

# Deploy VMware Private AI on HGX servers with Broadcom Ethernet Networking

## Table of contents

|                                                                 |    |
|-----------------------------------------------------------------|----|
| Executive Summary . . . . .                                     | 3  |
| Audience . . . . .                                              | 3  |
| Revision History . . . . .                                      | 4  |
| Core Components . . . . .                                       | 5  |
| HGX servers . . . . .                                           | 5  |
| Networking components . . . . .                                 | 6  |
| Recommended BIOS and firmware settings . . . . .                | 7  |
| ESX Settings . . . . .                                          | 8  |
| Change GPU's reset type to "flr" . . . . .                      | 8  |
| Passthru GPU, Thor2 and NVSwitch with hardware labels . . . . . | 8  |
| Sanity check on the ACS related settings . . . . .              | 9  |
| VM Settings . . . . .                                           | 10 |
| Deploy DLVM with DirectPath I/O . . . . .                       | 10 |
| Relaxing ACS settings for P2P between PCIe devices: . . . . .   | 13 |
| Additional performance enhancement settings: . . . . .          | 13 |
| Deploy DLVM with vGPU . . . . .                                 | 13 |
| Performance . . . . .                                           | 14 |
| Running RDMA with Perftest . . . . .                            | 14 |
| Running GPUDirect RDMA with Perftest . . . . .                  | 15 |
| Running NCCL Tests . . . . .                                    | 17 |
| Running LLM benchmark in a single node . . . . .                | 24 |
| Inference sizing guidance . . . . .                             | 30 |
| Conclusion . . . . .                                            | 30 |
| Appendix . . . . .                                              | 31 |
| A: Firmware update . . . . .                                    | 31 |
| B: Configure Tomahawk5 ethernet switch . . . . .                | 33 |
| C: Configure GPU and Thor2 in VM . . . . .                      | 40 |
| D: Configure routing for the backend network . . . . .          | 50 |
| E: Customized NCCL_TOPO_FILE in VM . . . . .                    | 54 |
| F: Terminology . . . . .                                        | 57 |
| Reference . . . . .                                             | 57 |
| About the author . . . . .                                      | 58 |
| Acknowledgments . . . . .                                       | 58 |

## Executive Summary

This document serves as a follow-up to our earlier [Reference Design for Inference](#) that details the deployment of VMware Private AI on HGX servers using Broadcom Ethernet networking, specifically Thor2 NICs and a Tomahawk 5-based switch. Unlike the previous document, this guide focuses on a deployment using DirectPath I/O (passthrough) GPUs and Thor2 NICs, listing vGPU and NVIDIA AI Enterprise as optional choices.

The deployment guide covers end-to-end setup and configuration, including BIOS and firmware settings for HGX servers, ESX configuration for GPU and network devices, and test routines for NCCL, and LLM inference benchmark to validate expected performance.

This solution is designed for VMware Private AI Certified HGX systems featuring 4x or 8x H100/H200 GPUs with NVSwitch and NVLink interconnects, running AI inference workloads. The target environment is a VCF-based private cloud with Broadcom 400G BCM957608 NICs and clustered NVIDIA H100 GPUs connected through Ethernet.

The document primarily addresses deploying Linux-based AI/ML clusters over Ethernet in VCF, enabling GPU-based collectives and inference models across multiple hosts. Such clusters can scale with multiple NICs and GPUs per host, interconnected through one or more Ethernet switches.

### Key Topics Covered:

- **Broadcom NICs and NVIDIA GPUs:** How to use Broadcom NICs and NVIDIA GPUs on Ubuntu-based DLVMs within VCF.
- **Network configuration:** Configuration of Ethernet NICs and switches for RoCE with NVIDIA GPUs.
- **Benchmark testing:** Running benchmark tests with collective communications libraries such as NCCL.
- **LLM inference:** Launch LLM inference using NIM and vLLM and benchmark their performance.

### Verified Test Environment:

NVIDIA GPU-related tests detailed in this document were verified on a deep learning VM (DLVM) running Ubuntu 22.04.03 with a 5.15.0-89-generic kernel in VCF 9.0.

### Broadcom Thor2 (BCM57608 / BCM957608) NICs:

Broadcom's Thor2 (BCM57608 / BCM957608) family of NICs support RoCE and Peer Memory Direct (or "peer-to-peer direct memory access"), offering a maximum speed of 400Gbps. These NICs are available in both PCIe and OCP3.0 Form Factors.

### Additional Resources:

- Broadcom Ethernet NICs: <https://www.broadcom.com/products/ethernet-connectivity/network-adapters>
- CUDA: <https://developer.nvidia.com/cuda-downloads>
- NCCL: <https://developer.nvidia.com/nccl/nccl-download>

## Audience

- **New Users:** Organizations planning to procure or already owning NVIDIA-Certified HGX Systems who intend to deploy VMware Private AI for AI inference workloads on a VCF-based private cloud infrastructure with Broadcom networking hardware.
- **Infrastructure Architects:** VCF administrators, DevOps, and SRE teams responsible for the deployment and management of both physical and virtual infrastructures. This paper offers guidance on hardware, networking, and system configurations for integrating VMware Private AI.
- **Software Architects:** Data scientists and AI developers who will develop inference workloads on this platform. This document provides architecture and performance insights to optimize AI models and workloads.

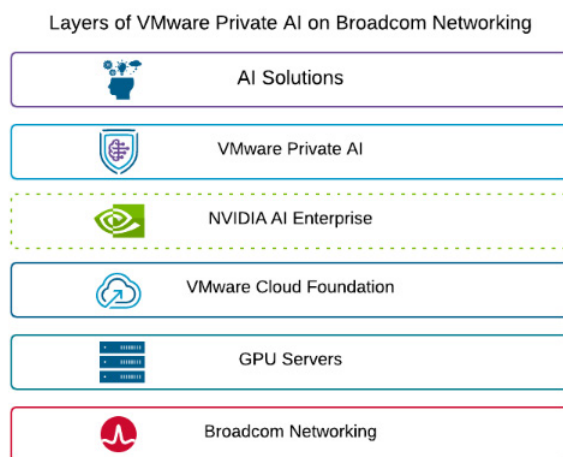
## Revision History

| Date     | Version | Change Summary                 |
|----------|---------|--------------------------------|
| Aug 2025 | 1       | Initial release based on VCF 9 |



## Core Components

Figure 1 illustrates the solution's layered architecture. Each layer is a critical integration point, typically requiring manual setup and configuration for optimal GenAI solution execution. VMware Private AI, a set of services within VMware Cloud Foundation (VCF), is available for purchase. NVIDIA AI Enterprise is an optional component in this deployment, shown as a dashed block. This is because a GPU can be utilized via either DirectPath I/O or vGPU (which requires NVIDIA AI Enterprise). This guide chooses DirectPath I/O for deployment.



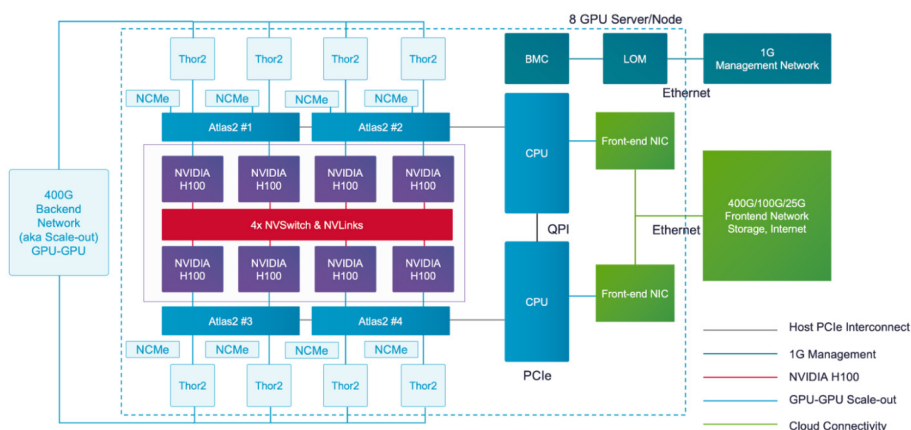
**Figure 1:** VMware Private AI on HGX systems layered architecture with Broadcom Ethernet networking

## HGX servers

The NVIDIA HGX platform features a single baseboard that supports either four or eight Hopper SXMs, or eight NVIDIA Blackwell or NVIDIA Blackwell Ultra SXMs.

An 8x GPU HGX server, commonly used in VMware Private AI Certified platforms (refer to the [Broadcom compatibility guide \(BCG\)](#) for details: click the link, then select “Desired Compatibility Guide > Platform & Compute > Systems / Servers > Select the VMware Private AI in the Feature”), typically contains the following components.

- 4x Broadcom Atlas 2 (PEX89XXX) PCIe Switches (synthetic mode enabled, each logically partitioned as 2)
- 8x NVIDIA H100/H200 GPUs
- 8x Thor 2 Ethernet NICs



**Figure 2:** Typical NIC, GPU, and PCIe switch configuration inside an HGX server

The use of one NIC per GPU enables each accelerator to have a dedicated high-bandwidth, low-latency path to the network. This design is particularly important when workloads exceed the capacity of a single server, such as in distributed LLM inference, model training and fine-tuning. In these cases, collective communication operations (e.g., all-reduce, all-gather) dominate performance, and bottlenecks can arise if GPUs must share network bandwidth. By maintaining a 1:1 GPU-to-NIC ratio, data movement between nodes can proceed in parallel without oversubscription, ensuring efficient scaling across multi-node clusters.

Networking components

Ethernet is an optimal choice for VMware Private AI inference deployments due to its simplicity, scalability, and cost-effectiveness. It also provides flexibility for future infrastructure needs, such as model customization, fine-tuning and training.

Proper networking is crucial for preventing bottlenecks and performance degradation in AI workloads when using VMware Private AI on HGX. Ethernet’s suitability for both AI inference and training is enhanced by advancements like RoCEv2 and lossless fabrics, along with its open ecosystem and scalability.

| Component       | Specifications                                                                                          |
|-----------------|---------------------------------------------------------------------------------------------------------|
| Ethernet NIC    | 8x Thor2: P1400GD 57608 400 Gb/s per server                                                             |
| Ethernet Switch | Manufacturer: UFISPACE, Model: S9321-64E F2B, OS: Sonic 4.5.0 daily_250417_0200_303-Enterprise_Advanced |
| Ethernet Cable  | QSFP56-DD 400GBASE-CR8-DAC-3.5M, Single port, breakout to 1x400Gb/s                                     |

Table 1. Network components are utilized in this deployment

Figure 3 shows an example of a connection layout. We recommend a minimum configuration for the Workload Domain(WLD) of 4 HGX servers and 1 workload network switch. Additional HGX servers can be integrated if the workload Ethernet switch has a higher radix (exceeding 32) or if a multi-layer network design employs multiple switches.

**Note:** To reduce the cost of the first WLD, it’s not necessary for all four servers to be GPU-capable. You can have four servers in the WLD, with only one being an HGX or GPU-capable server.

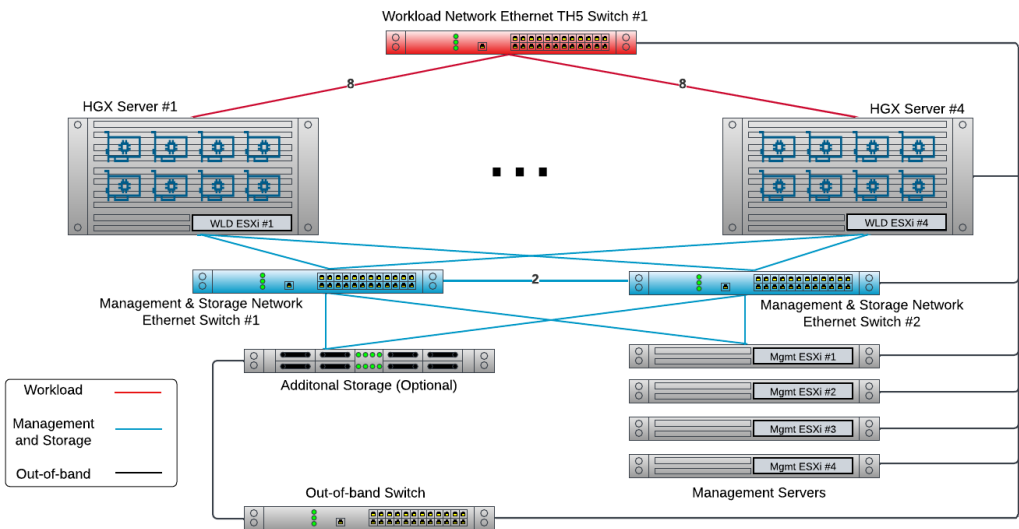


Figure 3: Physical architecture with a minimum of 4 HGX servers based on Ethernet switch radix = 32. For more information on the enabling products and technologies, refer to [Broadcom Networking](#).

## Recommended BIOS and firmware settings

- Keep the BIOS and other firmware updated to the latest version provided by your vendor. Refer to **Appendix A** about how to upgrade the firmware of the Atlas2 PCIe Switch.
- Enable ACS: For Dell Servers, enable “Virtualization Technology” in the iDRAC. Find similar settings from your vendor.
- Enable SRIOV: For Dell servers, enable “Global SRIOV.” Although SRIOV was not enabled on Thor2 NICs in this document, enabling it will not negatively impact DirectPath IO performance and is beneficial for other workload testing (e.g., vGPU, other IB or Ethernet NICs that require SRIOV).
- Enable above 4G decoding (also known as memory mapped I/O above 4GB or PCI 64-bit resource handling above 4G). This setting is typically found under the Advanced, Processor Configuration, or Memory Configuration sections in the BIOS setup.
- Performance Per Watt (OS) is the recommended system power profile setting (Figure 3). HPE servers have a similar profile setting. Once this is set, choose **High Performance** for ESXi power management.

ESX Settings

Change GPU’s reset type to “flr”

This step is required before passthrough GPUs in vSphere Client.

- 1. Edit the /etc/vmware/passthru.map file in ESX.
- 2. Find the entry for NVIDIA and change the resetMethod from “bridge” to “flr”.
- 3. Reboot Required to take effect

```
# Original content in /etc/vmware/passthru.map in ESXi
...
# NVIDIA (FLR issue on Ampere and Hopper GPUs)
10de ffff bridge false

# Change to the following
...
# NVIDIA (FLR issue on Ampere and Hopper GPUs)
10de ffff flr false
```

Passthru GPU, Thor2 and NVSwitch with hardware labels

Configure all GPUs and Thor2 NICs as passthrough, and ensure they are labeled correctly.

The purpose of this deployment was to assess the performance of a server configured to use DirectPath I/O (passthrough) for all its devices, dedicating them to a single VM. Although other configurations could involve passing through selected devices to multiple concurrent VMs, our primary objective was to enable passthrough for all GPUs and Thor2 NICs, ensuring correct labeling.

**Note:** For 8x GPU and 8x NIC HGX servers, the SBDF (Segment-Bus-Device-Function, or each device’s PCIe ID) is likely to match the configuration in Figure 4. For other server configurations, such as 4-GPU HGX servers, please refer to an upcoming technical paper on Shared-Passthru or contact VMware Support for guidance on specifying GPU and NIC IDs.

