Installation and Best Practices for Installing Cornelis CN5000 Omni-Path on Lenovo ThinkSystem Servers

Technical showcase of CN5000 Omni-Path deployment on Lenovo ThinkSystem Servers

Best practices for provisioning CN5000 hardware for a HPC cluster

Tunings needed to achieve best performance for HPC out of CN5000 Omni-Path Network

Details key lessons learned from a full deployment on a production HPC cluster

Conor Elrick

LENOVO PRESS

Table of Contents

1 Abstract

This paper provides a technical overview of the deployment and optimization of Cornelis Networks CN5000 Omni-Path® SuperNIC adapters and high-performance fabric switches within a Lenovo ThinkSystem–based HPC cluster. The guidance is based on a real-world implementation in the Lenovo HPC Benchmarking Centre using ThinkSystem SC750 V4 servers powered by Intel® Xeon® 6 processors. The document outlines key considerations for hardware provisioning, software deployment, and fabric validation, including integration with Lenovo Confluent management software for scalable cluster configuration and lifecycle management. It also presents performance validation using industry-standard benchmarks, demonstrating low-latency communication and near line-rate bandwidth across the fabric.

In addition, the paper highlights practical tuning techniques—including MPI configuration, context sharing, bulk transfer services, and SDMA optimization—to help ensure that cluster performance reflects the full capabilities of the underlying hardware.

The methodologies, results, and best practices presented are intended to help HPC architects, system administrators, and Lenovo solution teams deploy CN5000-based clusters efficiently and achieve consistent, high-performance outcomes in production environments.

2 Introduction

This paper presents a technical deep dive into the deployment of Cornelis Networks CN5000 Omni-Path® SuperNIC adapters and high-performance fabric switches within a Lenovo ThinkSystem–based HPC cluster. It is based on a real-world deployment in the Lenovo HPC Benchmarking Centre using ThinkSystem SC750 V4 servers powered by Intel® Xeon® 6 processors.

The document provides a reference architecture and best practices for deploying, validating, and tuning CN5000-based fabrics to achieve optimal performance in HPC environments. It covers both hardware and software considerations, including integration with Lenovo Confluent management software for scalable cluster provisioning and lifecycle management.

The paper is structured as follows:

- The first section introduces CN5000 technology, the Lenovo ThinkSystem platform, and the benchmarking environment.
- The second section outlines key hardware design and provisioning considerations for CN5000 deployments.
- The third section details deployment of the CN5000 software stack, including firmware management, OS integration, and user environment configuration.
- The final section presents validation methodologies, performance results, and tuning recommendations to ensure that measured performance reflects the full capabilities of the underlying hardware.

Performance validation demonstrates near line-rate bandwidth and low-latency communication across the fabric, confirming the effectiveness of the configuration and tuning approach. The lessons learned and best practices presented in this paper are intended to help system architects, HPC administrators, and Lenovo solution teams deploy CN5000-based clusters efficiently and achieve consistent, high-performance results.

2.1 Cornelis CN5000 Omni-Path Technology

The Cornelis CN5000 Omni-Path Fabric is the premier scale-out interconnect for enterprise AI and scientific computing, engineered to seamlessly link high-performance resources through a highly scalable, ultra-high-speed, and ultra-low-latency fabric. The CN5000 Omni-Path Fabric delivers an elite tier of networking features and functions that maximize return on compute investment.

The Cornelis CN5000 Omni-Path series is a complete, end-to-end 400Gb/s scale-out High-Performance Fabric purpose-built to accelerate tightly coupled HPC and AI workloads such as computational fluid dynamics (CFD), molecular dynamics, and large-scale model training spanning from departmental clusters to massive multi-rack supercomputers. The Lenovo ecosystem provides the full suite of fabric building blocks: highly optimized PCIe Gen5 SuperNICs (available as a single-port card, air-cooled or conduction-cooled), a high-density 48-port 1U edge switch, and a director-class 576-port switch platform for flat, low-diameter topologies complemented by qualified cabling options for dense deployments. Delivered as a unified 'host-to-switch-to-fabric-management' stack, the CN5000 guarantees predictable application execution times, eliminated tail latencies, and provides scaling to 100k endpoints.

For latency- and message-rate sensitive MPI and AI communication, the CN5000 guarantees lossless, congestion-free

fabric behavior through hardware-enforced credit-based flow control, advanced predictive congestion management, and fine-grained adaptive routing driven by real-time fabric telemetry. These capabilities preserve peak throughput under heavy multi-tenant, high-utilization conditions. The CN5000 SuperNIC is engineered for 400Gb/s line-rate bandwidth, and a massive, market-leading bidirectional message rate of over 800 million messages per second (800Mmps). These architectural breakthroughs enable exceptional halo-exchange and collective performance as solver partitions scale out, translating directly into higher sustained application throughput as cluster size grows and communication overhead dominates.

