfile
host1 slots=8
host2 slots=8
host2 slots=8
host4 slots=8

```

8.7.2.2.2 Test: All-Reduce (4 Nodes, 4 processes)

```

/opt/AMD/install/mpi/bin/mpirun --allow-run-as-root --bind-to none --hostfile hostfile \
-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_IB_DISABLE=0 \
-x NCCL_NET_GDR_LEVEL=SYS \
-x NCCL_NET_GDR_READ=1 \
-x NCCL_P2P_LEVEL=SYS \
-x NCCL_SHM_DISABLE=1 \
-x NCCL_IB_PCI_RELAXED_ORDERING=1 \
-x HSA_FORCE_FINE_GRAIN_PCIE=1 \
-x NCCL_DMABUF_ENABLE=0 \
-x LD_LIBRARY_PATH=/opt/AMD/install/mpi/lib/ \
-x NCCL_MIN_NCHANNELS=32 --mca pml ucx --mca osc ucx --mca spml ucx --mca btl ^vader,tcp,openib,uct \
/home/test/rccl-tests/build/all_reduce_perf -b 8 -e 16G -f 2 -g 8 -c 0

# nThread 1 nGpus 8 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 0 graph: 0
#
rccl-tests: Version develop:77ae744
# Using devices
# Rank 0 Pid 11779 on irvine-waco1 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 1 Pid 11779 on irvine-waco1 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 2 Pid 11779 on irvine-waco1 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 3 Pid 11779 on irvine-waco1 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 4 Pid 11779 on irvine-waco1 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 5 Pid 11779 on irvine-waco1 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 6 Pid 11779 on irvine-waco1 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 7 Pid 11779 on irvine-waco1 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 8 Pid 11798 on irvine-waco2 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 9 Pid 11798 on irvine-waco2 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 10 Pid 11798 on irvine-waco2 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 11 Pid 11798 on irvine-waco2 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 12 Pid 11798 on irvine-waco2 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 13 Pid 11798 on irvine-waco2 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 14 Pid 11798 on irvine-waco2 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 15 Pid 11798 on irvine-waco2 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 16 Pid 12575 on irvine-waco3 device 0 [0000:1b:00.0] AMD Instinct MI300X
# Rank 17 Pid 12575 on irvine-waco3 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 18 Pid 12575 on irvine-waco3 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 19 Pid 12575 on irvine-waco3 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 20 Pid 12575 on irvine-waco3 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 21 Pid 12575 on irvine-waco3 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 22 Pid 12575 on irvine-waco3 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 23 Pid 12575 on irvine-waco3 device 7 [0000:dd:00.0] AMD Instinct MI300X
# Rank 24 Pid 18975 on irvine-waco4 device 0 [0000:1b:00.0] AMD Instinct MI300X

```