PCI Devices

PASSTHROUGH-ENABLED DEVICESALL PCI DEVICES

TOGGLE PASSTHROUGHCONFIGURE SR-IOVHARDWARE LABEL

| <input type="checkbox"/> | ID           | Passthrough | SR-IOV           | Hardware Label | Vendor Name                    | Device Name                               |
|--------------------------|--------------|-------------|------------------|----------------|--------------------------------|-------------------------------------------|
| <input type="checkbox"/> | 0000:19:00.0 | Enabled     | Not Configurable | GPU0           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:1A:00.0 | Enabled     | Not Configurable | RoCE1          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:3B:00.0 | Enabled     | Not Configurable | GPU1           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:3C:00.0 | Enabled     | Not Configurable | RoCE1          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:4C:00.0 | Enabled     | Not Configurable | GPU2           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:4D:00.0 | Enabled     | Not Configurable | RoCE2          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:5D:00.0 | Enabled     | Not Configurable | GPU3           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:5E:00.0 | Enabled     | Not Configurable | RoCE3          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:83:00.0 | Enabled     | Not Configurable | NVS0           | NVIDIA Corporation             | GH100 [H100 NVSwitch]                     |
| <input type="checkbox"/> | 0000:84:00.0 | Enabled     | Not Configurable | NVS1           | NVIDIA Corporation             | GH100 [H100 NVSwitch]                     |
| <input type="checkbox"/> | 0000:85:00.0 | Enabled     | Not Configurable | NVS2           | NVIDIA Corporation             | GH100 [H100 NVSwitch]                     |
| <input type="checkbox"/> | 0000:86:00.0 | Enabled     | Not Configurable | NVS3           | NVIDIA Corporation             | GH100 [H100 NVSwitch]                     |
| <input type="checkbox"/> | 0000:9B:00.0 | Enabled     | Not Configurable | GPU4           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:9C:00.0 | Enabled     | Not Configurable | RoCE4          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:BB:00.0 | Enabled     | Not Configurable | GPU5           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:BC:00.0 | Enabled     | Not Configurable | RoCE5          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:CB:00.0 | Enabled     | Not Configurable | GPU6           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:CC:00.0 | Enabled     | Not Configurable | RoCE6          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |
| <input type="checkbox"/> | 0000:DB:00.0 | Enabled     | Not Configurable | GPU7           | NVIDIA Corporation             | GH100 [H100 SXM5 80GB]                    |
| <input type="checkbox"/> | 0000:DC:00.0 | Enabled     | Not Configurable | RoCE7          | Broadcom Inc. and subsidiaries | BCM57608 1x400G QSFP-DD PCIe Ethernet NIC |

Manage columnsDeselect All20 items

Figure 4: Setting hardware labels for Passthroughed GPU, Thor2, and NVSwitch

## Sanity check on the ACS related settings

In virtual environments, ESX enables PCIe Access Control Services (ACS) by default for security. To verify the ACS-related variables on ESX, check the following values.

```
# 1. disableACSCheck
[ESX] esxcli system settings kernel list -o disableACSCheck
Name                Type Configured Runtime Default Description
-----
disableACSCheck    Bool FALSE      FALSE    FALSE    Bypass ACS capability checks on all PCIE devices

# 2. atsSupport
[ESX] esxcli system settings kernel list -o atsSupport
Name                Type Configured Runtime Default Description
-----
atsSupport          Bool TRUE        TRUE     TRUE     Enable Support for PCIe ATS.

# enableACSDTP2P (Available only in ESXi 9)
[ESX] esxcli system settings kernel list -o enableACSDTP2P
Name                Type Configured Runtime Default Description
-----
enableACSDTP2P      Bool TRUE        TRUE     TRUE     Enable ACS Direct Translated P2P.

# If any of these values deviate from their defaults, set them as follows:
[ESX] esxcli system settings kernel set -s enableACSDTP2P -v true
```

Alternatively, you can use vCenter to check and configure these settings:

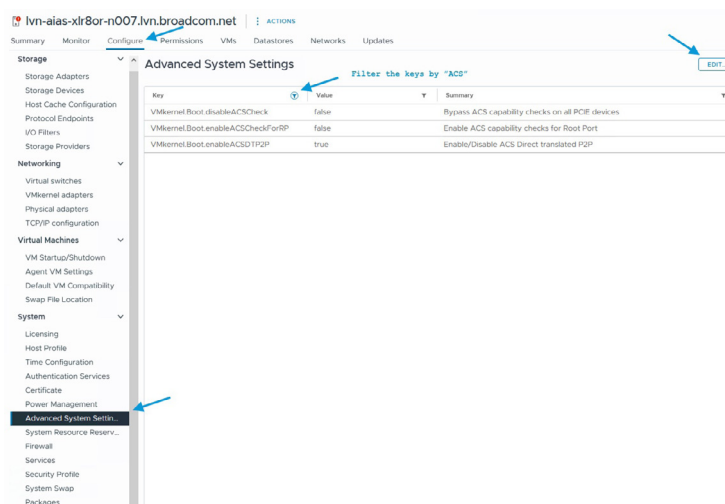


Figure 5: ACS-related settings in vSphere UI

1. Migrate all virtual machines (VMs) and place the host into maintenance mode.
2. Navigate to the host's Configure tab.
3. In the middle pane, go to **System** and click on **Advanced System Settings**.
4. Click **Edit**.
5. Filter the keys by "ACS" and "atssupport."
6. Review their current values.
7. If changes are required, remember that these are VMkernel settings and will necessitate a host reboot to take effect.

## VM Settings

While you have the option to create a VM with your own OS, leveraging the VMware Private AI Services allows for faster deployment through DLVM. First, follow the procedure to [Create a Content Library with Deep Learning VM Images](#) in vCenter. After this, you can proceed with deployment. Accessing a GPU can be achieved by two methods: DirectPath I/O or vGPU. If you are not using DLVM, make sure your VM hardware version is above 20.

### Deploy DLVM with DirectPath I/O

#### DLVM template customization

When deploying DLVM via DirectPath I/O, the template deployment procedure (step 7) presents three main categories of values. Since this document does not use vGPU stacks and workloads will be deployed separately, the five settings in the “vGPU Driver Installation” category and the 11 settings in the “DL Workload Automation” category can be disregarded. Only the information in the “base OS properties” category needs to be completed.

The screenshot displays the VMware DLVM Template Deployment UI. On the left, a vertical list of steps is shown, with '7 Customize template' highlighted. The main area is titled 'Customize template' and contains a message: 'Customize the deployment properties of this software solution.' Below this, a green status bar indicates 'All properties have valid values'. A table lists three categories of settings:

| Category                   | Number of settings |
|----------------------------|--------------------|
| > Base OS Properties       | 6 settings         |
| > vGPU Driver Installation | 5 settings         |
| > DL Workload Automation   | 11 settings        |

At the bottom right, there are three buttons: 'CANCEL', 'BACK', and 'NEXT'.

**Figure 6:** DLVM Template Deployment UI

In the base OS properties, the hostname, user password, and “Encoded user-data” section can be modified by assigning a static IP.

VMware-Deep-Learning-VM-Image-nvvgpu-9.0.1.0.24882593 - New Virtual Machine from Content Library

1 Select a name and folder  
2 Select a compute resource  
3 Review details  
4 License agreements  
5 Select storage  
6 Select networks  
7 **Customize template**  
8 Ready to complete

**Customize template**

Customize the deployment properties of this software solution.

All properties have valid values

| Property                       | Value                                                                                                                                                                                                                                                                                                                                    |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Instance ID                    | Required. A unique ID for the instance.<br>ID-ovf                                                                                                                                                                                                                                                                                        |
| Hostname                       | The hostname for the appliance.<br>dlvm                                                                                                                                                                                                                                                                                                  |
| URL to seed instance data from | This field is optional, but indicates that the instance should 'seed' user-data and meta-data from the given url. If set to 'http://tinyurl.com/sm-' is given, meta-data will be pulled from http://tinyurl.com/sm-meta-data and user-data from http://tinyurl.com/sm-user-data. Leave this empty if you do not want to seed from a url. |
| SSH public key                 | Optional. If set, the instance will populate the default user's authorized_keys with this value.                                                                                                                                                                                                                                         |
| Encoded user-data              | Base64-encoded text of user data for cloud-init. It typically includes scripts, commands, or metadata that the VM instance uses to configure itself, such as setting up users, installing packages, and initiating DL workloads.                                                                                                         |
| Default user's password        | Default single login password for the 'vmware' user. The user will use this password to login the VM for the first time, and will be                                                                                                                                                                                                     |

CANCEL BACK NEXT

**Figure 7:** Base OS Properties

The following `cloud-config.yaml` script can be used for [Assign a Static IP Address](#) to a DLVM.

```
#cloud-config.yaml
<instructions_for_your_DL_workload>

manage_etc_hosts: true

write_files:
  - path: /etc/netplan/50-cloud-init.yaml
    permissions: '0600'
    content: |
      network:
        version: 2
        renderer: networkd
        ethernets:
          ens33:
            dhcp4: false # disable DHCP4
            addresses: [x.x.x.x/x] # Set the static IP address and mask
            routes:
              - to: default
                via: x.x.x.x # Configure gateway
            nameservers:
              addresses: [x.x.x.x, x.x.x.x] # Provide the DNS server address. Separate multiple DNS server addresses with commas.

runcmd:
  - netplan apply
```

**Figure 8:** cloud-config.yaml for Setting Static IP

For air-gapped environments requiring a proxy server, consult the [Configure a Deep Learning VM with a Proxy Server](#) page

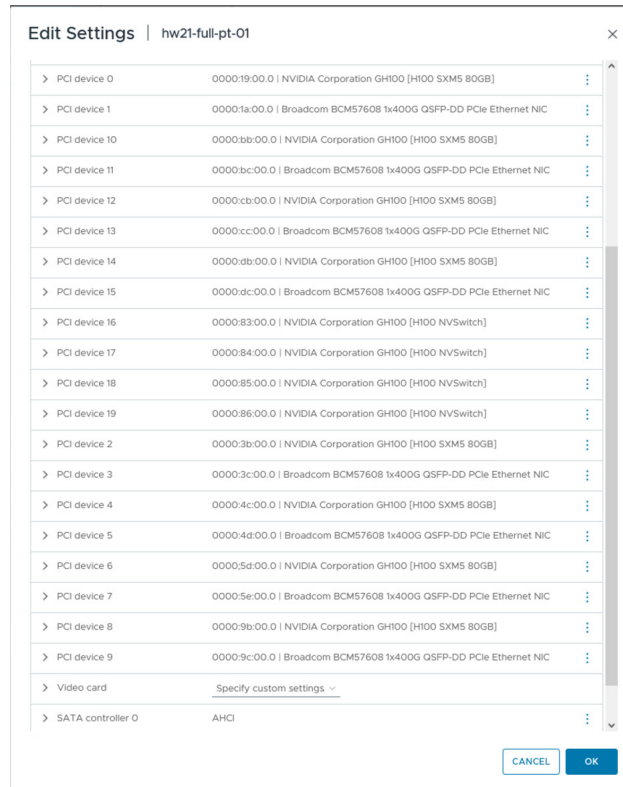
After preparing the `cloud_init.yaml` file, encode it using base64:

```
base64 -i cloud_init.yaml > encoded_output.txt
```

Finally, copy the content from `encoded_output.txt` into the "Encoded user-data" field.

**Add PCI devices by DirectPath I/O**

Add the 8x GPUs, 8x NICs, and 4x NVSwitches to the DLVM in the following sequence: GPU0, NIC0, GPU1, NIC1, ..., GPU7, NIC7, followed by NVS0 to NVS3.



**Figure 9:** Add 20x Passthrough Devices into a single VM.

**Best practice:** To expedite the process of adding 20 PCI devices to a VM (as shown in Figure 14), utilize the vHPC toolkit. The `--vm` flag can be replaced with the `--file` flag to input a list of VM names from a file.

```
# Download repo to your laptop or jumpbox
git clone https://github.com/vmware-archive/vhpc-toolkit.git
cd vhpc-toolkit
virtualenv -p python3 venv
source venv/bin/activate
pip install -r requirements.txt
pip install -e

# Modify config/vCenter.conf
server: testvc.aips.ai.broadcom.net
username: administrator@vsphere.local
password: {your_vSphere_password}

# Check the connection working fine from your jumpbox to vSphere
cd bin
./vhpc_toolkit view

# Add 8x GPUs, 8x Thor2 NICs, 4x NVswitches to your targetd DLVM
VM=hw21-full-pt-01
./vhpc_toolkit passthru --vm $VM --add --device 0000:19:00.0 0000:1A:00.0 0000:3B:00.0 0000:3C:00.0
0000:4C:00.0 0000:4D:00.0 0000:5D:00.0 0000:5E:00.0 0000:9B:00.0 0000:9C:00.0 0000:BB:00.0 0000:BC:00.0
0000:CB:00.0 0000:CC:00.0 0000:DB:00.0 0000:DC:00.0 0000:83:00.0 0000:84:00.0 0000:85:00.0 0000:86:00.0
--mmio_size 1280
# If you have multiple VMs, create a file that contains all the VM's names line by line
cat vms.txt
hw21-full-pt-01
```



```
hw21-full-pt-02
...
# Then use "--file" flag to add 20x PCIe devices to each VM
./vhpc_toolkit passthru --file vms.txt --add --device 0000:19:00.0 0000:1A:00.0 0000:3B:00.0 0000:3C:00.0
0000:4C:00.0 0000:4D:00.0 0000:5D:00.0 0000:5E:00.0 0000:9B:00.0 0000:9C:00.0 0000:BB:00.0 0000:BC:00.0
0000:CB:00.0 0000:CC:00.0 0000:DB:00.0 0000:DC:00.0 0000:83:00.0 0000:84:00.0 0000:85:00.0 0000:86:00.0
--mmio_size 1280

# Remove 20 PCI devices if you want to do so
./vhpc_toolkit passthru --vm $VM --remove --device 0000:19:00.0 0000:1A:00.0 0000:3B:00.0 0000:3C:00.0
0000:4C:00.0 0000:4D:00.0 0000:5D:00.0 0000:5E:00.0 0000:9B:00.0 0000:9C:00.0 0000:BB:00.0 0000:BC:00.0
0000:CB:00.0 0000:CC:00.0 0000:DB:00.0 0000:DC:00.0 0000:83:00.0 0000:84:00.0 0000:85:00.0 0000:86:00.0
```

Figure 10: Batch operation on several VMs using vhpc-toolkit

VM advanced settings

You need to add four extra settings to the advanced settings of the VM.

Relaxing ACS settings for P2P between PCIe devices:

- pciPassthru.allowP2P = true
- pciPassthru.relaxACSforP2P = true

Required settings for VM power-on:

Without these settings, the VM with Passthru GPUs will not power on.

- pciPassthru.use64bitMMIO = TRUE
- pciPassthru.64bitMMIOSizeGB = 1056

For each passthrough H100 GPU, a 128GB MMIO size is necessary. Therefore, if you pass through 8 H100/H200 GPUs and 8 NICs to the VM, pciPassthru.64bitMMIOSizeGB must be set to 1056 or higher (calculated as 128\*8 + 32). It is important to minimize the MMIO size, as a very large MMIO size can extend the VM’s boot process by a few seconds. This delay is due to the increased time required for the system to allocate and configure the MMIO address space for virtualized devices.

Additional performance enhancement settings:

- Set a reservation on the CPU’s GHz.
- Configure NUMA settings. (VM Options > CPU topology > Cores per socket)
- Set Latency Sensitivity to High.

Deploy DLVM with vGPU

The second choice is to use vGPU to deploy DLVM, refer to “[Deploying a Deep Learning VM in VMware Private AI Foundation with NVIDIA](#).” Alternatively, for direct deployment on a vSphere cluster, consult “[Deploy a Deep Learning VM Directly on a vSphere Cluster in VMware Private AI Foundation with NVIDIA](#).” In this case, ensure you complete the “vGPU Driver Installation” section within the DLVM template deployment UI as follows.

| vGPU Driver Installation            |                                                                                                                                                                                                                                                                                            |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5 settings                          |                                                                                                                                                                                                                                                                                            |
| vGPU license                        | The content of vGPU license file, i.e. "client_configuration_token.tok". Required for the vGPU driver to acquire a license.                                                                                                                                                                |
| NGC API key                         | If provided, the instance downloads the vGPU driver from the NVIDIA NGC Catalog using this API key.                                                                                                                                                                                        |
| Custom URL for vGPU Driver          | Specify a custom URL for downloading the vGPU driver when the "NGC API key" is not provided (useful for air-gapped environments). If the "NGC API key" is provided, this URL is optional and allows the instance to download the vGPU driver using the specified NVIDIA NGC resource URLs. |
| vGPU host driver version (Optional) | Optional. If set, the instance would install a vGPU guest driver that matches the specified host driver version.                                                                                                                                                                           |
| NVIDIA Portal API key (Deprecated)  | If provided, the instance downloads the vGPU driver from the NVIDIA Licensing Portal using this API key when neither the "NGC API key" nor the "Custom URL for vGPU Driver" is specified.                                                                                                  |

Figure 11: vGPU Settings in DLVM deployment

## Performance

Before running performance tests in VCF, configure the TH5 switch, GPUs, and NICs in the DLVM as described in appendices B, C, and D to ensure the expected performance.

### Running RDMA with Perftest

#### Install Perftest with CUDA

These instructions apply to the recompiled perftest utility, which can be run directly from the home directory. They assume that the CUDA toolkit has been installed.

```
sudo apt update
sudo apt install libpci-dev

wget https://github.com/linux-rdma/perftest/releases/download/25.01.0-0.80/perftest-25.01.0-0.80.g6730e97.tar.gz
tar xf perftest-25.01.0-0.80.g6730e97.tar.gz
cd perftest-25.01.0/
./autogen.sh && ./configure CUDA_H_PATH=/usr/local/cuda/include/cuda.h && make -j$(nproc)

# Verify the following cuda flags have been are set up correctly
./ib_send_bw -h | grep cuda
--use_cuda=<cuda device id> Use CUDA specific device for GPUDirect RDMA testing
--use_cuda_bus_id=<cuda full BUS id> Use CUDA specific device, based on its full PCIe address, for
GPUDirect RDMA testing
--use_cuda_dmabuf Use CUDA DMA-BUF for GPUDirect RDMA testing
```

Figure 12: Install Perftest and Check setup correctly

Next, we conducted an RDMA performance test using Thor2 adapters between two VMs. The results showed Ethernet throughput of approximately 392 Gb/s, which corresponds to the maximum achievable RoCEv2 line rate with a 4K MTU. This value accounts for the protocol overhead introduced by Ethernet, IP, UDP, and RoCEv2 headers, so the measured rate represents either the theoretical peak or is within 1–2 Gb/s of it for this configuration.