Exceptional ease of deployment and enterprise-grade operational openness are delivered through Cornelis CN5000 OPX Software. This lightweight, open-source, OpenFabrics-compliant stack includes host drivers, robust fabric management, and comprehensive diagnostics. OPX features a highly optimized OFI/libafbric provider and maintains full Verbs compatibility. This provides an efficient data path for modern MPI stacks (e.g., Open MPI, MPICH, MVAPICH) and AI communication libraries (e.g., NCCL/RCCL), enabling environments to adopt the CN5000 seamlessly without requiring application rewrites.

Built on the mature and optimized Omni-Path Architecture, the CN5000 specifically unlocks maximum compute efficiency. Key architectural pillars of the CN5000 Omni-Path solution include:

- Credit-Based Flow Control: Ensures packets are transmitted only when receive buffers have verified credits, preventing overruns and eliminating pause frames. Credit-based flow control regulates flow per virtual lane, enabling fair and deterministic allocation across competing workloads.

- Enhanced Congestion Avoidance: Uses forward and backward-propagating telemetry to dynamically adjust injection pacing at the source. This eliminates cascading congestion (congestion spread) in oversubscribed topologies and maintains full throughput even under heavy hotspot pressure.

- Fine-Grained Adaptive Routing (FGAR): Real-time telemetry builds a global heatmap of switch buffer utilization, enabling path selection based on active network states rather than static tables. It dynamically bypasses congested paths on a per-packet basis, reducing long-tail latency and accelerating collective communication.

- Dynamic Lane Scaling and Low-Latency Link Protection: Links degrade gracefully if individual lanes fail, maintaining connectivity instead of triggering full path teardowns. To maintain a sub-microsecond latency profile, link-level replay catches forward-error-correction (FEC) misses at the hardware layer, eliminating costly end-to-end retransmissions.

- Virtual Lanes & Quality of Service: Supports the configuration of up to four Virtual Lanes (VLs) plus one management VL, enforcing hardware-level isolation, dedicated Quality of Service (QoS) guarantees, and secure multi-tenancy for mixed AI and HPC workloads co-running on the same Lenovo infrastructure.

- Open and Interoperable: An open software stack (OPX) built on OFI/libfabric ensures direct, low-overhead integration with major software frameworks, simplifying heterogeneous cluster management.

The Cornelis CN5000 Omni-Path QSFP112 HFI Adapter is engineered for massive scalability, supporting environments from a few nodes to extreme-scale supercomputers of over 100,000 nodes. Delivering 400 Gbps of Omni-Path connectivity with massive message throughput, the CN5000 SuperNIC excels at streamlining communication-intensive, highly parallelized GPU/CPU workloads. Cornelis CN5000 is the elite interconnect choice for enterprise AI training, inference, and scientific computing with Lenovo ThinkSystem servers. A full list of current server support is available at https://lenovopress.lenovo.com/lp1594-thinksystem-ethernet-infiniband-omni-path-adapter-reference#pcie-opa-adapters.

2.2 Lenovo ThinkSystem V4 Servers

The Lenovo ThinkSystem V4 series contains Lenovo's latest generation of compute servers for AI and HPC, created to combine components from leading CPU and GPU vendors with Lenovo's Neptune water cooling technology. The ThinkSystem SC750 V4[1] is powered by the latest generation of Intel Xeon Processors supporting up to two processors with 128 cores and 256 threads per compute node and stands as a powerhouse for demanding HPC workloads. Its industry-leading direct water-cooling system ensures steady heat dissipation, allowing CPUs to maintain accelerated operation and achieve up to a 10% performance enhancement.

Completing the package with support for high-performance NVMe and high-speed, low latency networking such as Cornelis CN5000 Omni-Path, the SC750 V4 is your all-in-one solution for HPC workloads.

At its core, Lenovo Neptune applies 100% direct warm-water cooling, maximizing performance and energy efficiency without sacrificing accessibility or serviceability. The SC750 V4 is installed into the ThinkSystem N1380 Neptune enclosure which itself integrates seamlessly into a standard 19" rack cabinet. Featuring a patented blind-mate stainless steel dripless quick connection, SC750 V4 node trays can be added "hot" or removed for service without impacting other node trays in the enclosure.