```

# Rank 25 Pid 18975 on irvine-waco4 device 1 [0000:3d:00.0] AMD Instinct MI300X
# Rank 26 Pid 18975 on irvine-waco4 device 2 [0000:4e:00.0] AMD Instinct MI300X
# Rank 27 Pid 18975 on irvine-waco4 device 3 [0000:5f:00.0] AMD Instinct MI300X
# Rank 28 Pid 18975 on irvine-waco4 device 4 [0000:9d:00.0] AMD Instinct MI300X
# Rank 29 Pid 18975 on irvine-waco4 device 5 [0000:bd:00.0] AMD Instinct MI300X
# Rank 30 Pid 18975 on irvine-waco4 device 6 [0000:cd:00.0] AMD Instinct MI300X
# Rank 31 Pid 18975 on irvine-waco4 device 7 [0000:dd:00.0] AMD Instinct MI300X
#
#
# 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 54.58 0.00 0.00 N/A 53.03 0.00 0.00 N/A
# 16 4 float sum -1 53.96 0.00 0.00 N/A 54.36 0.00 0.00 N/A
# 32 8 float sum -1 53.65 0.00 0.00 N/A 54.29 0.00 0.00 N/A
# 64 16 float sum -1 54.08 0.00 0.00 N/A 64.56 0.00 0.00 N/A
# 128 32 float sum -1 54.53 0.00 0.00 N/A 55.62 0.00 0.00 N/A
# 256 64 float sum -1 55.31 0.00 0.01 N/A 55.09 0.00 0.01 N/A
# 512 128 float sum -1 56.05 0.01 0.02 N/A 55.97 0.01 0.02 N/A
# 1024 256 float sum -1 57.29 0.02 0.03 N/A 60.52 0.02 0.03 N/A
# 2048 512 float sum -1 58.96 0.03 0.07 N/A 59.35 0.03 0.07 N/A
# 4096 1024 float sum -1 62.39 0.07 0.13 N/A 60.23 0.07 0.13 N/A
# 8192 2048 float sum -1 64.28 0.13 0.25 N/A 62.08 0.13 0.26 N/A
# 16384 4096 float sum -1 64.95 0.25 0.49 N/A 76.97 0.21 0.41 N/A
# 32768 8192 float sum -1 73.38 0.45 0.87 N/A 75.69 0.43 0.84 N/A
# 65536 16384 float sum -1 87.13 0.75 1.46 N/A 90.08 0.73 1.41 N/A
# 131072 32768 float sum -1 127.3 1.03 1.99 N/A 127.1 1.03 2.00 N/A
# 262144 65536 float sum -1 122.4 2.14 4.15 N/A 125.3 2.09 4.05 N/A
# 524288 131072 float sum -1 121.6 4.31 8.35 N/A 118.3 4.43 8.59 N/A
# 1048576 262144 float sum -1 158.9 6.60 12.79 N/A 155.8 6.73 13.04 N/A
# 2097152 524288 float sum -1 197.9 10.60 20.53 N/A 198.1 10.59 20.51 N/A
# 4194304 1048576 float sum -1 219.1 19.14 37.09 N/A 219.6 19.10 37.00 N/A
# 8388608 2097152 float sum -1 261.4 32.09 62.17 N/A 260.9 32.15 62.29 N/A
# 16777216 4194304 float sum -1 340.3 49.30 95.51 N/A 338.5 49.56 96.03 N/A
# 33554432 8388608 float sum -1 531.7 63.11 122.27 N/A 531.9 63.09 122.23 N/A
# 67108864 16777216 float sum -1 735.2 91.28 176.85 N/A 734.0 91.43 177.14 N/A
# 134217728 33554432 float sum -1 1408.2 95.31 184.66 N/A 1390.6 96.52 187.00 N/A
# 268435456 67108864 float sum -1 1739.4 154.32 299.00 N/A 1743.2 153.99 298.35 N/A
# 536870912 134217728 float sum -1 3114.3 172.39 334.00 N/A 3120.1 172.07 333.38 N/A
# 1073741824 268435456 float sum -1 6009.4 178.68 346.19 N/A 6024.9 178.22 345.29 N/A
# 2147483648 536870912 float sum -1 11929 180.02 348.80 N/A 11973 179.36 347.52 N/A
# 4294967296 1073741824 float sum -1 24307 176.70 342.35 N/A 24003 178.94 346.69 N/A
# 8589934592 2147483648 float sum -1 46921 183.07 354.70 N/A 46818 183.48 355.49 N/A
# 17179869184 4294967296 float sum -1 94232 182.31 353.23 N/A 93287 184.16 356.81 N/A
# Out of bounds values : 0 OK
# Avg bus bandwidth : 97.2593
#
$ cat hostfile
host1 slots=1
host2 slots=1
host2 slots=1
host4 slots=1

```

8.7.3 Debugging RCCL

This section contains information on debugging RCCL.

8.7.3.1 RCCL Environment Variable: NCCL_DEBUG

The RCCL environment variable `NCCL_DEBUG` can be used to output debug info during the execution of the RCCL tests. The use of the `NCCL_DEBUG` is shown in [MultiNode RCCL Collectives Using Open MPI](#). Passing a value `NCCL_DEBUG=INFO` outputs more verbose info during the tests and can be used to check if Peer Memory Direct is indeed being used in addition to other useful debug info.

Correct CPU and GPU binding is necessary for optimal system performance. Sometimes the affinity settings are affected by job schedulers such as SLURM which may degrade RCCL performance. `NCCL_DEBUG=INFO` prints information that can be used to confirm the affinity masks are set correctly.

8.7.3.2 RCCL Environment Variable: NCCL_SOCKET_IFNAME

RCCL uses an out-of-band TCP connection for bootstrapping. Sometimes, there can be misconfigured IP interfaces (such as `docker0`) that can confuse RCCL. `NCCL_DEBUG=INFO` displays the network interface used by RCCL for bootstrapping. To avoid any potential RCCL bootstrap related problem, use the environment variable `NCCL_SOCKET_IFNAME=<ifname>` to specify the desired bootstrap interface name as shown in [MultiNode RCCL Collectives Using Open MPI](#).

With `NCCL_DEBUG=INFO`, the RCCL output will show several lines indicating the RCCL bootstrap interface name as follows:

```
[0] NCCL INFO Bootstrap : Using eno8303:10.77.69.182<0>
```

8.7.3.3 RCCL Environment Variable: NCCL_NET_GDR_LEVEL

The RCCL environment variable `NCCL_NET_GDR_LEVEL` can be used to enable or disable the use of Peer Memory Direct during the RCCL tests. Passing a value `NCCL_NET_GDR_LEVEL=0` disables the use of Peer Memory Direct between the GPU and the NIC. This variable can be used to isolate if a given RCCL problem is related to Peer Memory Direct or not.

When Peer Memory Direct is being used, the `NCCL_DEBUG=INFO` output should include multiple lines indicating `NET/IB/<>/GDRDMA` as follows:

```
NCCL INFO Channel 00/0 : 15[e6000] -> 0[1c000] [receive] via NET/IB/0/GDRDMA comm 0x1aae1d0 nRanks 32
NCCL INFO Channel 01/0 : 14[d4000] -> 1[42000] [send] via NET/IB/6/GDRDMA comm 0x2736290 nRanks 32
```

When Peer Memory Direct is disabled via `NCCL_NET_GDR_LEVEL=0` or Peer Memory Direct can not be used for any reason, the `NCCL_DEBUG=INFO` output should include multiple lines indicating `NET/IB/0` as follows:

```
NCCL INFO Channel 00/0 : 10[55000] -> 1[42000] [receive] via NET/IB/0 comm 0x1a72740 nRanks 32
```

8.7.3.4 RCCL Environment Variable: NCCL_IB_HCA

The RCCL Environment variable `NCCL_IB_HCA` as shown in section [MultiNode RCCL Collectives using Open MPI](#) can be used to select and deselect the RDMA capable NICs from the RCCL tests. This environment variable can be useful to isolate a given problem NIC(s).

Chapter 9: 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-dev
sudo apt install gcc make libtool autoconf librdmacm-dev rdmacm-utils infiniband-diags ibverbs-utils
perfctest 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-232.0.155.5.tar.gz
cd netxtreme-peer-mem-232.0.155.5
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===== 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-232.0.155.5.tar.gz
cd libbnxt_re-232.0.155.5
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=====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\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 "$BUILT_MD5SUM" ]]; then
    echo -e "Failed to get the md5sum of the built libbnxt_re lib\n\n\n"
    exit 1
elif [[ -z "$INSTALLED_MD5SUM" ]]; then
    echo -e "Failed to get the md5sum of the installed libbnxt_re lib\n\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"

```

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-232.0.155.5.tar.gz
cd netxtreme-peer-mem-232.0.155.5
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-232.0.155.5.tar.gz
cd libbnxt_re-232.0.155.5
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: Helpful ROCm Commands

The `rocm-smi` command and its options provide very useful information related to the AMD GPUs and how the GPU.