**Note:** The following log reports GID Index = 3, indicating that the connection was established over Ethernet. This GID index will later be required when setting the NCCL environment variables for multi-GPU communication tests. For reference, when using IB, the GID index is 0.

```
# VM --> VM
# Start Server on Host1
./ib_send_bw -a -q 4 -F --report_gbits -d bnxt_re0

# Start Client on Host2
$ ./ib_send_bw -a -q 4 -F --report_gbits -d bnxt_re1 192.168.1.1

-----
Send BW Test
Dual-port      : OFF          Device      : bnxt_re1
Number of qps  : 4           Transport  type : IB
Connection type : RC          Using SRQ    : OFF
PCIe relax order: ON         Lock-free   : OFF
ibv_wr* API    : OFF         Using DDP    : 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 0x2c04 PSN 0x5c9264
GID: 00:00:00:00:00:00:00:00:255:255:192:168:01:02
local address: LID 0000 QPN 0x2c03 PSN 0x5e09b5
```

```

GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:02
local address: LID 0000 QPN 0x2c02 PSN 0x99ce25
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:02
local address: LID 0000 QPN 0x2c01 PSN 0x7e2fd9
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:02
remote address: LID 0000 QPN 0x2c04 PSN 0x102c5f
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
remote address: LID 0000 QPN 0x2c03 PSN 0x921b94
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
remote address: LID 0000 QPN 0x2c02 PSN 0x67c418
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
remote address: LID 0000 QPN 0x2c01 PSN 0x2c3a50
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
-----
#bytes      #iterations  BW peak[Gb/sec]  BW average[Gb/sec]  MsgRate[Mpps]
2           4000         0.075829         0.075499            4.718701
4           4000         0.15             0.15                4.761922
8           4000         0.31             0.31                4.784546
16          4000         0.61             0.61                4.745077
32          4000         1.22             1.22                4.765979
64          4000         2.46             2.45                4.792416
128         4000         4.90             4.89                4.779144
256         4000         9.73             9.73                4.749133
512         4000         19.65            19.60               4.786252
1024        4000         39.38            39.34               4.802630
2048        4000         77.83            77.75               4.745431
4096        4000         157.54           156.52              4.776593
8192        4000         281.88           281.87              4.301011
16384       4000         347.21           347.09              2.648107
32768       4000         386.64           386.47              1.474257
65536       4000         389.95           389.82              0.743519
131072      4000         391.11           391.09              0.372977
262144      4000         391.63           391.60              0.186729
524288      4000         391.86           391.85              0.093423
1048576     4000         392.01           392.00              0.046730
2097152     4000         392.08           392.08              0.023370
4194304     4000         392.11           392.11              0.011686
8388608     4000         392.13           392.13              0.005843
-----

```

Figure 13: RDMA test between 2 VMs across 2 hosts.

### Running GPUDirect RDMA with PerfTest

Next, we conducted the GDR test using the `ib_send_bw` test. This test was performed with a Thor2 NIC and an NVIDIA GPU on Host1 (NIC0), and a Thor2 NIC and GPU on Host2 (NIC0). For optimal peer-memory direct performance, it is recommended to use GPUs located closest to the NIC on the PCIe Bus.

A bandwidth of 202 Gb/s was observed, which is consistent with Thor2's lack of Address Translation Services (ATS) capability. ESX mandates an Input-Output Memory Management Unit (IOMMU), such as Intel VT-d or AMD I/O Virtualization (IOV), for DirectPath I/O devices due to security and convenience considerations. ATS is a performance-enhancing feature that enables a PCIe endpoint to execute DMA transactions directly with another PCIe endpoint when either IOMMU, ACS, or both are active. The Broadcom CSG team is currently developing a workaround for Thor2, and ATS is expected to be integrated into the forthcoming Thor3 release.

```
# VM --> VM
# Start Server on Host1
./ib_send_bw -a -q 4 -F --report_gbits -d bnxt_re0 --use_cuda=0

# Start Client on Host2
$ ./ib_send_bw -a -q 4 -F --report_gbits -d bnxt_re1 --use_cuda=1 192.168.1.1
Perftest doesn't supports CUDA tests with inline messages: inline size set to 0
initializing CUDA
Listing all CUDA devices in system:
CUDA device 0: PCIe address is 04:00
CUDA device 1: PCIe address is 04:02
CUDA device 2: PCIe address is 04:04
CUDA device 3: PCIe address is 04:06
CUDA device 4: PCIe address is 04:08
CUDA device 5: PCIe address is 04:0A
CUDA device 6: PCIe address is 04:0C
CUDA device 7: PCIe address is 04:0E

Picking device No. 1
[pid = 2489221, dev = 1] device name = [NVIDIA H100 80GB HBM3]
creating CUDA Ctx
making it the current CUDA Ctx
CUDA device integrated: 0
cuMemAlloc() of a 67108864 bytes GPU buffer
allocated GPU buffer address at 00007fce50000000 pointer=0x7fce50000000
-----
Send BW Test
Dual-port      : OFF          Device      : bnxt_re1
Number of qps  : 4           Transport   type : IB
Connection type : RC         Using SRQ    : OFF
PCIe relax order: ON         Lock-free    : OFF
ibv_wr* API    : OFF         Using DDP    : 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 0x2c04 PSN 0x482e3e
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:01
local address: LID 0000 QPN 0x2c03 PSN 0x36ce50
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:01
local address: LID 0000 QPN 0x2c02 PSN 0xe3395a
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:01
local address: LID 0000 QPN 0x2c01 PSN 0x64e6a1
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:01
remote address: LID 0000 QPN 0x2c04 PSN 0xbc115e
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
remote address: LID 0000 QPN 0x2c03 PSN 0x1d42f0
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
remote address: LID 0000 QPN 0x2c02 PSN 0x4aa17a
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
remote address: LID 0000 QPN 0x2c01 PSN 0x78fc41
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:01:01
-----
#bytes    #iterations    BW peak[Gb/sec]    BW average[Gb/sec]    MsgRate[Mpps]
2          4000            0.076923           0.075898               4.743634
4          4000            0.16               0.16                   4.855734
8          4000            0.31               0.30                   4.740087
16         4000            0.62               0.62                   4.862285
32         4000            1.25               1.24                   4.835015
64         4000            2.51               2.51                   4.894155
128        4000            5.00               4.99                   4.868493
256        4000            9.87               9.79                   4.780318
512        4000            19.98              19.95                  4.871167
1024       4000            40.16              40.09                  4.893700
2048       4000            79.34              79.25                  4.837061
4096       4000            154.93             154.80                 4.724255
8192       4000            181.54             181.30                 2.766474
```

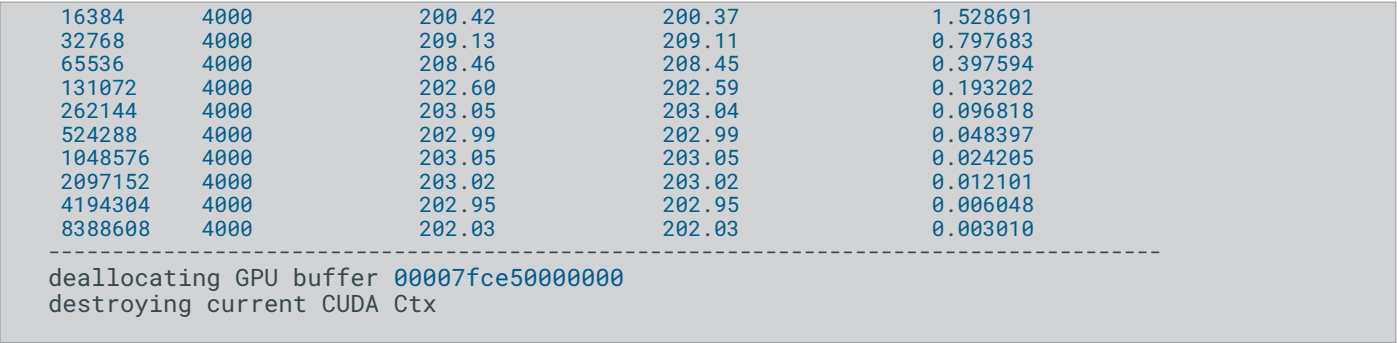


Figure 14: GDR test on 2 VMs across 2 hosts

Running NCCL Tests

NVIDIA Collective Communication Library (NCCL) is the central software component for enabling efficient multi-GPU workloads. It manages inter-GPU communication across PCIe, NVLink, InfiniBand, RoCE, and other high-speed interconnects. The figure below provides an overview of the software stack, from the AI application layer down to the hardware.

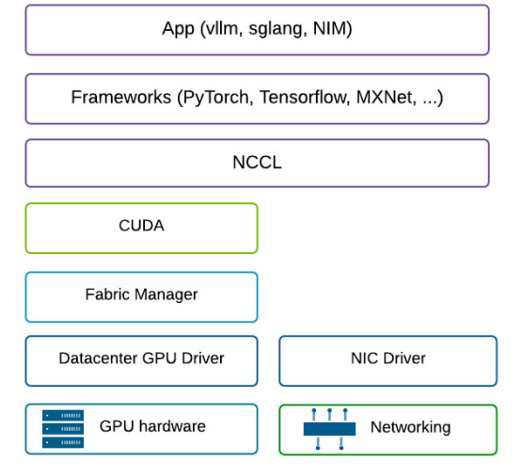


Figure 15: Software stacks of an AI application (from top to bottom)

As listed in Table 2 parallelism strategies used in LLM inference, fine-tuning, and training across multiple GPUs relies on particular NCCL collective operations. Therefore, it is crucial for NCCL to function reliably and efficiently within this environment.

| Parallism               | Collective Operations required                                                                                           |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Data Parallelism / FSDP | All-reduce, all-gather, reduce-scatter                                                                                   |
| Tensor Parallelism      | All-reduce, all-gather, reduce-scatter                                                                                   |
| Pipeline Parallelism    | Send / Receive                                                                                                           |
| Expert Parallelism      | All-to-all                                                                                                               |
| GDRCOPY git clone       | <a href="https://github.com/NVIDIA/gdrCOPY.git">https://github.com/NVIDIA/gdrCOPY.git</a> (current version GDR-Copy 2.5) |

Table 2: Parallelism and collective operations required

NCCL implements communication primitives as CUDA operations (kernels) which can be inserted into the computing workflow. For network communication, a CPU proxy thread will assist the GPU. NCCL is also open-source and available on Github.

Open MPI is necessary for running NCCL across multiple nodes in a cluster, primarily for launching NCCL tests and processes. UCX can also be used with Open MPI for this purpose. It's important to note that NCCL itself does not rely on Open MPI or UCX for its core operations.

Table 2 lists the specific software component versions used in this deployment. It is recommended to install and execute Open MPI, UCX, and nccl-tests (a sanity check and benchmark code for NCCL)s a non-root user. Passwordless SSH should also be configured for this non-root user.

| Component         | Version                                                                                                                         |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------|
| CUDA              | 12.8                                                                                                                            |
| UCX               | 1.18                                                                                                                            |
| Open MPI          | 5.0.8                                                                                                                           |
| NCCL              | 2.26.2                                                                                                                          |
| GDRCopy git clone | <a href="https://github.com/NVIDIA/gdrCOPY.git">https://github.com/NVIDIA/gdrCOPY.git</a> (current version GDR-CopyGDRCopy 2.5) |

### Setting up the environment variables

To set up environment variables, use the following commands:

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

### Install GDRCopy

[GDRCopy](#): A user-space library and kernel module that enables user processes to map GPU device memory directly into their CPU virtual address space

- Leverages NVIDIA GPUDirect RDMA (GDR) APIs.
- Allows the CPU to directly read/write GPU memory using standard `memcpy()` instructions, eliminating CUDA driver overhead.
- Ideal for small, frequent data transfers (for example, headers, control signals, synchronization buffers) where latency is critical.

GDRCopy significantly reduces latency for small transfers by bypassing CUDA's heavyweight `memcpy` path, improving performance from approximately 7–10  $\mu$ s to around 1  $\mu$ s.

```
wget https://github.com/NVIDIA/gdrCOPY/archive/refs/tags/v2.5.zip
unzip -q v2.5.zip

cd gdrCOPY-2.5/packages

sudo apt install build-essential devscripts debhelper fakeroot pkg-config dkms

# Install gdrCOPY

CUDA=/usr/local/cuda-12.8 ./build-deb-packages.sh

sudo dpkg -i gdrdrv-dkms_2.5-1_amd64.Ubuntu22_04.deb
sudo dpkg -i libgdrapi_2.5-1_amd64.Ubuntu22_04.deb
sudo dpkg -i gdrCOPY-tests_2.5-1_amd64.Ubuntu22_04+cuda12.8.deb
sudo dpkg -i gdrCOPY_2.5-1_amd64.Ubuntu22_04.deb
```

```
sudo dpkg -i gdrCOPY-tests_2.5-1_amd64.Ubuntu22_04+cuda12.4.deb

# Verify the GDR copy libraries are installed to dkms
ls /lib/modules/5.15.0-89-generic/updates/dkms/ | grep gdr
gdrdrv.ko

# Check gdrCOPY service
sudo systemctl status gdrCOPY

# Sanity Check
$ gdrCOPY_sanity
Total: 36, Passed: 31, Failed: 0, Waived: 5

List of waived tests:
  basic_v2_forcepci_cumemalloc
  basic_v2_forcepci_vmmalloc
  basic_with_tokens
  data_validation_mix_mappings_cumemalloc
  data_validation_v2_forcepci_cumemalloc

# Other gdrCOPY verification
gdrCOPY_copybw
gdrCOPY_copypat
gdrCOPY_apiperf -s 8
```

Figure 15: Install GDRCopy

### Installing UCX for NVIDIA GPUs

Follow these examples to install UCX for NVIDIA GPUs:

```
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

# Install to NFS
../contrib/configure-release --with-cuda=/usr/local/cuda --with-verbs --with-go=no --enable-
debug --prefix=/mnt/nfs/ucx-1.18.1/ucx_install

make -j ${nproc}
make -j ${nproc} install

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

Figure 16: Install UCX

### Installing Open MPI for NVIDIA GPUs

Use the following commands to install Open MPI for NVIDIA GPUs:

```
# The Open MPI build needs to point to the UCX installation $HOME/ucx/ucx_install as shown in the steps
below.
# On Ubuntu please install flex
sudo apt install flex

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

cd ompi
```

```
./autogen.pl
mkdir build
mkdir ompi_install
cd build
../configure --prefix=$HOME/ompi/ompi_install --with-cuda=/usr/local/cuda --with-cuda-lib-
dir=/usr/local/cuda/targets/x86_64-linux/lib/stubs --with-verbs --with-ucx=$HOME/ucx/ucx_in-
stall --enable-debug
make -j $(nproc)
make -j $(nproc) install

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

Figure 17: Install Open MPI

### Compiling NCCL tests

[nccl-tests](#) is used for both the correctness and performance of NCCL operations. It is available on GitHub and can be built by cloning the repository. This example assumes that CUDA and NCCL libraries are in their expected locations for the NCCL test. If CUDA is not installed in `/usr/local/cuda`, specify `CUDA_HOME` during the make process. Similarly, if NCCL is not in `/usr`, you can specify `NCCL_HOME`.

```
git clone https://github.com/NVIDIA/nccl-tests.git
cd nccl-tests
make -j $(nproc)

export OMPI_VER=5.0.8
make -j $(nproc) MPI=1 NAME_SUFFIX=_mpi MPI_HOME=/mnt/nfs/openmpi-${OMPI_VER}/ompi_install
```

Figure 18: Compiling nccl-tests

Once compiled, the following collectives can be tested:

- all\_gather
- all\_reduce
- all\_to\_all
- all\_to\_allv
- broadcast
- gather
- reduce
- reduce\_scatter
- scatter
- send\_recv