This modular design ensures easy serviceability and extreme performance density, making the SC750 V4 the go-to choice for compute clusters of all sizes - from departmental/workgroup levels to the world's most powerful supercomputers – from Exascale to Everyscale.

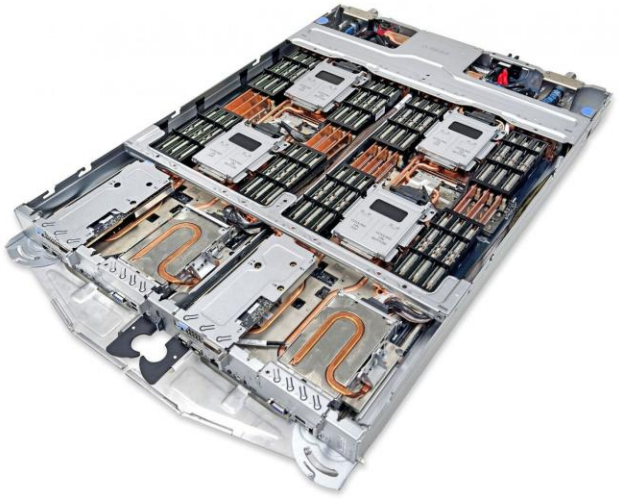

Figure 1 Lenovo ThinkSystem SC750 V4 Server with two dual socket nodes. Photo taken from LenovoPress

2.3 Testing in Lenovo Benchmarking Cluster

The setup we will use for the work presented here from the Lenovo Benchmarking Cluster is a rack containing a Lenovo ThinkSystem N1380 Neptune Direct Water-Cooled Chassis[2]. Within the chassis are 8 trays of the Lenovo ThinkSystem

[1] https://lenovopress.lenovo.com/lp2009-thinksystem-sc750-v4-neptune-server

[2] https://lenovopress.lenovo.com/datasheet/ds0187-lenovo-thinksystem-n1380-neptune

SC750 V4 Neptune Server each consisting of 2 compute “nodes”.

The nodes are configured with the following hardware:

- Dual socket Intel Xeon 6980P 128C/256T Processors operating with a max all-core turbo of 3.2GHz[3]
- 24 units of 64GB DDR5 Memory operating at 6400MT/s for a total system memory of 1536GB
- On-board 25GbE controller for non performance-limited traffic

And they are setup with the following software options

- Operating system: Rocky Linux 9.6
- UEFI option Workload Profile set to High Performance Computing except for:
    - SNC Enabled
    - L0p Enabled
    - Uncore Frequency Scaling Enabled
    - VirtualNuma Disabled

Additionally, we make use of the confluent control server working as both a management server for deploying the cluster and also a DNS server for resolving hostnames.

[3] https://www.intel.com/content/www/us/en/products/sku/240777/intel-xeon-6980p-processor-504m-cache-2-00-ghz/specifications.html

3 Installation of Hardware

In this section we review the initial installation of the CN5000 hardware in the Lenovo Benchmarking cluster, starting with the CN5000 Switch. Firstly we detail the racking the racking of the switch, out-of-the-box configuration, and we perform microcode updates to get the switch ready for the production environment. Secondly we detail the installation of the CN5000 SuperNIC Omni-Path adapters into existing Lenovo ThinkSystem SC750 V4 servers, perform similar microcode updates and then cable together a fabric for benchmarking.

3.1 Installation of CN5000 switch

Figure 2 CN5000 Edge switch racked in the Lenovo Benchmarking Centre

The Cornelis CN5000 Omni-Path 48 port switch comes in 6 options, depending on the power input, cooling method and required orientation on the rack [1]. For out setup we used the air-cooled port-to-fan model number CN5SWE48G2AP which takes up 1U rack space and was compatible with port locations on the ThinkSystem SC750 V4 servers. The front of the switch consists of 48 QSFP112 compatible ports each capable of 400Gb/s bandwidth with the back of the switch offering a single Base-T RJ45 port for switch management. Once the switch was installed in the rack, we connected the management port of the switch to an existing management ethernet network with a DNS server which allowed the switch to get an initial IP address over DHCP once powered on.