B.1 Checking the Type of GPUs on the Host

To check the type of GPU on the host, use the following commands:

```
$ rocm-smi --showhw --showallinfo | grep -i card
```

```
GPU[0]      : Card series:      AMD Instinct MI300X OAM
GPU[0]      : Card model:       0x74a1
GPU[0]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[0]      : Card SKU:     MI3SRIOV
GPU[1]      : Card series:      AMD Instinct MI300X OAM
GPU[1]      : Card model:       0x74a1
GPU[1]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[1]      : Card SKU:     MI3SRIOV
GPU[2]      : Card series:      AMD Instinct MI300X OAM
GPU[2]      : Card model:       0x74a1
GPU[2]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[2]      : Card SKU:     MI3SRIOV
GPU[3]      : Card series:      AMD Instinct MI300X OAM
GPU[3]      : Card model:       0x74a1
GPU[3]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[3]      : Card SKU:     MI3SRIOV
GPU[4]      : Card series:      AMD Instinct MI300X OAM
GPU[4]      : Card model:       0x74a1
GPU[4]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[4]      : Card SKU:     MI3SRIOV
GPU[5]      : Card series:      AMD Instinct MI300X OAM
GPU[5]      : Card model:       0x74a1
GPU[5]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[5]      : Card SKU:     MI3SRIOV
GPU[6]      : Card series:      AMD Instinct MI300X OAM
GPU[6]      : Card model:       0x74a1
GPU[6]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[6]      : Card SKU:     MI3SRIOV
GPU[7]      : Card series:      AMD Instinct MI300X OAM
GPU[7]      : Card model:       0x74a1
GPU[7]      : Card vendor:    Advanced Micro Devices, Inc. [AMD/ATI]
GPU[7]      : Card SKU:     MI3SRIOV
```

B.2 Checking the PCIe BUS ID of Each GPU

To check the PCIe BUS ID of each GPU, use the following commands:

```
$ rocm-smi --showbus
```

```
===== ROCm System Management Interface =====
===== PCI Bus ID =====
GPU[0]      : PCI Bus: 0000:1C:00.0
GPU[1]      : PCI Bus: 0000:42:00.0
GPU[2]      : PCI Bus: 0000:55:00.0
GPU[3]      : PCI Bus: 0000:68:00.0
GPU[4]      : PCI Bus: 0000:9E:00.0
GPU[5]      : PCI Bus: 0000:C2:00.0
GPU[6]      : PCI Bus: 0000:D4:00.0
GPU[7]      : PCI Bus: 0000:E6:00.0
=====
===== End of ROCm SMI Log =====
```