### Single node NCCL tests

Single-node NCCL collectives achieve an expected bus bandwidth of approximately 480 GB/s. This performance is observed because single-node operations do not utilize Thor2 and, by default, do not employ RoCE. This behavior persists even when Broadcom bnxt\_en and bnxt\_re drivers are removed.

```
# if using NCCL 2.27.x, set the following environment variable
export NCCL_TOPO_FILE=/mnt/nfs/vm_topo_bnxt_mod.xml
# if using earlier NCCL version, the above NCCL_TOPO_FILE is NOT required in NCCL-test in single node

# Light load test
./build/all_reduce_perf_mpi -b 8 -e 128M -f 2 -g 8

# Heavy load test
./build/all_reduce_perf_mpi -b 8 -e 16G -f 2 -g 8

# nThread 1 nGpus 8 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters:
1 validation: 1 graph: 0
#
# Using devices
# Rank 0 Group 0 Pid 582824 on pt02 device 0 [0000:04:00] NVIDIA H100 80GB HBM3
# Rank 1 Group 0 Pid 582824 on pt02 device 1 [0000:04:02] NVIDIA H100 80GB HBM3
# Rank 2 Group 0 Pid 582824 on pt02 device 2 [0000:04:04] NVIDIA H100 80GB HBM3
# Rank 3 Group 0 Pid 582824 on pt02 device 3 [0000:04:06] NVIDIA H100 80GB HBM3
# Rank 4 Group 0 Pid 582824 on pt02 device 4 [0000:04:08] NVIDIA H100 80GB HBM3
# Rank 5 Group 0 Pid 582824 on pt02 device 5 [0000:04:0a] NVIDIA H100 80GB HBM3
# Rank 6 Group 0 Pid 582824 on pt02 device 6 [0000:04:0c] NVIDIA H100 80GB HBM3
# Rank 7 Group 0 Pid 582824 on pt02 device 7 [0000:04:0e] NVIDIA H100 80GB HBM3

NCCL version 2.27.5+cuda12.9
#
#
# size count type redop root time out-of-place in-place
# (B) (elements) (us) (GB/s) busbw #wrong (us) (GB/s) busbw #wrong
#
8 2 float sum -1 41.49 0.00 0.00 0 41.44 0.00 0.00 0
16 4 float sum -1 40.58 0.00 0.00 0 40.07 0.00 0.00 0
32 8 float sum -1 59.54 0.00 0.00 0 41.69 0.00 0.00 0
64 16 float sum -1 42.15 0.00 0.00 0 41.65 0.00 0.00 0
128 32 float sum -1 42.89 0.00 0.01 0 41.52 0.00 0.01 0
256 64 float sum -1 42.16 0.01 0.01 0 41.96 0.01 0.01 0
512 128 float sum -1 42.55 0.01 0.02 0 42.31 0.01 0.02 0
1024 256 float sum -1 41.83 0.02 0.04 0 42.27 0.02 0.04 0
2048 512 float sum -1 41.95 0.05 0.09 0 43.16 0.05 0.08 0
4096 1024 float sum -1 41.99 0.10 0.17 0 43.77 0.09 0.16 0
8192 2048 float sum -1 42.32 0.19 0.34 0 42.03 0.19 0.34 0
16384 4096 float sum -1 42.94 0.38 0.67 0 42.11 0.39 0.68 0
32768 8192 float sum -1 42.01 0.78 1.37 0 41.79 0.78 1.37 0
65536 16384 float sum -1 42.50 1.54 2.70 0 42.68 1.54 2.69 0
131072 32768 float sum -1 43.58 3.01 5.26 0 43.24 3.03 5.30 0
262144 65536 float sum -1 44.35 5.91 10.34 0 45.15 5.81 10.16 0
524288 131072 float sum -1 47.49 11.04 19.32 0 47.08 11.14 19.49 0
1048576 262144 float sum -1 49.76 21.07 36.88 0 48.17 21.77 38.09 0
2097152 524288 float sum -1 47.34 44.30 77.53 0 48.26 43.45 76.05 0
4194304 1048576 float sum -1 55.75 75.23 131.66 0 55.56 75.49 132.10 0
8388608 2097152 float sum -1 84.85 98.86 173.01 0 83.89 99.99 174.98 0
16777216 4194304 float sum -1 125.7 133.46 233.56 0 125.2 134.00 234.50 0
33554432 8388608 float sum -1 201.2 166.75 291.81 0 201.6 166.44 291.27 0
67108864 16777216 float sum -1 325.8 205.96 360.43 0 326.1 205.81 360.16 0
134217728 33554432 float sum -1 588.4 228.13 399.22 0 587.6 228.43 399.76 0
268435456 67108864 float sum -1 1112.0 241.39 422.43 0 1111.9 241.41 422.47 0
536870912 134217728 float sum -1 2148.7 249.86 437.26 0 2146.1 250.16 437.79 0
1073741824 268435456 float sum -1 4018.0 267.23 467.66 0 4014.2 267.49 468.10 0
2147483648 536870912 float sum -1 7938.2 270.53 473.42 0 7935.9 270.60 473.56 0
4294967296 1073741824 float sum -1 15764 272.46 476.81 0 15786 272.07 476.12 0
8589934592 2147483648 float sum -1 31329 274.19 479.82 0 31363 273.89 479.31 0
17179869184 4294967296 float sum -1 62422 275.22 481.63 0 62400 275.32 481.80 0

# Out of bounds values : 0 OK
# Avg bus bandwidth : 155.78
```

Figure 19: Virtual NCCL performance in a single HGX host

**Multi-node NCCL tests**

Before proceeding, complete the following pre-checks:

- Network configuration:
  - Ensure NIC IPs and routings are properly configured as Appendix D.
  - Set up passwordless SSH between all hosts.
- NUMA balancing:
  - Disable NUMA balancing on every host. This can be done at runtime by running `echo 0 > /proc/sys/kernel/numa_balancing` or `sysctl -w kernel.numa_balancing=0`.
  - For persistent disablement after reboots, add `kernel.numa_balancing=0` to `/etc/sysctl.d/99-sysctl.conf`.
- Environment variables:
  - Verify that `PATH` and `LD_LIBRARY_PATH` are correctly set in the `.bashrc` file for each node used in the test.
  - Example configuration:
 

```
export LD_LIBRARY_PATH=$HOME/mpi/mpi_install/lib:$HOME/ucx/ucx_install/lib:$LD_LIBRARY_PATH
export PATH=$HOME/mpi/mpi_install/bin:$HOME/ucx/ucx_install/bin:$PATH
```
- NCCL settings:
  - Set `NCCL_TOPO_FILE=/mnt/nfs/vm_topo_bnxt_mod.xml`.
  - Set `NCCL_IB_GID_INDEX=3`.
- NCCL\_TOPO\_FILE Example: Please refer to Appendix E

An example `NCCL_TOPO_FILE` for a VM with 8 GPUs, 8 Thor2 NICs, and 4 NVSwitches is provided below. This file was generated by sequentially adding PCIe devices (GPU0, NIC0, GPU1, NIC1, ..., GPU7, NIC7) and then attaching all NVSwitches. The crucial aspect is the structural arrangement and ensuring that the SBDF values within the VM correspond to the physical devices, particularly in pairs of proximate GPUs and NICs

```
$ ./run_2nodes_nccl_allreduce_bnxt.sh
bnxt_en          3399680  1 bnxt_re
bnxt_re          503808  0
ib_cm            131072  1 rdma_cm
ib_core          393216  9 rdma_cm, rpcrdma, iw_cm, bnxt_re, ib_iser, ib_umad, rdma_ucm, ib_uverbs, ib_cm
ib_iser          49152   0
ib_umad          40960   0
ib_uverbs        163840  3 nvidia_peermem, bnxt_re, rdma_ucm
libahci          45056   1 ahci
libcrc32c        16384   5 nf_contrack, nf_nat, btrfs, nf_tables, raid456
libiscsi         69632   1 ib_iser
nvidia           11522048 97 nvidia_uvm, nvidia_peermem, gdrdrv, nvidia_modeset
nvidia_peermem   16384   0
rdma_cm          122880  3 rpcrdma, ib_iser, rdma_ucm
scsi_transport_iscsi 139264  2 ib_iser, libiscsi
tls              114688  1 bnxt_en
Executing 1: /mnt/nfs/openmpi-5.0.8/mpi_install/bin/mpirun --np 16 -H pt01:8,pt02:8 -x NCCL_TOPO_FILE=/mnt/nfs/vm_topo_bnxt_mod.xml -x NCCL_TOPO_DUMP_FILE=/tmp/topo_dump_file.txt -x NCCL_DEBUG=VERSION -x NCCL_DEBUG_SUBSYS=TUNING --mca pm1 ucx --mca btl_openib_warn_no_device_params_found 0 --mca btl ^openib -x NCCL_SOCKET_IFNAME=ens33 -x NCCL_NET_GDR_LEVEL=1 -x NCCL_P2P_LEVEL=NVL -x NCCL_IB_DISABLE=0 -x NCCL_IB_HCA=bnxt_re0:1,bnxt_re1:1,bnxt_re2:1,bnxt_re3:1,bnxt_re4:1,bnxt_re5:1,bnxt_re6:1,bnxt_re7:1 -x NCCL_IB_GID_INDEX=3 -x NCCL_P2P_NET_CHUNKSIZE=262144 /home/vmware/nccl-tests/build/all_reduce_perf_mpi -b 8 -e 16g -f 2 -g 1
Run-1 Starting command at 2025-07-31 06:18:10
[pt01:04729] SET NCCL_TOPO_FILE=/mnt/nfs/vm_topo_bnxt_mod.xml
[pt01:04729] SET NCCL_TOPO_DUMP_FILE=/tmp/topo_dump_file.txt
[pt01:04729] SET NCCL_DEBUG=VERSION
[pt01:04729] SET NCCL_DEBUG_SUBSYS=TUNING
[pt01:04729] SET NCCL_SOCKET_IFNAME=ens33
[pt01:04729] SET NCCL_NET_GDR_LEVEL=1
```

```

[pt01:04729] SET NCCL_P2P_LEVEL=NVL
[pt01:04729] SET NCCL_IB_DISABLE=0
[pt01:04729] SET 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
[pt01:04729] SET NCCL_IB_GID_INDEX=3
[pt01:04729] SET NCCL_P2P_NET_CHUNKSIZE=262144
# nThread 1 nGpus 1 minBytes 8 maxBytes 17179869184 step: 2(factor) warmup iters: 5 iters: 20 agg iters: 1 validation: 1 graph: 0
#
# Using devices
# Rank 0 Group 0 Pid 4801 on pt01 device 0 [0000:04:00] NVIDIA H100 80GB HBM3
# Rank 1 Group 0 Pid 4802 on pt01 device 1 [0000:04:02] NVIDIA H100 80GB HBM3
# Rank 2 Group 0 Pid 4803 on pt01 device 2 [0000:04:04] NVIDIA H100 80GB HBM3
# Rank 3 Group 0 Pid 4804 on pt01 device 3 [0000:04:06] NVIDIA H100 80GB HBM3
# Rank 4 Group 0 Pid 4805 on pt01 device 4 [0000:04:08] NVIDIA H100 80GB HBM3
# Rank 5 Group 0 Pid 4806 on pt01 device 5 [0000:04:0a] NVIDIA H100 80GB HBM3
# Rank 6 Group 0 Pid 4807 on pt01 device 6 [0000:04:0c] NVIDIA H100 80GB HBM3
# Rank 7 Group 0 Pid 4808 on pt01 device 7 [0000:04:0e] NVIDIA H100 80GB HBM3
# Rank 8 Group 0 Pid 4022 on pt02 device 0 [0000:04:00] NVIDIA H100 80GB HBM3
# Rank 9 Group 0 Pid 4025 on pt02 device 1 [0000:04:02] NVIDIA H100 80GB HBM3
# Rank 10 Group 0 Pid 4023 on pt02 device 2 [0000:04:04] NVIDIA H100 80GB HBM3
# Rank 11 Group 0 Pid 4024 on pt02 device 3 [0000:04:06] NVIDIA H100 80GB HBM3
# Rank 12 Group 0 Pid 4026 on pt02 device 4 [0000:04:08] NVIDIA H100 80GB HBM3
# Rank 13 Group 0 Pid 4027 on pt02 device 5 [0000:04:0a] NVIDIA H100 80GB HBM3
# Rank 14 Group 0 Pid 4028 on pt02 device 6 [0000:04:0c] NVIDIA H100 80GB HBM3
# Rank 15 Group 0 Pid 4029 on pt02 device 7 [0000:04:0e] NVIDIA H100 80GB HBM3

NCCL version 2.26.2+cuda12.8
#
#
# size count type redop root out-of-place in-place
# (B) (elements) (us) (GB/s) (GB/s) #wrong (us) (GB/s) (GB/s) #wrong
#
8 2 float sum -1 76.62 0.00 0.00 0 34.56 0.00 0.00 0
16 4 float sum -1 34.60 0.00 0.00 0 34.31 0.00 0.00 0
32 8 float sum -1 34.65 0.00 0.00 0 155.1 0.00 0.00 0
64 16 float sum -1 41.11 0.00 0.00 0 34.55 0.00 0.00 0
128 32 float sum -1 34.65 0.00 0.01 0 34.32 0.00 0.01 0
256 64 float sum -1 90.25 0.00 0.01 0 34.70 0.01 0.01 0
512 128 float sum -1 42.59 0.01 0.02 0 34.80 0.01 0.03 0
1024 256 float sum -1 62.85 0.02 0.03 0 35.51 0.03 0.05 0
2048 512 float sum -1 36.61 0.06 0.10 0 36.66 0.06 0.10 0
4096 1024 float sum -1 38.53 0.11 0.20 0 37.76 0.11 0.20 0
8192 2048 float sum -1 38.92 0.21 0.39 0 38.02 0.22 0.40 0
16384 4096 float sum -1 39.81 0.41 0.77 0 39.05 0.42 0.79 0
32768 8192 float sum -1 40.85 0.80 1.50 0 40.30 0.81 1.52 0
65536 16384 float sum -1 43.23 1.52 2.84 0 42.72 1.53 2.88 0
131072 32768 float sum -1 46.63 2.81 5.27 0 46.11 2.84 5.33 0
262144 65536 float sum -1 55.55 4.72 8.85 0 55.12 4.76 8.92 0
524288 131072 float sum -1 80.83 6.49 12.16 0 80.69 6.50 12.18 0
1048576 262144 float sum -1 84.50 12.41 23.27 0 83.96 12.49 23.42 0
2097152 524288 float sum -1 90.45 23.18 43.47 0 90.87 23.08 43.27 0
4194304 1048576 float sum -1 118.4 35.43 66.44 0 118.7 35.32 66.23 0
8388608 2097152 float sum -1 189.0 44.37 83.20 0 188.0 44.61 83.65 0
16777216 4194304 float sum -1 272.8 61.50 115.31 0 272.1 61.66 115.62 0
33554432 8388608 float sum -1 472.7 70.99 133.11 0 473.4 70.88 132.90 0
67108864 16777216 float sum -1 943.2 71.15 133.40 0 934.6 71.80 134.63 0
134217728 33554432 float sum -1 1668.6 80.44 150.82 0 1676.8 80.05 150.09 0
268435456 67108864 float sum -1 3047.1 88.10 165.18 0 3068.8 87.47 164.01 0
536870912 134217728 float sum -1 5818.1 92.28 173.02 0 5823.0 92.20 172.87 0
1073741824 268435456 float sum -1 11313 94.91 177.95 0 11304 94.99 178.11 0
2147483648 536870912 float sum -1 22459 95.62 179.28 0 22499 95.45 178.97 0
4294967296 1073741824 float sum -1 48019 89.44 167.71 0 48179 89.15 167.15 0
8589934592 2147483648 float sum -1 101990 84.22 157.92 0 101818 84.37 158.19 0
17179869184 4294967296 float sum -1 209830 81.88 153.52 0 209765 81.90 153.56 0
# Out of bounds values : 0 OK
# Avg bus bandwidth : 61.1074
#

Finished command at 2025-07-31 06:18:46
Elapsed time: 36 seconds

```

Figure 20: Virtual NCCL performance across two HGX hosts

## Running LLM benchmark in a single node

To evaluate server performance, we used models of increasing size, determined by the number of billions of parameters. This section begins with a performance comparison of Llama-3.1-70B NIM, served on 4 out of 8 GPUs and 8 NICs in a VM in DirectPath I/O mode, against the same server in a bare-metal configuration. Subsequently, we detail the use of vLLM to serve the GPT-OSS-120B open-source model and benchmark its performance using EvalScope.

### Llama-3.1-70b performance using NIM by GenAI-Perf

The following outlines the TensorRT-LLM-backed NIM launch scripts

```
# Export the NGC API key, then Docker login to NGC
export NGC_API_KEY=YOUR_KEY
echo "$NGC_API_KEY" | docker login nvcr.io --username '$oauthtoken' --password-stdin

# (Optional) List available NIMs. This step requires to install NGC CLI.
ngc registry image list --format_type csv nvcr.io/nim/*

# Choose a container name for bookkeeping
export CONTAINER_NAME=Llama-3.1-70b-instruct

# The container name from the previous ngc registry image list command
Repository=nim/meta/llama-3.1-70b-instruct
Tag=1.3.3

# Choose a LLM NIM Image from NGC
export IMG_NAME="nvcr.io/${Repository}:${Tag}"

# Choose a path on your system to cache the downloaded models
export LOCAL_NIM_CACHE=~/.cache/nim
mkdir -p "$LOCAL_NIM_CACHE"

# List the compatible profiles of the NIM
docker run --rm --runtime=nvidia -e NGC_API_KEY --gpus=all $IMG_NAME list-model-profiles

export PORT=8000
export NCCL_CUMEM_ENABLE=0
export NIM_MODEL_PROFILE=tensorrt_llm-h100-fp8-tp4-pp1-throughput

docker run -it --rm --name=$CONTAINER_NAME-$PORT \
  --runtime=nvidia \
  --gpus all \
  -e NGC_API_KEY \
  -v "$LOCAL_NIM_CACHE:/opt/nim/.cache" \
  -u $(id -u) \
  -p $PORT:8000 \
  -e NIM_MODEL_PROFILE \
  -e NCCL_CUMEM_ENABLE \
  --shm-size=16GB \
  $IMG_NAME

# For air-gapped environments, an additional environment variable is required in the Docker run command.
-e http_proxy="http://xxxxx:password@IP:port"
-e https_proxy="http://xxxxx:password@IP:port"
-e no_proxy="localhost,127.0.0.1,yyy"
```

Figure 21: Launch Llama-3.1-70b NIM

We will now set up a GenAI-Perf warm-up load test.

```
# Launch a triton inference server container with only CPU
export RELEASE="24.10"
docker run -it --rm --runtime=nvidia \
  --net=host \
  -v $(pwd):/workspace/host \
  nvcr.io/nvidia/tritonserver:${RELEASE}-py3-sdk \
  /bin/bash

# Log in with your Huggingface credential for accessing llama-3 tokenizer
pip install huggingface_hub
huggingface-cli login

# Run GenAI-Perf within triton inference server container for warm up
export INPUT_SEQUENCE_LENGTH=200
export INPUT_SEQUENCE_STD=0
export OUTPUT_SEQUENCE_LENGTH=50
export CONCURRENCY=10

genai-perf profile \
  -m meta/llama-3.1-70b-instruct \
  --endpoint-type chat \
  --service-kind openai \
  --streaming \
  -u localhost:8000 \
  --synthetic-input-tokens-mean $INPUT_SEQUENCE_LENGTH \
  --synthetic-input-tokens-stddev $INPUT_SEQUENCE_STD \
  --concurrency $CONCURRENCY \
  --output-tokens-mean $OUTPUT_SEQUENCE_LENGTH \
  --extra-inputs max_tokens:$OUTPUT_SEQUENCE_LENGTH \
  --extra-inputs min_tokens:$OUTPUT_SEQUENCE_LENGTH \
  --extra-inputs ignore_eos:true \
  --tokenizer meta-llama/Meta-Llama-3.1-70B-Instruct \
  -- \
  -v \
  --max-threads=256
```

Figure 22: Launch GenAI-Perf

#### Performance comparison: virtual vs bare metal

Experiments involved sweeping concurrency levels from 1 to 100. Results for larger workloads were obtained using a pair of Input Sequence Length (ISL) and Output Sequence Length (OSL), representing a generation chatbot (ISL=500, OSL=2000). The numbers on the points in the performance graphs indicate the number of concurrent users at each point.