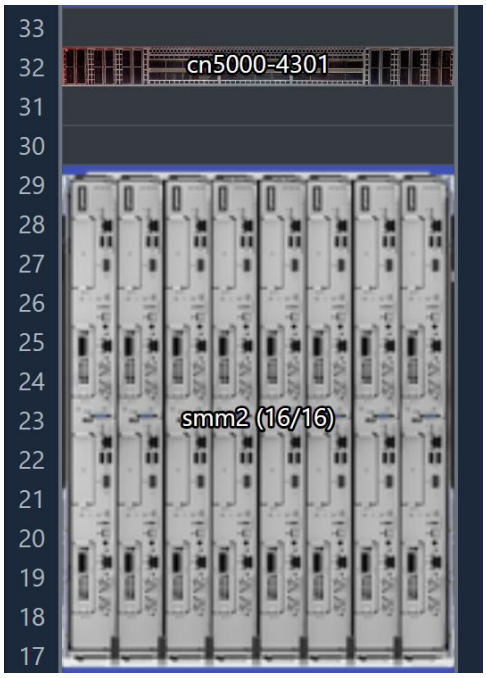

Figure 3 Review of the rack layout for one CN5000 switch and 16 nodes of ThinkSystem SC750 V4 inside the vertically oriented ThinkSystem N1380 enclosure. Taken from a private NetBox[4] installation.

Once the switch was pingable on its DHCP IP address, we need to connect to it with either ssh or using the http(s) OpenBMC web interface in order to do initial switch configuration. We will use a combination of both to showcase each method. The default login credentials for logging into the switch are set to `admin` and `admin123` out of the box which we can use to login to both interfaces for the first login which are changed for subsequent logins. The first thing that was done on the switch was to update the IP address to be static and to be on the correct subnet. This can be done via “Settings” > “Network” > “eth0” as shown in Figure 4, or using the `network ip` commands in the text session.

Next we will flash the firmware on the switch to match the recommended level from Lenovo. The Lenovo EveryScale team periodically publishes a “best recipe” guide which details which combination of firmware levels has been tested in the internal labs and known to be working optimally in order for customers and partners to replicate the setup on their own clusters. The latest best recipe is available at https://support.lenovo.com/us/en/solutions/ht510136. In addition, firmware for the Switches and SuperNICs is available directly from https://customercenter.cornelis.com.

[4] https://github.com/netbox-community/netbox

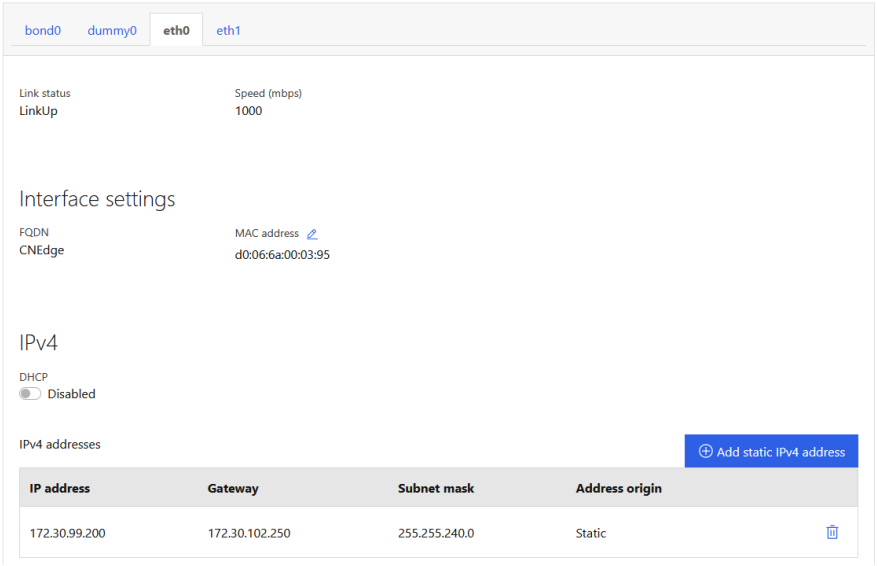

Figure 4 Screenshot of OpenBMC Web interface, configuring the network interface on eth0 to be on the desired subnet.

There are two firmware bundles to update on the switch – one for the BMC interface and one for the OmniPath fabric. The procedure is the same in each case. Inside the switch firmware tar bundle, there will be a ".pkg" file which contains the actual payload to be installed on the switch. With the web interface, there is an option to upload this file and then reboot the switch to install it. With the console interface, we have to either pull this payload onto the switch from somewhere accessible from the switch or we push the payload to the switch from somewhere which can access the switch. This method is detailed in reference [2]. In our setup we did the latter, pushing the payload file from our confluent management server to `/tmp/images/` directory on the switch filesystem using the following:

```
scp -O /install/firmware/OPA_FW/CN5000_BMCFirmware-12.1.0.0.74/CN5000_BMCFirmware-12.1.0.0.74/CN5000.BMCFirmware.12_1_0_0_74.pkg admin@cn5000_4301:/tmp/images
```