B.3 Checking the Processes Running on Each GPU

To check the process running on each GPU, use the following commands:

```
$ rocm-smi --showpidgpus
```

```
$ while true; do rocm-smi --showpidgpus ; sleep 1; done
```

```
===== ROCm System Management Interface =====
===== GPUs Indexed by PID =====
PID 2632418 is using 8 DRM device(s):
0 3 4 2 1 7 6 5
=====
===== End of ROCm SMI Log =====
```

```
===== ROCm System Management Interface =====
===== GPUs Indexed by PID =====
PID 2632418 is using 8 DRM device(s):
0 3 4 2 1 7 6 5
=====
===== End of ROCm SMI Log =====
```

```
$ while true; do rocm-smi --showpidgpus ; sleep 1; done
```

```
===== ROCm System Management Interface =====
===== GPUs Indexed by PID =====
PID 2632526 is using 1 DRM device(s):
6
PID 2632524 is using 1 DRM device(s):
```

```
4
PID 2632522 is using 1 DRM device(s):
2
PID 2632520 is using 1 DRM device(s):
0
PID 2632527 is using 1 DRM device(s):
7
PID 2632525 is using 1 DRM device(s):
5
PID 2632523 is using 1 DRM device(s):
3
PID 2632521 is using 1 DRM device(s):
1
```

```
=====
===== End of ROCm SMI Log =====
```

```
===== ROCm System Management Interface =====
===== GPUs Indexed by PID =====
```

```
PID 2632526 is using 1 DRM device(s):
6
PID 2632524 is using 1 DRM device(s):
4
PID 2632522 is using 1 DRM device(s):
2
PID 2632520 is using 1 DRM device(s):
0
PID 2632527 is using 1 DRM device(s):
7
PID 2632525 is using 1 DRM device(s):
5
PID 2632523 is using 1 DRM device(s):
3
PID 2632521 is using 1 DRM device(s):
1
```

```
=====
===== End of ROCm SMI Log =====
```


B.4 Checking the PCIe Bandwidth in Use for Each GPU

To check the PCIe bandwidth in use for each GPU, use the following commands:

```
$ rocm-smi -b
```

```
$ while true; do rocm-smi -b ; sleep 1; done
```

```
===== ROCm System Management Interface =====
===== Measured PCIe Bandwidth =====
GPU[0]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.018
GPU[1]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 1.385
GPU[2]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.026
GPU[3]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.009
GPU[4]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.026
GPU[5]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.728
GPU[6]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.035
GPU[7]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 0.000
=====
===== End of ROCm SMI Log =====
```

```
===== ROCm System Management Interface =====
===== Measured PCIe Bandwidth =====
GPU[0]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 1441.457
GPU[1]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 18290.621
GPU[2]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 111.046
GPU[3]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 8335.574
GPU[4]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 20420.376
GPU[5]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21314.964
GPU[6]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21409.510
GPU[7]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21592.331
=====
===== End of ROCm SMI Log =====
```

```
===== ROCm System Management Interface =====
===== Measured PCIe Bandwidth =====
GPU[0]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21303.333
GPU[1]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21573.090
GPU[2]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21352.123
GPU[3]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21854.864
GPU[4]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21693.946
GPU[5]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21848.413
GPU[6]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21749.485
GPU[7]      : Estimated maximum PCIe bandwidth over the last second (MB/s): 21810.975
=====
===== End of ROCm SMI Log =====
```

Appendix C: 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.

C.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
```

C.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
```

C.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
```

C.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 D: PCIe Link Speed and Width Related Scripts

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

D.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
```

D.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 E: References

E.1 Broadcom Ethernet Network Adapter User Guide

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

E.2 ROCm Software installation on Linux

<https://rocm.docs.amd.com/projects/install-on-linux/en/latest/>

Appendix F: Terminology

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

Table 3: 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
OCF	Open Compute Project
Open MPI	Open Message Passing Interface
PFC	Priority Flow Control
RCCL	ROCm Communication Collectives Library
RDMA	Remote Direct Memory Access
NIC	Network Interface Card
RoCE	RDMA over Converged Ethernet
ROCm	AMD open source software stack designed for GPU compute
SONiC	Software for Open Networking in the cloud
UCX	Unified Communication X
VLAN	Virtual Local Area Network
XGMI	AMD Infinity Fabric

Revision History

957608-AN205; March 20, 2025

Updated:

- [Broadcom Ethernet NIC Software Installation with Peer Memory Direct](#)
- [Topology and Sample Test Results](#)
- [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-AN204; November 5, 2024

Updated:

- Updated revision history.

957608-AN203; October 28, 2024

Updated:

- Formatting updates.

957608-AN202; October 22, 2024

Updated:

- Updated GPU information.

957608-AN201; September 23, 2024

Added:

- General content updates.

957608-AN200; July 31, 2024

Initial release.