Table 4 outlines the hardware configurations for both bare-metal and virtualized systems. A key difference is the use of DPIO in the virtualized setup.

| Factor             | Bare metal                    | Virtual                                   |
|--------------------|-------------------------------|-------------------------------------------|
| Logical processors | 208                           | 24 allocated to the VM                    |
| GPUs               | 8x NVIDIA H100-SXM-80GB       | Passthru 8x NVIDIA H100 GPU & 4x NVSwitch |
| Memory             | 2TB                           | 32GB (1.97 TB for other VMs/workloads)    |
| Storage            | 8x 3.84TB NVMe SSD            | 256GB VM Hard Disk on vSAN                |
| OS                 | Ubuntu 22.04.3                | Ubuntu 22.04.03 DLVM in VCF 9.0           |
| GPU drivers        | Data Center Driver 550.144.03 | Data Center Driver 550.144.03             |
| CUDA               | 12.4                          | 12.4                                      |

| Factor       | Bare metal                                                                                                                                                | Virtual                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| NIM          | <ul style="list-style-type: none"><li>• llama-3.1-70b-instruct</li><li>• Tag: 1.3.3</li><li>• profile: tensorrt_llm-h100-fp8-tp4-pp1-throughput</li></ul> | <ul style="list-style-type: none"><li>• llama-3.1-70b-instruct</li><li>• Tag: 1.3.3</li><li>• profile: tensorrt_llm-h100-fp8-tp4-pp1-throughput</li></ul> |
| tritonserver | 24.10-py3-sdk                                                                                                                                             | 24.10-py3-sdk                                                                                                                                             |
| GenAI-Perf   | 0.0.11                                                                                                                                                    | 0.0.11                                                                                                                                                    |

To ensure a fair comparison, NIM with Tensor Parallelism using 4 H100 GPUs is launched. In both the bare metal and virtual environments, only 4 out of 8 GPUs are utilized for benchmarking. On GenAI perf, online mode is used by setting `--streaming` to benchmark the online application. The throughput (output tokens per second) for the generation chatbot is then plotted against the Time to First Token (TTFT) in Figure 23.

**Note:** The performance results presented in the following two subsections reflect observed results under specific conditions. Identical performance cannot be guaranteed, as it may vary depending on your hardware, software configuration, or workload. These results are also subject to be improved with future releases or updates to the software stack.

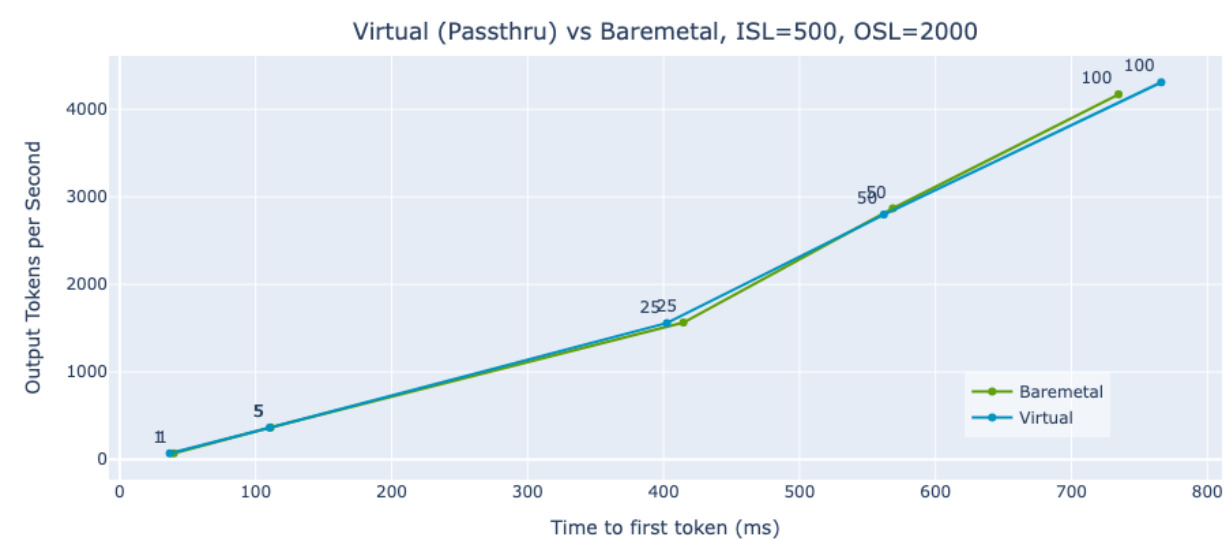


Figure 23: Throughput and TTFT: virtual (passthrough) vs bare metal across concurrency levels (1~100).

**gpt-oss-120b performance using vLLM**

OpenAI's state-of-the-art open-weight models, known as [gpt-oss](#), is also able to be served with full context length in the DLVM. These models are specifically designed for robust reasoning, agentic tasks, and a wide array of developer applications.

```

pip install -U "huggingface_hub[cli]"
huggingface-cli login

# Use UV (a fast package control tool) to install the latest vllm
uv venv gpt-oss --python 3.12 --seed && source gpt-oss/bin/activate
uv pip install --pre vllm==0.10.1+gptoss --extra-index-url https://wheels.vllm.ai/gpt-oss/ --extra-index-url https://download.pytorch.org/whl/nightly/cu128 --index-strategy unsafe-best-match

# Use vmware customized NCCL topology file to launch vllm
export NCCL_TOPO_FILE=vm_topo_bnxt_mod.xml
# Launch by 8x H100-SXM using tensor_parallelism = 8
vllm serve openai/gpt-oss-120b --served-model-name gpt-oss-120b --port 8801 --tensor-parallel-size 8
--gpu_memory_utilization 0.96

```

**Figure 24:** Launch gpt-oss-120b using vllm

Upon successful launch, the following log will confirm that the model has been initialized with the maximum available context length (128K tokens), estimating a maximum concurrency of approximately 96 for serving a 128K context window

```

(VllmWorker TP6 pid=837629) INFO 08-08 14:44:56 [gpu_worker.py:275] Available KV cache memory: 57.24 GiB
(EngineCore_0 pid=837466) INFO 08-08 14:44:56 [kv_cache_utils.py:993] GPU KV cache size: 6,679,504 tokens
(EngineCore_0 pid=837466) INFO 08-08 14:44:56 [kv_cache_utils.py:997] Maximum concurrency for 131,072 tokens per request: 95.83x

```

**Figure 25:** vllm launch logs

We can now benchmark it using EvalScope, another open-source tool.

```

# Install evalscope
uv venv evalscope --python 3.10 && source evalscope/bin/activate
uv pip install 'evalscope[perf]' -U
uv pip install 'evalscope[app]' -U

# Use evalscope to conduct stress test benchmark
evalscope perf \
  --parallel 1 5 10 20 50 100 200 400 800 \
  --number 5 10 20 40 100 200 400 800 1600 \
  --model gpt-oss-120b \
  --url http://127.0.0.1:8801/v1/completions \
  --api openai \
  --dataset random \
  --max-tokens 1024 \
  --min-tokens 1024 \
  --prefix-length 0 \
  --min-prompt-length 1024 \
  --max-prompt-length 1024 \
  --log-every-n-query 20 \
  --tokenizer-path ~/.cache/huggingface/hub/models--openai--gpt-oss-120b/snapshots/eabf0c518da7584a2e7d-ab4ab272709785a72126/ \
  --extra-args '{"ignore_eos": true}'

```

**Figure 26:** Scripts to use evalscope to conduct stress test benchmark gpt-oss-120b



Benchmark results, using a random dataset, show that a concurrency of 1200 yields the highest performance, achieving 13.2 requests/sec and a throughput of 10217 tokens/sec.

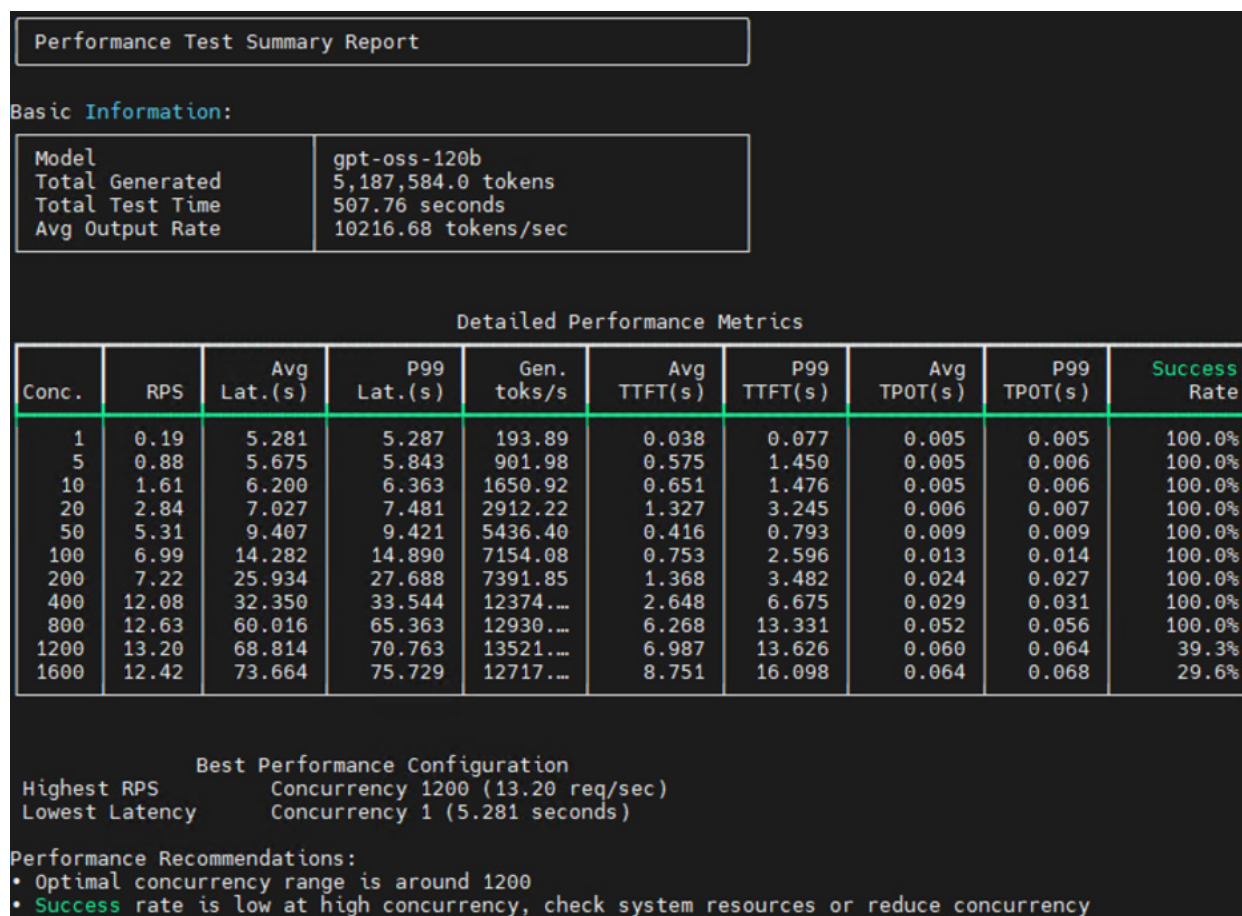


Figure 27: Benchmark gpt-oss-120b with Evalslope



EvalScope provides accuracy benchmarks for various datasets, such as AIME25, and can also incorporate user-defined datasets.

```
# Download aime25 dataset
hf download --repo-type dataset opencompass/AIME2025 --local-dir ./AIME2025_dataset

# Use evalscope to conduct accuracy benchmark
# Create a run_aime25.py
from evalscope.constants import EvalType
from evalscope import TaskConfig, run_task

task_cfg = TaskConfig(
    model='gpt-oss-120b', # Model name
    api_url='http://127.0.0.1:8801/v1', # Model service address
    eval_type=EvalType.SERVICE, # Evaluation type, here using service evaluation
    datasets=['aime25'], # Dataset to test
    dataset_args={
        'aime25': {
            'local_path': '/home/vmware/evalscope/gpt-oss-scripts/AIME2025_dataset' # Use
            local dataset path
        }
    },
    generation_config={
        'extra_body': {"reasoning_effort": "high"} # Model generation parameters, set to
        high reasoning level
    },
    eval_batch_size=10, # Concurrent batch size
    timeout=60000, # Timeout in seconds
)

run_task(task_cfg=task_cfg)

# Conduct accuray benchmark
python run_aime25.py

# Use Evalscope UI to view the results in a user-friendly way
evalscope app --lang en
```

Figure 28: Scripts to use evalscope to evaluate accuracy benchmark on gpt-oss-120b

We can see that gpt-oss-120b reaches 0.9333 score for AIME2025-I and 1 for AIME2025-II test.

```
/home/vmware/python-3.10/AIME2025_dataset report table:
+-----+-----+-----+-----+-----+-----+-----+
| Model | Dataset | Metric | Subset | Num | Score | Cat.0 |
+-----+-----+-----+-----+-----+-----+-----+
| gpt-oss-120b | aime25 | AveragePass@1 | AIME2025-I | 15 | 0.9333 | default |
+-----+-----+-----+-----+-----+-----+-----+
| gpt-oss-120b | aime25 | AveragePass@1 | AIME2025-II | 15 | 1 | default |
+-----+-----+-----+-----+-----+-----+-----+
| gpt-oss-120b | aime25 | AveragePass@1 | OVERALL | 30 | 0.9667 | - |
+-----+-----+-----+-----+-----+-----+-----+

2025-08-07 18:44:29,742 - evalscope - INFO - Skipping report analysis (`analysis_report=False`).
2025-08-07 18:44:29,742 - evalscope - INFO - Dump report to: ./outputs/20250807_183029/reports/gpt-oss-120b/aime25.json

2025-08-07 18:44:29,742 - evalscope - INFO - Evaluation finished on /home/vmware/python-3.10/AIME2025_dataset
2025-08-07 18:44:29,745 - evalscope - INFO - Overall report table:
+-----+-----+-----+-----+-----+-----+-----+
| Model | Dataset | Metric | Subset | Num | Score | Cat.0 |
+-----+-----+-----+-----+-----+-----+-----+
| gpt-oss-120b | aime25 | AveragePass@1 | AIME2025-I | 15 | 0.9333 | default |
+-----+-----+-----+-----+-----+-----+-----+
| gpt-oss-120b | aime25 | AveragePass@1 | AIME2025-II | 15 | 1 | default |
+-----+-----+-----+-----+-----+-----+-----+
| gpt-oss-120b | aime25 | AveragePass@1 | OVERALL | 30 | 0.9667 | - |
+-----+-----+-----+-----+-----+-----+-----+

2025-08-07 18:44:29,745 - evalscope - INFO - Finished evaluation for gpt-oss-120b on ['aime25']
2025-08-07 18:44:29,745 - evalscope - INFO - Output directory: ./outputs/20250807_183029
```

Figure 29: Accuracy evaluation results of gpt-oss-120b using AIME25 datasets

## Inference sizing guidance

Additional sizing guidance can refer to: [LLM Inference Sizing and Performance Guidance - VMware Cloud Foundation \(VCF\) Blog](#)

## Conclusion

This deployment guide has provided a comprehensive walkthrough for deploying VMware Private AI on HGX servers using Broadcom Ethernet networking, specifically focusing on Thor2 NICs and a Tomahawk 5-based Ethernet switch for LLM inference workloads. We detailed the necessary configurations, from BIOS and ESX settings to a VM setup with DirectPath I/O for GPUs and Thor2 NICs. Key networking aspects, including the single-leaf switch topology with 24-bit subnets and optimized QoS settings on the SONiC OS, were thoroughly covered to ensure efficient data transfer for AI workloads. Furthermore, the guide outlined the installation and sanity checks for NVIDIA GPU drivers, Fabric Manager, and Thor2 drivers, culminating in the execution of various performance benchmarks such as RDMA with Perftest, NCCL collectives, and LLM inference tests using NIM and vLLM. By following the detailed steps and best practices presented, users can successfully establish a robust and high-performance environment for AI inference within a VCF-based private cloud infrastructure leveraging Broadcom's networking hardware.