Once copied over to the switch, we can initiate the firmware update with the command `firmware update`. We can check the progress of the update by using the command `firmware update -s`:

```
admin@CNEdge -> firmware update -s
BMC:
    Image 1: Booted and Active
    Image 2: Currently updating
ASIC A:
    Image 1: Booted and Active
    Image 2: Inactive image
```

Once we get to the stage when "Currently updating" changes to "Staged for update", we can issue a `reboot force` command to reboot the switch and install the staged firmware. Once this is complete we do the same thing for the ASIC A firmware (named for us CN5000.SwitchFirmware.12_1_0_0_73-V1.pkg), and then confirm the installation was successful by running

`firmware version` on the switch:

```
admin@CNEdge -> firmware version
Firmware Versions:


Switch BMC Chip version: 12.1.0.0.74
ASIC Chip A version: 12_1_0_0_73
```

3.2 Installation of the CN5000 SuperNIC

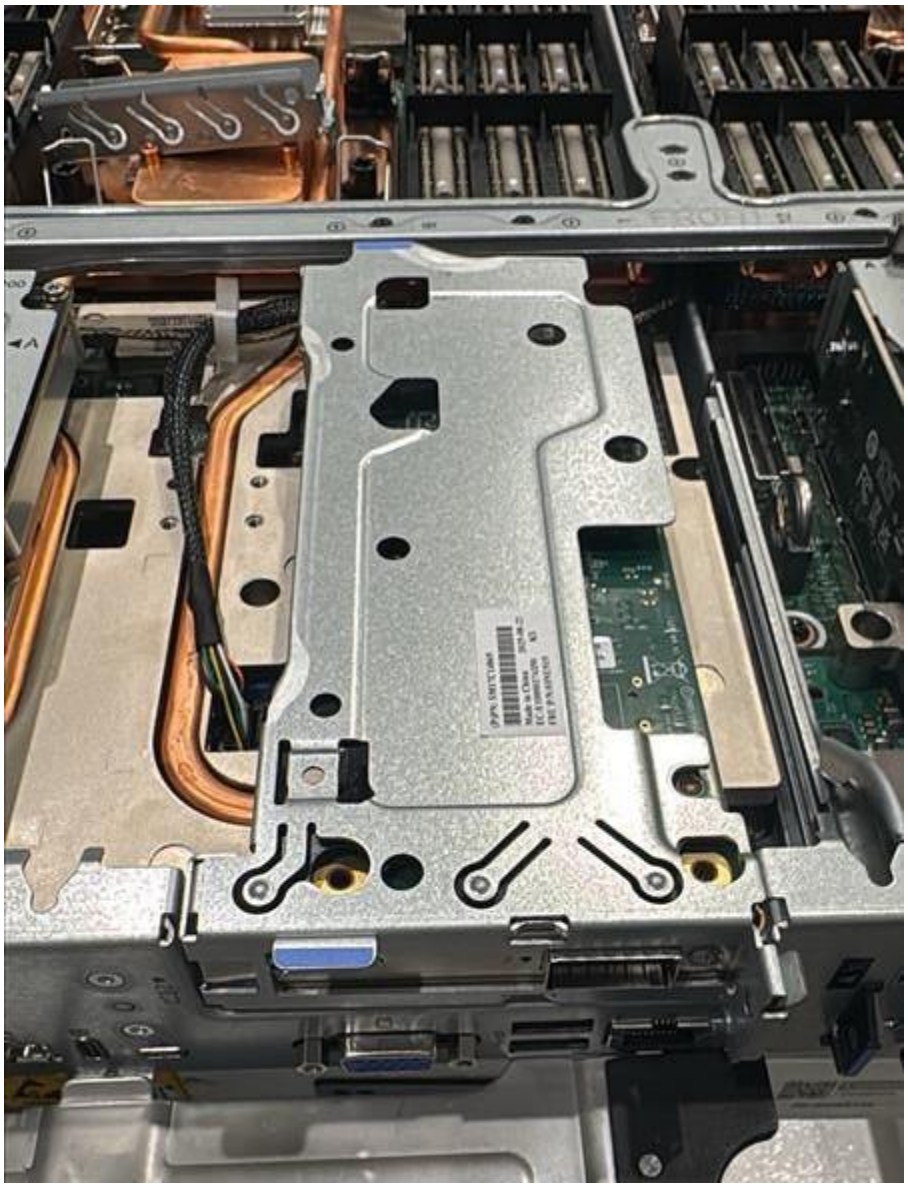

Figure 5 The CN5000 SuperNIC being installed in one of the ThinkSystem SC750V4 servers in the Lenovo Benchmarking Centre