Ready to get started on your AI and ML journey? Check out these helpful resources:

- [Complete this form](#) to contact us!
- Read the [VMware Private AI solution brief](#).
- Learn more about [VMware Private AI](#).
- Connect with us on Twitter at [@VMwareVCF](#) and on LinkedIn at [VMware VCF](#).

## Appendix

### A: Firmware update

#### Atlas2 PCIe Switch Board (PSB) firmware

For the latest PSB firmware updates, please contact your vendor. This document is based on version 4.160.3.0 and above. A new firmware update will be released soon; please install the most recent version when available.

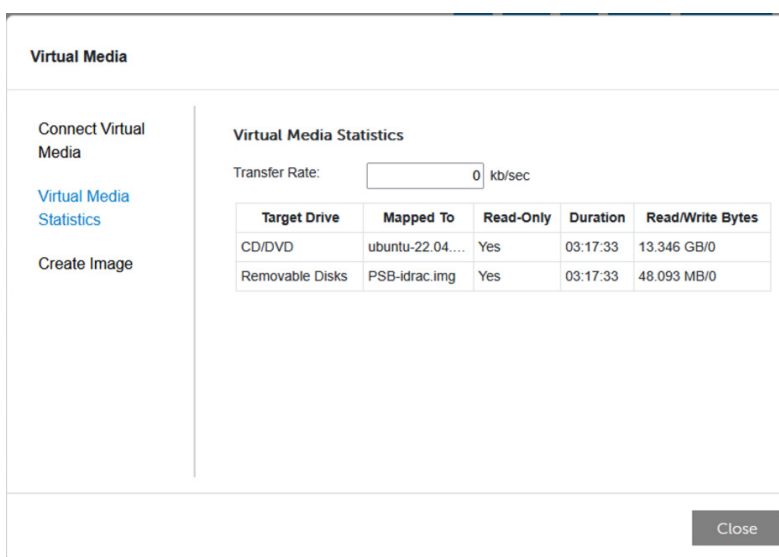
#### For servers with dual-system installations

Reboot to bare metal and update the PSB using the command:

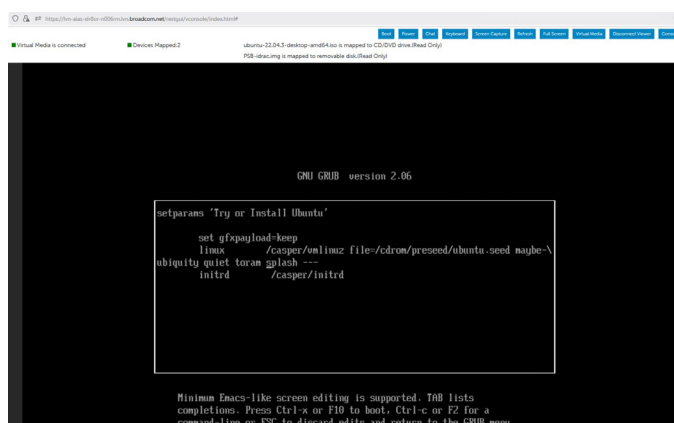
```
chmod +x Firmware_xxx.BIN
sudo ./Firmware_xxx.BIN
```

#### For servers with only ESX installed:

1. To update the PSB's firmware, you will need to boot into an Ubuntu live-ISO. This process may take approximately 30 minutes to load into RAM.



2. Select Try or install Ubuntu.



3. Once logged into the live-ISO, run the same bare metal commands to upgrade the PSB's firmware (this will take around 3 minutes). You can check the logs to verify which firmware version has been installed. Reboot the host normally into ESXi.

```
ubuntu@ubuntu: ~  
[ 18.965139] mpt3sas_cm7: "PEX89144" : FWVersion(04.103.00.00), ChipRevision(0xb0), BiosVersion(0  
0.00.00.00)  
ubuntu@ubuntu:~$ chmod +x Firmware_FGPJP_LN64_4.160.3.0_A00.BIN  
ubuntu@ubuntu:~$ sudo ./Firmware_FGPJP_LN64_4.160.3.0_A00.BIN  
Collecting inventory...  
..  
Running validation...  
Update Package is valid for this system  
Device: PowerEdge XE9680 PSB Firmware  
The version of this Update Package is greater than the currently installed version.  
Software application name: Fryer U.2 (SW1)  
Package version: 4.160.3.0  
Installed version: 4.103.0.6  
Device: PowerEdge XE9680 PSB Firmware  
The version of this Update Package is greater than the currently installed version.  
Software application name: Fryer U.2 (SW2)  
Package version: 4.160.3.0  
Installed version: 4.103.0.6  
Device: PowerEdge XE9680 PSB Firmware  
The version of this Update Package is greater than the currently installed version.  
Software application name: Fryer U.2 (SW3)  
Package version: 4.160.3.0  
Installed version: 4.103.0.6  
Device: PowerEdge XE9680 PSB Firmware  
The version of this Update Package is greater than the currently installed version.  
Software application name: Fryer U.2 (SW4)  
Package version: 4.160.3.0  
Installed version: 4.103.0.6  
Requires Power Cycle for activating the updated firmware
```

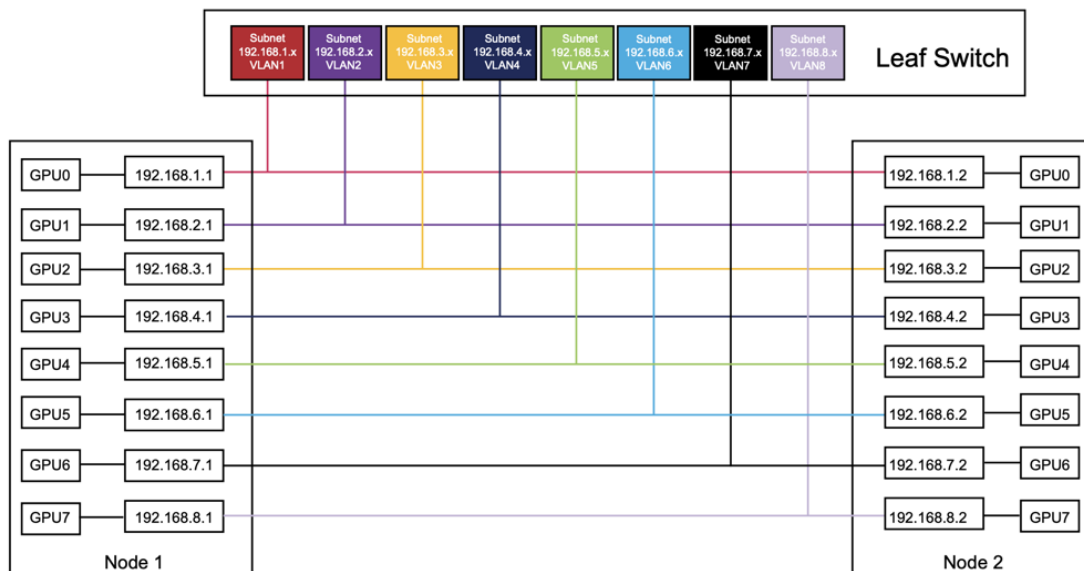
4. Reboot the host normally into ESX.

## B: Configure Tomahawk5 ethernet switch

### Single-leaf switch topology with 24-bit subnets

This deployment uses a Layer 3 switch and 24-bit IP subnets, with a topology where multiple hosts connect to the same leaf switch. The configuration details are as follows:

1. Each host is equipped with 8 NICs and 8 GPUs.
2. The leaf switch functions as a Layer 3 switch, capable of routing packets based on IPv4 addresses.
3. The network employs 8 IP subnets and 8 VLANs. VLANs are exclusively configured on switch ports, not on the NICs.
4. Each VLAN on the switch is assigned an IP address from a unique subnet within the set of eight used in the network.
5. Inter-VLAN routing is enabled on the switch.
6. Each of the 8 NICs on a host belongs to a unique subnet from the network's eight subnets. For symmetrical connectivity, NIC1 on every host connects to a switch port in the same VLAN, NIC2 on every host connects to a switch port in the same VLAN, and so on.
7. Source-based routing is applied to each NIC. The default gateway for each NIC is set to the IP address of the switch port VLAN to which it is connected. This setup requires eight additional routing tables (numbered 101 through 108).



**Figure :** 24-bit subnet scheme

There is an example of the TH5 switch configuration for the 24-bit subnet scheme listed in the appendix.

Example: 24-bit subnet scheme on SONiC OS

The Ethernet switch is configured with VLANs, Priority Flow Control (PFC), and Data Center Quantized Congestion Notification (DCQC-N) to optimize network performance, especially for NVIDIA H100 GPU clusters. Refer to Section 2.5 of the Broadcom 957608 - Ethernet Networking Guide for NVIDIA H100 GPU Clusters (<https://docs.broadcom.com/doc/957608-AN3XX>) for detailed information on these settings, which are crucial for low latency and high throughput in demanding GPU cluster applications.

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

```

!
queue 3
    type dwrr
    weight 50
!
queue 4
    type dwrr
    weight 50
!
queue 6
    type strict
!
!
!
qos map dscp-tc ROCE
    dscp 0-3,5-23,25,27-47,49-63 traffic-class 0
    dscp 24,26 traffic-class 3
    dscp 4 traffic-class 4
    dscp 48 traffic-class 6
!
qos map dot1p-tc ROCE
    dot1p 0-2,5-7 traffic-class 0
    dot1p 3 traffic-class 3
    dot1p 4 traffic-class 4
!
qos map tc-queue ROCE
    traffic-class 0 queue 0
    traffic-class 1 queue 1
    traffic-class 2 queue 2
    traffic-class 3 queue 3
    traffic-class 4 queue 4
    traffic-class 5 queue 5
    traffic-class 6 queue 6
    traffic-class 7 queue 7
!
qos map tc-pg ROCE
    traffic-class 3 priority-group 3
    traffic-class 4 priority-group 4
    traffic-class 0-2,5-7 priority-group 7
!
qos map pfc-priority-queue ROCE
    pfc-priority 0 queue 0
    pfc-priority 1 queue 1
    pfc-priority 2 queue 2
    pfc-priority 3 queue 3
    pfc-priority 4 queue 4
    pfc-priority 5 queue 5
    pfc-priority 6 queue 6
    pfc-priority 7 queue 7
!
# Setting each port
interface Ethernet 0
    description "Node1 NIC1"
    mtu 9100
    speed 400000
    fec RS
    standalone-link-training
    unreliable-los auto
    no shutdown
    switchport access Vlan 5
    queue 3 wred-policy ROCE
    scheduler-policy ROCE
    qos-map dscp-tc ROCE
    qos-map dot1p-tc ROCE
    qos-map tc-queue ROCE
    qos-map tc-pg ROCE
    qos-map pfc-priority-queue ROCE
    priority-flow-control priority 3
    priority-flow-control priority 4
    priority-flow-control watchdog action drop
    priority-flow-control watchdog on detect-time 200
    priority-flow-control watchdog restore-time 400
!
interface Ethernet 8

```

```

description "Node1 NIC2"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 6
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 16
description "Node1 NIC3"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 7
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 24
description "Node1 NIC4"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 8
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 32
description "Node1 NIC5"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 4

```

```

queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 40
description "Node1 NIC6"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 3
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 48
description "Node1 NIC7"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 1
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 56
description "Node1 NIC8"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 2
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE

```



```

priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 64
  description "Node2 NIC1"
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  unreliable-los auto
  no shutdown
  switchport access Vlan 5
  queue 3 wred-policy ROCE
  scheduler-policy ROCE
  qos-map dscp-tc ROCE
  qos-map dot1p-tc ROCE
  qos-map tc-queue ROCE
  qos-map tc-pg ROCE
  qos-map pfc-priority-queue ROCE
  priority-flow-control priority 3
  priority-flow-control priority 4
  priority-flow-control watchdog action drop
  priority-flow-control watchdog on detect-time 200
  priority-flow-control watchdog restore-time 400
!
interface Ethernet 72
  description "Node2 NIC2"
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  unreliable-los auto
  no shutdown
  switchport access Vlan 6
  queue 3 wred-policy ROCE
  scheduler-policy ROCE
  qos-map dscp-tc ROCE
  qos-map dot1p-tc ROCE
  qos-map tc-queue ROCE
  qos-map tc-pg ROCE
  qos-map pfc-priority-queue ROCE
  priority-flow-control priority 3
  priority-flow-control priority 4
  priority-flow-control watchdog action drop
  priority-flow-control watchdog on detect-time 200
  priority-flow-control watchdog restore-time 400
!
interface Ethernet 80
  description "Node2 NIC3"
  mtu 9100
  speed 400000
  fec RS
  standalone-link-training
  unreliable-los auto
  no shutdown
  switchport access Vlan 7
  queue 3 wred-policy ROCE
  scheduler-policy ROCE
  qos-map dscp-tc ROCE
  qos-map dot1p-tc ROCE
  qos-map tc-queue ROCE
  qos-map tc-pg ROCE
  qos-map pfc-priority-queue ROCE
  priority-flow-control priority 3
  priority-flow-control priority 4
  priority-flow-control watchdog action drop
  priority-flow-control watchdog on detect-time 200
  priority-flow-control watchdog restore-time 400
!
interface Ethernet 88

```

```

description "Node2 NIC4"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 8
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 96
description "Node2 NIC5"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 4
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 104
description "Node2 NIC6"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 3
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 112
description "Node2 NIC7"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown

```

```

switchport access Vlan 1
queue 3 wred-policy ROCE
scheduler-policy ROCE
qos-map dscp-tc ROCE
qos-map dot1p-tc ROCE
qos-map tc-queue ROCE
qos-map tc-pg ROCE
qos-map pfc-priority-queue ROCE
priority-flow-control priority 3
priority-flow-control priority 4
priority-flow-control watchdog action drop
priority-flow-control watchdog on detect-time 200
priority-flow-control watchdog restore-time 400
!
interface Ethernet 120
description "Node2 NIC8"
mtu 9100
speed 400000
fec RS
standalone-link-training
unreliable-los auto
no shutdown
switchport access Vlan 2
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
!
# Setting Vlan
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
!

```

For a 31-bit scheme, please refer to <https://docs.broadcom.com/docs/957608-AN3XX>.

C: Configure GPU and Thor2 in VM

NVIDIA GPU

NVIDIA GPU driver installation

The following details installing the NVIDIA GPU driver to enable GPU functionality.

- 1. Access CUDA Toolkit Archive:  
Go to the official NVIDIA CUDA Toolkit Archive: <https://developer.nvidia.com/cuda-toolkit-archive>.
- 2. Select CUDA Version:  
Choose a CUDA version compatible with your hardware (e.g., CUDA 12.9.1).

|                  |              |               |                 |        |         |            |
|------------------|--------------|---------------|-----------------|--------|---------|------------|
| Operating System | Linux        | Windows       |                 |        |         |            |
| Architecture     | x86_64       | arm64-sbsa    | aarch64-jetson  |        |         |            |
| Distribution     | Amazon-Linux | Azure-Linux   | Debian          | Fedora | KylinOS | OpenSUSE   |
|                  | Oracle-Linux | RHEL          | Rocky           | SLES   | Ubuntu  | WSL-Ubuntu |
| Version          | 12           |               |                 |        |         |            |
| Installer Type   | deb (local)  | deb (network) | runfile (local) |        |         |            |

- 3. Download the CUDA runfile:  
Download the corresponding self-extracting runfile. For CUDA 12.9.1, use:

```
wget https://developer.download.nvidia.com/compute/cuda/12.9.1/local_installers/cuda_12.9.1_575.57.08_linux.run
```

- 4. Initiate installation:  
Execute the runfile with superuser privileges:

```
sudo sh cuda_12.9.1_575.57.08_linux.run
```

5. Navigate the installation UI:

During installation, select the **Open Kernel module type** under **Options > Driver Options** and acknowledge the MIT license.

```
CUDA Installer
- [X] Driver
  [X] 575.57.08
+ [X] CUDA Toolkit 12.9
  [X] CUDA Demo Suite 12.9
  [X] CUDA Documentation 12.9
- [ ] Kernel Objects
  [ ] nvidia-fs
Options
Install

Up/Down: Move | Left/Right: Expand | 'Enter': Select | 'A': Advanced options
```

6. Consider NVIDIA GPUDirect Storage (GDS) and nvidia-fs:

Decide whether to install **nvidia-fs** based on your GDS needs. Install it if you plan to use GDS for direct data transfer between storage and GPU memory; otherwise, you can skip it.

```
CUDA Driver
[ ] Do not install any of the OpenGL-related driver files
[ ] Do not install the nvidia-drm kernel module
[ ] Update the system X config file to use the NVIDIA X driver
- [X] Override kernel module type
  [ ] proprietary
  [X] open
Change directory containing the kernel source files
Change kernel object output directory
Done

Up/Down: Move | Left/Right: Expand | 'Enter': Select | 'A': Advanced options
```

## NVIDIA Fabric Manager installation

The Fabric Manager (FM) package is required for servers with an NVswitch that connects multiple NVLink-capable GPUs. This package installs the necessary core components and registers the `nvidia-fabricmanager` daemon as a system service, which enables NVLink peer-to-peer support by registering the GPUs with the NVLink fabric.

```
# Based on the previous GPU driver version, install associated fabric manager
apt info nvidia-fabricmanager-575 -a

sudo apt install nvidia-fabricmanager-575=575.57.08-1
sudo apt install nvidia-fabricmanager-dev-575=575.57.08-1

# Enable fabric manager to start up when VM is powered on
sudo systemctl enable nvidia-fabricmanager
sudo systemctl start nvidia-fabricmanager
sudo systemctl status nvidia-fabricmanager

# Check training status and wait, if showing "In progress", then wait until, the state of FM is working
as "Completed"
nvidia-smi -q -i 0 | grep -i -A 2 Fabric

Fabric
      State                : Completed
      Status               : Success

# Troubleshooting: in some rare cases, you might run into the "Masked" situation
systemctl status nvidia-fabricmanager.service
Loaded: masked (Reason: Unit nvidia-fabricmanager.service is masked.)
Active: inactive (dead)

# Unmask FM services
sudo systemctl unmask nvidia-fabricmanager.service

# If there are some incompatible packages installed, use apt purge command
sudo apt purge nvidia-fabricmanager-570
sudo apt purge nvidia-fabricmanager-dev-570
```

## Sanity check for GPU management and FM service

After installing the GPU driver and Fabric Manager (FM) service, run `nvidia-smi` to confirm functionality. This command verifies GPU recognition and FM's NVLink facilitation. A successful output will list GPUs, display driver and CUDA versions, show GPU utilization, and critically, include NV indicators (for example, NV18 for 18 NVLinks), confirming active NVLink interconnects. Errors or missing info suggest installation or hardware issues, requiring troubleshooting.

```
$ nvidia-smi

$ nvidia-smi nvlink -s

$ nvidia-smi topo -m
```

|      | GPU0 | GPU1 | GPU2 | GPU3 | GPU4 | GPU5 | GPU6 | GPU7 | CPU Affinity | NUMA Affinity | GPU NUMA ID |
|------|------|------|------|------|------|------|------|------|--------------|---------------|-------------|
| GPU0 | X    | NV18 | NV18 | NV18 | NV18 | NV18 | NV18 | NV18 | 0-15 0       | N/A           |             |
| GPU1 | NV18 | X    | NV18 | NV18 | NV18 | NV18 | NV18 | NV18 | 0-15 0       | N/A           |             |
| GPU2 | NV18 | NV18 | X    | NV18 | NV18 | NV18 | NV18 | NV18 | 0-15 0       | N/A           |             |
| GPU3 | NV18 | NV18 | NV18 | X    | NV18 | NV18 | NV18 | NV18 | 0-15 0       | N/A           |             |
| GPU4 | NV18 | NV18 | NV18 | NV18 | X    | NV18 | NV18 | NV18 | 16-31 1      | N/A           |             |
| GPU5 | NV18 | NV18 | NV18 | NV18 | NV18 | X    | NV18 | NV18 | 16-31 1      | N/A           |             |
| GPU6 | NV18 | NV18 | NV18 | NV18 | NV18 | NV18 | X    | NV18 | 16-31 1      | N/A           |             |
| GPU7 | NV18 | NV18 | NV18 | NV18 | NV18 | NV18 | NV18 | X    | 16-31 1      | N/A           |             |

```
Legend:
X      = Self
SYS    = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
NODE   = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
PHB    = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
PXB    = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
PIX    = Connection traversing at most a single PCIe bridge
NV#    = Connection traversing a bonded set of # NVLinks
```

### Sanity checking CUDA by testing cuda-sample

To confirm proper CUDA library function and integration, compile and validate unit tests from `cuda-samples`. Focus on `deviceQuery`, `cudaTensorCoreGemm`, `nbody`, and `p2pbw` for a robust assessment.

- `deviceQuery` verifies CUDA driver installation and communication with hardware, and runtime API accessibility, providing GPU details.
- `cudaTensorCoreGemm` validates Tensor Core functionality and performance, ensuring their proper utilization and expected throughput for deep learning.
- `nbody` assesses GPU computational power and memory bandwidth under sustained load, confirming CUDA environment stability for complex simulations.
- `p2pbw` measures inter-GPU data transfer rates for multi-GPU systems, confirming NVLink/PCIe direct access performance.

Systematic compilation and validation of these samples ensure CUDA installation integrity, identify configuration issues, and optimize CUDA library function before deploying complex applications.

```
# Prerequisite
sudo apt-get install cmake freeglut3-dev libfreeimage-dev libgl1-mesa-dev libglu1-mesa-dev libvulkan-dev

# Download
wget https://github.com/NVIDIA/cuda-samples/archive/refs/tags/v12.9.zip
unzip -q v12.9.zip

# Compile & build
mkdir build && cd build
cmake ..
make -j $(nproc)
# compile will require 100% cpu usage

# Conduct deviceQuery sanity test
~/cuda-samples-12.9/build$ ./Samples/1_Uutilities/deviceQuery/deviceQuery

# Troubleshooting: if showing the below
cudaGetDeviceCount returned 802
-> system not yet initialized
Result = FAIL

# Then reboot the system typically due to the fabric manager not being loaded

# Run cudaTensorCoreGemm
~/cuda-samples-12.9$ ./build/Samples/3_CUDA_Features/cudaTensorCoreGemm/cudaTensorCoreGemm

# Run nbody test to see
vmware@pt02:~/cuda-samples-12.9$
./build/Samples/5_Domain_Specific/nbody/nbody -benchmark -numdevices=8 -numbodies=2097152

# Run p2pbw test to check communication bandwidth between each GPUs
./Samples/5_Domain_Specific/p2pBandwidthLatencyTest/p2pBandwidthLatencyTes
```

### Install NCCL

We recommend you download NCCL from the [NCCL legacy downloads page](#), ensuring that the chosen version matches your installed CUDA runtime. While building from source is possible, the recommended approach is to install precompiled packages via apt.

Note: This requires to have an NVIDIA developer account.

```
# Add NVIDIA package repository
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb
sudo apt-get update

# Install NCCL (adjust version to match your CUDA runtime)
sudo apt install libncccl2=2.27.5-1+cuda12.9 libncccl-dev=2.27.5-1+cuda12.9

# Verify NCCL installation
sudo apt install plocate
sudo updatedb
locate nccl | grep "libncccl.so" | tail -n1 | sed -r 's/^\.*\.so\.//'

# (optional) Uninstall NCCL
sudo apt purge libncccl*
```

### Troubleshooting NCCL in DLVM

Debugging NCCL issues in a Deep Learning Virtual Machine (DLVM) involves compiling a debug version of the NCCL source code with debugging symbols. Then, use a debugger like GDB to examine execution flow, memory, and variables. Set breakpoints and step through code to pinpoint the root cause of issues such as incorrect configurations or memory corruption. After identification, report issues to the NCCL team on GitHub.

<https://github.com/NVIDIA/nccl-tests/issues> or <https://github.com/NVIDIA/nccl/issues>

```
# Download NCCL source code
git clone https://github.com/NVIDIA/nccl.git
# Check out a specific version of NCCL
cd nccl && git checkout v2.27.5-1
# Keep debug symbols
export DEB_BUILD_OPTIONS=nostrip
# Build for all GPU architecture
make -j src.build
# Build only for Hopper series
make -j $(nproc) src.build NVCC_GENCODE="-gencode=arch=compute_90,code=sm_90" DEBUG=1
# Install tools to create Debian packages
sudo apt install build-essential devscripts debhelper fakeroot
# Build NCCL deb package
make -j $(nproc) pkg.debian.build
ls build/pkg/deb/
sudo dpkg -i *.deb
# Then build your code (for example, nccl-test) with this debug version of NCCL, then use gdb to debug
gdb -q --args ./build/all_reduce_perf_mpi -b 8 -e 128M -f 2 -g 8
(gdb) set environment NCCL_DEBUG=INFO
(gdb) set environment NCCL_TOPO_FILE=/mnt/nfs/vm_topo_mod.xml
(gdb) run
(gdb) bt
(gdb) frame #1
# When you find the specific line of code that causing issues
# If you get "system not yet initialized" errors, please make sure cuda-sample code is working correctly.
# If you get a segfault at the path discovery phase, this typically means you need to use a customized
NCCL topology file in the virtual environment.
```



### Thor2 (BCM57608 / BCM957608) NICs

#### Thor2 driver installation and sanity check

Broadcom offers a tarball with each NIC GA release. This tarball includes all necessary software and firmware for every NIC part number, along with an automated installer.

The latest GA release tarball is publicly accessible via the **Downloads** tab on the following links:

- <https://www.broadcom.com/products/ethernet-connectivity/network-adapters/p1400g>
- <https://www.broadcom.com/products/ethernet-connectivity/network-adapters/p2200g>

For any GA release not available publicly, customers can contact Broadcom directly.

The following are the commands to install Thor2 drivers and sanity checks afterwards.

```
# Install necessary prerequisites
sudo apt update
sudo apt install rdma-core infiniband-diags rdmacm-utils librdmacm-dev librdmacm1 ibverbs-utils linux-modules-extra-$(uname -r)

# Check existing rdma naming rules
ls /usr/lib/udev/rules.d/60-rdma-persistent-naming.rules
# Replace to from NAME_FALLBACK to NAME_KERNEL

# Download the latest Thor2 NIC driver
# The latest drivers can be found at: https://www.broadcom.com/products/ethernet-connectivity/network-adapters/p1400g
wget https://docs.broadcom.com/docs-and-downloads/ethernet-network-adapters/NXE/BRCM_234.1.124.0/bcm_234.1.124.0b.tar.gz

tar xf

# Install Thor2 drivers
sudo bash ./install.sh -v -i ens65np0 -i ens67np0 -i ens69np0 -i ens71np0 -i ens73np0 -i ens75np0 -i ens77np0 -i ens79np0 -f -g

# 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
[ 20.495025] bnxt_en: module verification failed: signature and/or required key missing - tainting kernel
[ 21.359611] Broadcom NetXtreme-C/E/S driver bnxt_en v1.10.3-234.0.154.0
[ 25.020937] bnxt_re: Broadcom NetXtreme-C/E RoCE Driver bnxt_re v234.0.154.0 (June 23, 2025)

# Check bnxt_en
modinfo bnxt_en | grep -i ver
version: 1.10.3-234.0.154.0
description: Broadcom NetXtreme-C/E/S network driver
srcversion: 2B187EDBD374C6754A04C9E
vermagic: 5.15.0-143-generic SMP mod_unload modversions

# Check bnxt_re
modinfo bnxt_re | grep -i ver
version: 234.0.154.0
description: Broadcom NetXtreme RoCE Driver
srcversion: D50FE839BAA36E25869FCFB
depends: ib_core,ib_uverbs,bnxt_en,mlx_compat
vermagic: 5.15.0-143-generic SMP mod_unload modversions

# Check ib_peer_mem
modinfo ib_peer_mem | grep -i ver
modinfo: ERROR: Module ib_peer_mem not found.
# If nvidia-peermem is already installed nvidia-peermem, then "ib_peer_mem not found" is expected.
```

```
# Check nvidia-peermem
modinfo nvidia-peermem | grep -i ver
version:      570.133.20
srcversion:    5FABF5F7C8B8936A5196441
depends:       nvidia,ib_uverbs
vermagic:     5.15.0-143-generic SMP mod_unload modversions

# Check dkms
dkms status
gdrdrv/2.5, 5.15.0-142-generic, x86_64: installed
gdrdrv/2.5, 5.15.0-143-generic, x86_64: installed
netxtreme-peer-mem/234.0.154.0, 5.15.0-142-generic, x86_64: installed (Differences between built and in-
stalled modules)
netxtreme-peer-mem/234.0.154.0, 5.15.0-143-generic, x86_64: installed (Original modules exist)
nvidia/570.133.20, 5.15.0-142-generic, x86_64: installed
nvidia/570.133.20, 5.15.0-143-generic, x86_64: installed

# Check existing modules
sudo lsmod | grep -E 'nvidia|ib|bnxt'
nvidia_drm                110592      0
nvidia_modeset            1708032    1 nvidia_drm
nvidia_peermem            16384      0
nvidia                    11522048   96 nvidia_peermem,gdrdrv,nvidia_modeset
libcrc32c                 16384      5 nf_conntrack,nf_nat,btrfs,nf_tables,raid456
bnxt_re                   507904      0
ib_uverbs                 163840     2 nvidia_peermem,bnxt_re
ib_core                   417792     2 bnxt_re,ib_uverbs
mlx_compat                69632      3 bnxt_re,ib_core,ib_uverbs
drm_kms_helper            315392     5 mgag200,nvidia_drm,nouveau
bnxt_en                   3399680     1 bnxt_re
drm                       622592     8 drm_kms_helper,nvidia,mgag200,drm_ttm_helper,nvidia_drm,ttm,nouveau
tls                       114688     1 bnxt_en
libahci                   49152      1 ahci

# The firmware version running on the card can be checked via the Broadcom-provided NIC CLI tool.
sudo niccli -d
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----

1 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens65np0
   MAC Address          : 8C:84:74:5A:4D:20
   PCI Address          : 0000:04:01.0

2 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens67np0
   MAC Address          : 8C:84:74:5A:44:F0
   PCI Address          : 0000:04:03.0

3 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens69np0
   MAC Address          : 8C:84:74:5A:4B:10
   PCI Address          : 0000:04:05.0

4 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens71np0
   MAC Address          : 8C:84:74:5A:0F:90
   PCI Address          : 0000:04:07.0

5 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens73np0
   MAC Address          : 8C:84:74:5A:27:80
   PCI Address          : 0000:04:09.0

6 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens75np0
   MAC Address          : 8C:84:74:5A:38:E0
   PCI Address          : 0000:04:0B.0

7 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens77np0
   MAC Address          : 8C:84:74:5A:69:50
   PCI Address          : 0000:04:0D.0

8 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : ens79np0
   MAC Address          : 8C:84:74:5A:44:E0
   PCI Address          : 0000:04:0F.0
```

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

```
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----
```

|    | BoardId(Rev) | MAC Address       | FwVersion   | PCIAddr      | Type | Mode |
|----|--------------|-------------------|-------------|--------------|------|------|
| 1) | BCM57608(B1) | 8C:84:74:5A:4D:20 | 234.0.145.0 | 0000:04:01.0 | NIC  | PCI  |
| 2) | BCM57608(B1) | 8C:84:74:5A:44:F0 | 234.0.145.0 | 0000:04:03.0 | NIC  | PCI  |
| 3) | BCM57608(B1) | 8C:84:74:5A:4B:10 | 234.0.145.0 | 0000:04:05.0 | NIC  | PCI  |
| 4) | BCM57608(B1) | 8C:84:74:5A:0F:90 | 234.0.145.0 | 0000:04:07.0 | NIC  | PCI  |
| 5) | BCM57608(B1) | 8C:84:74:5A:27:80 | 234.0.145.0 | 0000:04:09.0 | NIC  | PCI  |
| 6) | BCM57608(B1) | 8C:84:74:5A:38:E0 | 234.0.145.0 | 0000:04:0B.0 | NIC  | PCI  |
| 7) | BCM57608(B1) | 8C:84:74:5A:69:50 | 234.0.145.0 | 0000:04:0D.0 | NIC  | PCI  |
| 8) | BCM57608(B1) | 8C:84:74:5A:44:E0 | 234.0.145.0 | 0000:04:0F.0 | NIC  | PCI  |

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

```
sudo niccli -i 1 show --pkg_ver
```

```
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----
```

```
Package Information :
Active Package Version      : 234.1.124.0
Package Version on NVM      : 234.1.124.0
Primary SBI Version         : 232.1.193.0
Secondary SBI Version       : 232.1.193.0
Primary SRT Version         : 234.0.145.0
Secondary SRT Version       : 234.0.145.0
Primary CRT Version         : 234.0.145.0
Secondary CRT Version       : 234.0.145.0
```

```
# Check cable
sudo niccli -i 1 cable -m --show
```

```
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----
```

```
Identifier                  : 0x18 (QSFP-DD Double Density 8X Pluggable Transceiver (INF-
8628))
Power class                  : 1
Max power                   : 0.25W
Connector                   : 0x23 (No separable connector)
Cable assembly length       : 3.50m
Transmitter technology       : 0x0a (Copper cable, unequalized)
Attenuation at 5GHz         : 10db
Attenuation at 7GHz         : 15db
Attenuation at 12.9GHz      : 19db
Attenuation at 25.8GHz      : 0db
Vendor name                 : Amphenol
Vendor PN                   : NDYYYH-0001
Vendor rev                  : K
Vendor SN                   : APF224601042W1
Date code                   : 221115
Vendor OUI                  : 78:a7:14
Revision compliance         : Rev. 5.0
Module State                : 0x03 (ModuleReady)
LowPwrAllowRequestHW        : Off
LowPwrRequestSW             : Off
```

```
# Check Link status
for interface in enp26s0np0 enp60s0np0 enp77s0np0 enp94s0np0 enp156s0np0 enp188s0np0 enp204s0np0
enp220s0np0; do sudo ethtool $interface | grep Link; done
```

```
[sudo] password for vmware:
Link partner advertised link modes: Not reported
Link partner advertised pause frame use: Symmetric
Link partner advertised auto-negotiation: No
Link partner advertised FEC modes: Not reported
Link detected: yes
...
```

**Verify the correct RoCE QOS configuration, RDMA options, and PCIe relaxed ordering on NICs**

To confirm the correct installation of the RoCE QOS CFG pkg (bnxt\_re\_conf) after driver installation, check for the existence of the `/etc/bnxt_re/bnxt_re.conf` file. The QoS values configured in this file should align with the default values.