Next we review the installation procedure for the CN5000 SuperNIC on Lenovo ThinkSystem Servers. On the Lenovo ThinkSystem SC750 V4 there are 4 PCIe Expansion ports available for network adapters (2 per node). In our benchmark setup, we make use of one of these PCIe ports per node for a ThinkSystem Cornelis CN5000 Omni-Path 1-Port QSFP112

HFI SuperNIC DWC Adapter. For remote ssh sessions on the servers as well as mounting a shared filesystem, we make use of the onboard 25G controller.

Before connecting the QSFP links between the switch and the nodes, we first want to flash the firmware on the SuperNICs to also be at the level recommended by the latest Lenovo EveryScale Best recipe. Note than in order to do this, we need to be in a booted OS with the HFI1 driver loaded. This is the main topic of the next section of this paper, but for now we assume this is already loaded onto the nodes. From the confluent management node, we copy over both the `updateAgent` tool and the SuperNIC payload firmware to all of the compute servers:

```
noderun gnr4317-gnr4332 scp CN5000_FW_UPDATE_AGENT-12.1.0.0.59/updateAgent\
        {nodename}:/root/
noderun gnr4317-gnr4332 scp CN5000.SuperNICFirmware.12_1_0_0_72-V1.pkg\
        {nodename}:/root/
```

Then we can install it using the following commands:

```
nodeshell gnr4317-gnr4332 "./updateAgent CN5000.SuperNICFirmware.12_1_0_0_72-V1.pkg"
nodepower gnr4317-gnr4332 reboot
```

Upon reboot, we can run the following command on each server to check the firmware updated successfully:

```
$ nodeshell gnr4317 "./updateAgent -V"
gnr4317: HFI hfi1_0 activeComponentImageSetVersionString: 12_1_0_0_72
```

3.3 Cabling the CN5000 Fabric

Once the firmware levels are in the desired state, we proceed to attach the cables between the CN5000 Switch and SuperNICs. Out of the box ports 1,2,11,12,14,16,18,20,30,32,34,36,37,38,47,48 were configured as single links with the remaining ports configured for breakout / splitter connections.

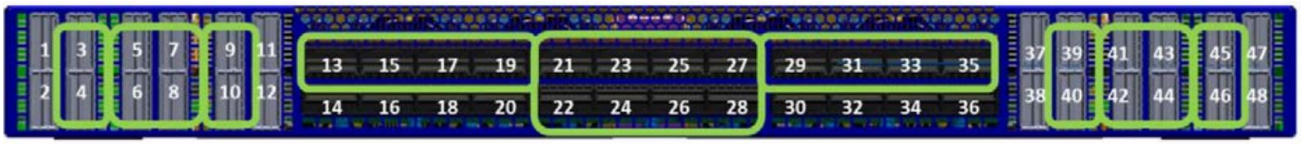

Figure 6 Out of the Box port configuration on CN5000 switch. The ports circled green are configured for splitter connections. Taken from Fabric Installation Guide [3].

The ports can be reconfigured to be split/single links from the switch OS using the `port config modify` command:

```
admin@CNEdge -> help port config
[…]
config: Configure port settings
        show: Show the configuration status for all ports
        modify: Modify port configuration
            speed: Modify the supported speed of the port
                <PORT_NAME>: The name of the port to modify value
                    <VALUE>: Desired Value
```

```
                    val: 25G, 100G
            width: Modify the supported width of the port
                <PORT_NAME>: The name of the port to modify value
                    <VALUE>: Desired Value
                        val: 1X, 2X, 3X, 4X
            subdivision: Modify the supported subdivision of the port
                <PORT_NAME>: The name of the port to modify value
                    <VALUE>: Desired Value
                        val: 2X, 4X
            width_downgrade: Modify the supported width downgrade of the port
                <PORT_NAME>: The name of the port to modify value
                    <VALUE>: Desired Value
                        val: 1X, 2X, 3X, 4X
[…]
```

For our setup with 16 nodes, we used the 16 ports on the switch that were configured out of the box for single links:

```
admin@CNEdge -> port show
Port     Port     Link     Width       Per    Link   Link   Tx Packet   Rx Packet    Port     Port   Cable        Cable              Cable    Cable
Name     State    State    TX    RX    Lane   Qual   Down   Count       Count        Type     Avail  Type         Vendor             Length   S/N
-------- -------- -------- ----- ----- ------ ------ ------ ----------- ------------ -------- ------ ------------ ------------------ -------- ---------------
1A       Link_Up  Active   4x    4x      100Gb   5         2 8491819404 117826643566 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2R7
2A       Link_Up  Active   4x    4x      100Gb   5       583 2312707207 197006865255 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2PG
11A      Link_Up  Active   4x    4x      100Gb   5         0 1974848065 195963512977 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2T8
12A      Link_Up  Active   4x    4x      100Gb   5         0 1941144513 194706772118 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2N3
14A      Link_Up  Active   4x    4x      100Gb   5         3 2564414290 246559495764 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2TG
16A      Link_Up  Active   4x    4x      100Gb   5         0 1938322807 195070673396 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2RM
18A      Link_Up  Active   4x    4x      100Gb   5         0 1843468968 184916845186 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2PK
20A      Link_Up  Active   4x    4x      100Gb   5         0 1842324513 183942276645 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2PF
30A      Link_Up  Active   4x    4x      100Gb   5         0 2247548390 226049561489 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2RU
32A      Link_Up  Active   4x    4x      100Gb   5         0 2136147192 213994905369 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2N7
34A      Link_Up  Active   4x    4x      100Gb   5         0 2150792215 216861222913 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2PM
36A      Link_Up  Active   4x    4x      100Gb   5         0 2011236561 202257824020 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2LT
37A      Link_Up  Active   4x    4x      100Gb   5         0 6157461507  61050182473 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2TE
38A      Link_Up  Active   4x    4x      100Gb   5         1 1937855684 195995711915 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2MU
47A      Link_Up  Active   4x    4x      100Gb   5         3 1892403073 190887473245 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C2LG
48A      Link_Up  Active   4x    4x      100Gb   5         2 1981007713 200826775585 QSFP       Y    DAC          Amphenol             2.0m APF2523N02C39J
```

Figure 7 Output of port show command on the CN5000 switch, showing 16 2m DAC connections with state Link_Up

Figure 8: Side view of a full 16 node N1380 DWC enclosure connected over an CN5000 Omni-Path network (cables with the blue tab).

4 Setup of Software

In this section, we discuss the deployment of a comprehensive diskless OS image using Lenovo confluent [4] that can be used for benchmarking the CN5000 Omni-Path fabric. A lot of the work done here is to build up diskless images that be tweaked if needed over time and then (re-)deployed everywhere in the cluster. For a diskfull setup the steps will be very similar, except one can just reboot the server when changes are needed instead of redeploying the image.

4.1 Baseline Operating System Image

When reviewing the Lenovo EveryScale best recipe for the provisioned system, there will be a software bundle available for “OPXSoftware” associated with a particular operating system that you should use as the base for your image[5]. For our setup we used Rocky Linux 9.6.

We first deploy a “base” rocky image on one of the servers with the CN5000 SuperNICs installed to setup things like repo mirrors and required kernelargs for a confluent image.

```
nodedeploy gnr4317 -n rocky-9.6-x86_64-base-cn5k
```

We will use this node to build up our image which we then make diskless and then deploy onto the other servers. Within the booted OS, we first install all the standard tools we need for compiling and building software on the HPC cluster, as well as the packages needed for the shared parallel filesystem.

4.2 OPX installation

For the OPX installation, we start by downloading the OPX tar bundle on our repo server which is accessible by all the servers in the cluster, extract the contents and creating a repo for DNF to understand.

```
tar -xvzf CN5000_OPXSoftware-RHEL9.6-12.1.1.0.18.tgz
mkdir 9/
mkdir 9/x86_64
mkdir 9/x86_64/12.1.1.0.18
cd 9/x86_64/12.1.1.0.18
cp -r ../../../CN5000_OPXSoftware-RHEL9.6-12.0.1.0.4/* .
createrepo .
cd ../
ln -s 12.1.1.0.18 latest
```

Where in the last step we setup a symlink “latest” to allow for multiple installations that we can easily jump between. On the compute server we add the following to `/etc/yum.repos.d/` to point to our installation:

```
[cn5000-latest-local]
```

[5] Additional OPX bundles associated with an alternative OS or with an older release are available at https://customercenter.cornelis.com/

```
baseurl = http://repo.hpc.eu.lenovo.com/repo/cn5000/9/$basearch/latest/
enabled = 1
gpgcheck = 0
name = Rocky-9-cn5000-local
```

At this point we need to decide which version of the OPX Software we want to install based off of the hardware in the cluster [3]. There is one version that works with the CUDA stack for nVidia GPUs, one that works alongside the ROCM stack for AMD GPUs and one that is for servers without GPUs. For this installation we used the version without GPU support. On the compute server we run the following to install the required packages:

```
dnf makecache
dnf install cn5000_pkgs_non_gpu_meta-12_1_1_0-18.x86_64
```

At this point we now need to build the OPXS module against the running kernel. From the tar bundle with the rpms we installed earlier there should be a ".src.rpm" file which we will need to copy over to the compute node (in this case it was "opxs-kernel-updates-5.14.0_570.12.1.el9_6.x86_64-244.src.rpm").