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

# sudo niccli --help qos

# sudo niccli -i 1 qos --ets --show
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----

IEEE 8021QAZ ETS Configuration TLV:
  PRIQ_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%

# Loop on 8x Thor2 NICs
for i in {1..8}; do sudo niccli -i $i qos --ets --show; done

# List the priority to traffic class mapping
sudo niccli -i 1 qos --listmap --pri2cos
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----

Base Queue is 0 for port 0
-----
Priority  TC  HW Queue ID
-----
0         0    4
1         0    4
2         0    4
3         1    0
4         0    4
5         0    4
6         0    4
7         2    5
RoCE Bi-Di Optimization : Disabled
```

```
# Loop on 8x
for i in {1..8}; do sudo niccli -i $i qos --listmap --pri2cos; done

# -D|--dscp2prio : query the dscp to priority mapping.
sudo niccli -i 1 qos --dscp2prio

-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----
dscp2prio mapping:

        priority:3 dscp:26
        priority:7 dscp:48
# Loop on 8x
for i in {1..8}; do sudo niccli -i $i qos --dscp2prio; done

Next, we will examine the RDMA option, RoCE performance profiles, and the enablement of PCIe relaxed ordering on the NICs.
# RDMA Option on the NIC
# get
sudo niccli -i 1 nvm -g support_rdma --scope 0
-----
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
-----
support_rdma = True
# loop all NICs
for i in {1..8}; do sudo niccli -i $i nvm -g support_rdma --scope 0; done
# set
sudo niccli -i 1 nvm -s support_rdma --scope 0 --value 1

# RoCE performance profile on the NIC
# get
sudo niccli -i 1 nvm -g performance_profile
performance_profile = RoCE
# loop all NICs
for i in {1..8}; do sudo niccli -i $i nvm -g performance_profile; done
# set
sudo niccli -i 1 nvm -s performance_profile --value 1

# Enable PCIe relaxed ordering on the NIC
# get
sudo niccli -i 1 nvm -g pcie_relaxed_ordering
pcie_relaxed_ordering = Disabled
# loop all NICs
for i in {1..8}; do sudo niccli -i $i nvm -g pcie_relaxed_ordering; done
# set
sudo niccli -i 1 nvm -s performance_profile --value 1
```

### Verifying DCBx NVM CFG on NICs

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 LLDP should be disabled if the RoCE driver needs to configure the QoS for the interface

```
# Loop on 8x Thor2 NICs to check these option values
for i in {1..8}; do sudo niccli -i $i nvm -g dcbx_mode --scope 0; done

dcbx_mode = Disabled

for i in {1..8}; do sudo niccli -i $i nvm -g lldp_nearest_bridge --scope 0; done

lldp_nearest_bridge = Disabled

for i in {1..8}; do sudo niccli -i $i nvm -g lldp_nearest_non_tpmr_bridge --scope 0; done

lldp_nearest_non_tpmr_bridge = Disabled
```

## D: Configure routing for the backend network

This section outlines a sample netplan file for two hosts within a cluster, using a single Leaf TH5 switch configuration for a 24-bit subnet network (as shown in earlier section). A similar addressing scheme can be applied to other hosts.

The netplan file includes 8 Ethernet interfaces, labeled eth1 through eth8 for illustrative purposes. These should be replaced with the actual Ethernet interface names on the host. Each interface is assigned an IP address with a 24-bit subnet mask. For more complex networks, such as a 31-bit subnet network, refer to section 2.4.2 in the [BCM957608 - Ethernet Networking Guide for NVIDIA H100 GPU Clusters](#).

### Host1 netplan file

/etc/netplan/00-installer-config-24bit-subnet-host1.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens65np0:
      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
    ens67np0:
      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
    ens69np0:
      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
    ens71np0:
      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
    ens73np0:
      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
ens75np0:
  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
ens77np0:
  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
ens79np0:
  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

```

Next, apply the netplan files and perform a sanity check.

```

# Apply netplan host file
sudo netplan try    # to test the configuration
sudo netplan apply  # to apply the configuration permanently

# 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 routing tables should be populated

```

(base) vmware@pt01:~$ ip rule show
0:      from all lookup local
32758:  from 192.168.4.2 lookup 104 proto static
32759:  from 192.168.7.2 lookup 107 proto static
32760:  from 192.168.6.2 lookup 106 proto static
32761:  from 192.168.1.2 lookup 101 proto static
32762:  from 192.168.2.2 lookup 102 proto static
32763:  from 192.168.3.2 lookup 103 proto static
32764:  from 192.168.5.2 lookup 105 proto static
32765:  from 192.168.8.2 lookup 108 proto static
32766:  from all lookup main
32767:  from all lookup default

(base) vmware@pt01:~$ ip route list table 101
default via 192.168.1.254 dev ens65np0 proto static

```

```
(base) vmware@pt01:~$ ip route list table 102
default via 192.168.2.254 dev ens67np0 proto static
(base) vmware@pt01:~$ ip route list table 103
default via 192.168.3.254 dev ens69np0 proto static
(base) vmware@pt01:~$ ip route list table 104
default via 192.168.4.254 dev ens71np0 proto static
(base) vmware@pt01:~$ ip route list table 105
default via 192.168.5.254 dev ens73np0 proto static
(base) vmware@pt01:~$ ip route list table 106
default via 192.168.6.254 dev ens75np0 proto static
(base) vmware@pt01:~$ ip route list table 107
default via 192.168.7.254 dev ens77np0 proto static
(base) vmware@pt01:~$ ip route list table 108
default via 192.168.8.254 dev ens79 proto static
```

### Host2 netplan file

/etc/netplan/00-installer-config-24bit-subnet-host2.yaml

```
network:
  version: 2
  renderer: networkd
  ethernet:
    ens65np0:
      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
    ens67np0:
      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
    ens69np0:
      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
    ens71np0:
      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
    ens73np0:
      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
ens75np0:
  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
ens77np0:
  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
ens79np0:
  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

```

#### Confirm routing between different NICs across different hosts

Once routing is configured in the backend network, as detailed in “Configuring Host Routing for the Backend Network,” you can use ping and traceroute to confirm reachability among all NICs on every host.

```

#!/bin/bash

host1_ip_list=(192.168.1.1 192.168.2.1 192.168.3.1 192.168.4.1 192.168.5.1 192.168.6.1 192.168.7.1
192.168.8.1)

host2_ip_list=(192.168.1.2 192.168.2.2 192.168.3.2 192.168.4.2 192.168.5.2 192.168.6.2 192.168.7.2
192.168.8.2)

for i in ${host1_ip_list[@]}
do
  for j in ${host2_ip_list[@]}
  do
    echo -e "\nping -I $i $j\n===== \n"
    ping -I $i -c 1 $j > /dev/null
    if [ $? == 0 ]; then
      echo "Ping Passed"
    else
      echo -e "Ping Failed\n\n\n"
      exit 5
    fi
  done
done

```

## E: Customized NCCL\_TOPO\_FILE in VM

An example `NCCL_TOPO_FILE` for a VM with 8 GPUs, 8 Thor2 NICs, and 4 NVSwitches is provided below. This file is suited for a VM that sequentially adds PCIe devices (GPU0, NIC0, GPU1, NIC1, ..., GPU7, NIC7), then adds all NVSwitches. The structural arrangement is crucial and you must ensure that the SBDF values within the VM correspond to the physical devices, particularly in pairs of the GPUs and NICs under the same PCIe switch.

```
<system version="1">
  <cpu host_hash="0xabd045be9cccc099" numaid="0" affinity=
ty="5555,55555555,55555555,55555555,55555555,55555555,55555555,55555555" arch="x86_64" vendor="GenuineIntel" fami-
lyid="6" modelid="143">
    <pci busid="0000:16:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2330" link_speed="32.0 GT/s PCIe" link_width="16">
      <pci busid="0000:04:00.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
        <gpu dev="0" sm="90" rank="0" gdr="1">
          <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
          <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
          <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
          <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
        </gpu>
      </pci>
    <pci busid="0000:04:01.0" class="0x020000" vvendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
      <nic>
        <net name="bnxt_re0" dev="0" latency="0" speed="400000" port="1" guid="0x204d5afeff74848e" max-
conn="131073" gdr="1"/>
      </nic>
    </pci>
  </pci>
  <pci busid="0000:38:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2330" link_speed="32.0 GT/s PCIe" link_width="16">
    <pci busid="0000:04:03.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
      <nic>
        <net name="bnxt_re1" dev="1" latency="0" speed="400000" port="1" guid="0xf0445afeff74848e" max-
conn="131073" gdr="1"/>
      </nic>
    </pci>
  <pci busid="0000:04:02.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
    <gpu dev="1" sm="90" rank="1" gdr="1">
      <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
      <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
      <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
      <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
    </gpu>
  </pci>
  <pci busid="0000:49:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2331" link_speed="32.0 GT/s PCIe" link_width="16">
    <pci busid="0000:04:05.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
      <nic>
        <net name="bnxt_re2" dev="2" latency="0" speed="400000" port="1" guid="0x104b5afeff74848e" max-
conn="131073" gdr="1"/>
      </nic>
    </pci>
  <pci busid="0000:04:04.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
    <gpu dev="2" sm="90" rank="2" gdr="1">
      <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
      <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
      <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
      <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
    </gpu>
  </pci>
  <pci busid="0000:5a:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2331" link_speed="32.0 GT/s PCIe" link_width="16">
```

```

    <pci busid="0000:04:07.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
    <nic>
    <net name="bnxt_re3" dev="3" latency="0" speed="400000" port="1" guid="0x900f5afeff74848e" max-
conn="131073" gdr="1"/>
    </nic>
  </pci>
  <pci busid="0000:04:06.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
    <gpu dev="3" sm="90" rank="3" gdr="1">
    <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
    <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
    </gpu>
  </pci>
</pci>
</cpu>
<cpu host_hash="0xabd045be9cccc099" numaid="1" affinity="aaaa,aaaaaaaa,aaaaaaaa,aaaaaaaa,aaaaaaaa,aaaaa
aaa,aaaaaaaa" arch="x86_64" vendor="GenuineIntel" familyid="6" modelid="143">
  <pci busid="0000:98:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2332" link_speed="32.0 GT/s PCIe" link_width="16">
    <pci busid="0000:04:09.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
    <nic>
    <net name="bnxt_re4" dev="4" latency="0" speed="400000" port="1" guid="0x80275afeff74848e" max-
conn="131073" gdr="1"/>
    </nic>
  </pci>
  <pci busid="0000:04:08.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
    <gpu dev="4" sm="90" rank="4" gdr="1">
    <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
    <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
    </gpu>
  </pci>
</pci>
  <pci busid="0000:b8:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2332" link_speed="32.0 GT/s PCIe" link_width="16">
    <pci busid="0000:04:0b.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
    <nic>
    <net name="bnxt_re5" dev="5" latency="0" speed="400000" port="1" guid="0xe0385afeff74848e" max-
conn="131073" gdr="1"/>
    </nic>
  </pci>
  <pci busid="0000:04:0a.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
    <gpu dev="5" sm="90" rank="5" gdr="1">
    <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
    <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
    </gpu>
  </pci>
</pci>
  <pci busid="0000:c8:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2332" link_speed="32.0 GT/s PCIe" link_width="16">
    <pci busid="0000:04:0d.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
    <nic>
    <net name="bnxt_re6" dev="6" latency="0" speed="400000" port="1" guid="0x50695afeff74848e" max-
conn="131073" gdr="1"/>
    </nic>
  </pci>
  <pci busid="0000:04:0c.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
    <gpu dev="6" sm="90" rank="6" gdr="1">
    <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
    <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>

```

```

        <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
    </gpu>
</pci>
</pci>
<pci busid="0000:d8:00.0" class="0x060400" vendor="0x1000" device="0xc030" subsystem_vendor="0x1028"
subsystem_device="0x2332" link_speed="32.0 GT/s PCIe" link_width="16">
    <pci busid="0000:04:0f.0" class="0x020000" vendor="0x14e4" device="0x1760" subsystem_
vvendor="0x14e4" subsystem_device="0x9140" link_speed="32.0 GT/s PCIe" link_width="16">
        <nic>
            <net name="bnxt_re7" dev="7" latency="0" speed="400000" port="1" guid="0xe0445afeff74848e" max-
conn="131073" gdr="1"/>
        </nic>
    </pci>
    <pci busid="0000:04:0e.0" class="0x030200" vendor="0x10de" device="0x2330" subsystem_
vendor="0x10de" subsystem_device="0x16c1" link_speed="32.0 GT/s PCIe" link_width="16">
        <gpu dev="7" sm="90" rank="7" gdr="1">
            <nvlink target="0000:04:12.0" count="5" tclass="0x068000"/>
            <nvlink target="0000:04:11.0" count="5" tclass="0x068000"/>
            <nvlink target="0000:04:10.0" count="4" tclass="0x068000"/>
            <nvlink target="0000:04:13.0" count="4" tclass="0x068000"/>
        </gpu>
    </pci>
</pci>
</cpu>
</system>

```

## F: Terminology

| Term   | Definition                                                    |
|--------|---------------------------------------------------------------|
| ACS    | PCIe Access Control Service                                   |
| BTL    | Byte Transfer Layer                                           |
| CNI    | Container Network Interface                                   |
| DCBX   | Data Center Bridging Exchange protocol                        |
| DCQCN  | Data Centre quantizedCenter Quantized Congestion Notification |
| DLVM   | Deep Learning Virtual Machines                                |
| DKMS   | Dynamic Kernel Module Support                                 |
| DRS    | Distributed Resource Scheduler                                |
| DSCP   | Differentiated Services Code Point                            |
| DSM    | Data Service Manager                                          |
| ESA    | Express Storage Architecture for vSAN                         |
| ECN    | Explicit Congestion Notification                              |
| HA     | High Availability                                             |
| IOMMU  | Input Output Memory Management Unit                           |
| LLM    | Large Language Model                                          |
| LCM    | Life Cycle Management                                         |
| NCCL   | NVIDIA Communications Collective Library                      |
| NGC    | NVIDIA GPU Cloud                                              |
| OOB    | Out- Of- Band                                                 |
| OSA    | Original Storage Architecture for vSAN                        |
| PFC    | Priority Flow Control                                         |
| RDMA   | Remote Direct Memory Access                                   |
| RoCE   | RDMA over Converged Ethernet                                  |
| Sonic  | Software for Open Networking in the cloudCloud                |
| UCX    | Unified Communication X                                       |
| VCF    | VMware Cloud Foundation                                       |
| VDS    | Virtual Distributed Switch                                    |
| vGPU   | NVIDIA GRID Virtual GPU (C-Series)                            |
| VIB    | vSphere Installation Bundles                                  |
| VKS    | vSphere Kubernetes Service                                    |
| VM     | Virtual Machine                                               |
| VI WLD | Virtual Infrastructure Workload Domain                        |
| VLAN   | Virtual Local Area Network                                    |

## Reference

1. [VMware Private AI](#)
2. [VMware Private AI Foundation with NVIDIA on HGX Servers Reference Design for Inference](#)
3. [BCM957608 - Ethernet Networking Guide for NVIDIA H100 GPU Clusters](#)
4. [LLM Inference Sizing and Performance Guidance - VMware Cloud Foundation \(VCF\) Blog](#)
5. [InfiniBand and RoCE Setup and Performance Study on vSphere 7.x](#)
6. [How to run gpt-oss with vLLM | OpenAI Cookbook](#)

## About the author

Dr. Yuankun Fu is a performance engineer at Broadcom focusing on optimizing AI and HPC performance.

## Acknowledgments

The author thanks Ramesh Radhakrishnan, Agustin Malanco, Justin Murray, and Julie Brodeur from Broadcom's VMware Cloud Foundation division; and Sourav Dutta, Velibor Markovski, Surendra Anubolu, and Robin Grindley from Broadcom's CSG team for supporting the paper.