From there we build the module against the kernel:

```
rpmbuild --rebuild --define "_topdir /root/" --define 'dist %{nil}' -- target x86_64 --define 'kver
$(uname -r)' opxs-kernel-updates-5.14.0_570.12.1.el9_6.x86_64-244.src.rpm
```

Then within `/root/RPMS/x86_64/` directory, there should be 2 created rpms that we want to copy back to the repo server to be made available to all other nodes:

```
scp gnr4317:/root/RPMS/x86_64/* repo:/repo/cn5000/9/x86_64/latest/
ssh repo
cd /repo/cn5000/9/x86_64/latest/
createrepo .
```

Once these are copied over we can install them on our compute node with

```
dnf makecache
dnf -y install kmod-opxs-kernel-updates-5.14.0_570.58.1.el9_6.x86_64-63.x86_64 opxs-kernel-updates-
devel-5.14.0_570.58.1.el9_6.x86_64-63.x86_64
```

At this point we are ready to build a diskless image based on what is installed on our test node – note that since a reboot is required to load the hfi driver correctly, any changes to the OPX packages or update to the underlying linux kernel will require the packages to be rebuilt against the kernel and a new image created.

4.3 Startup of Fabric manager

The fabric manager service needs to be run on one of the servers connected to the switch and cannot be run on the switch itself. For our setup we chose the node `gnr4317` to be the node with the fabric manager service running. With this node:

- We set the node to not automatically power down when not in use (whilst other nodes on the fabric are free to power down when not in use)
- We set the node to be in a separate slurm partition to other nodes on the fabric, so that the other nodes that aren't running the fabric manager have higher priority of being used
- The `opafm` service is set to start on boot

For the single port adapter, the HFI driver sees two ports on each adapter with only the second one being usable. In order to start the opafm service, we adjust the fabric manager config to scan for connections using the second port on the adapter:

```
$ cat /etc/opa-fm/opa.xml
[…]
    <Name>fm0</Name> <!-- also for logging with _sm, _fe, _pm appended -->
    <Hfi>1</Hfi> <!-- local HFI to use for FM instance, 1=1st HFI -->
    <Port>2</Port> <!-- local HFI port to use for FM instance, 1=1st Port -->
[…]
```

With this set, we are then at a stage where we can start the fabric manager and test the fabric:

```
[root@gnr4317 ~]# systemctl start opafm
[root@gnr4317 ~]# systemctl status opafm
● opafm.service - OPA Fabric Manager
     Loaded: loaded (/usr/lib/systemd/system/opafm.service; disabled; preset: disabled)
     Active: active (running) since Mon 2026-05-11 15:24:54 CEST; 1min 24s ago
    Process: 54942 ExecStart=/usr/lib/opa-fm/bin/opafmd -D (code=exited, status=0/SUCCESS)
   Main PID: 54943 (opafmd)
      Tasks: 533 (limit: 3355442)
     Memory: 432.4M
        CPU: 7.404s
     CGroup: /system.slice/opafm.service
             ├─54943 /usr/lib/opa-fm/bin/opafmd -D
             └─54944 /usr/lib/opa-fm/runtime/sm -e sm_0
[…]
May  11  15:26:11  gnr4317.hpc.eu.lenovo.com  fm0_sm[54944]:  PROGR[topology]:  SM:  topology_main:
DISCOVERY CYCLE END. 1 SWs, 16 HFIs, 16 end ports, 33 total ports, 1 SM(s), 1030 packets, 0 retries,
0.111 sec sweep
```

5 Validation of Setup

In this section we perform a validation of the setup that was installed in the Lenovo benchmarking cluster, testing to make sure all links are working correctly and reviewing the latency and bandwidth performance we can measure from the OS.

⚠ Note: Performance measurements detailed in this section are for the specific hardware that was installed in our test setup. These measurements should not be taken as a commitment from Lenovo to match these results on another system where results can vary due to factors such as cable types, cable length and switch hops.

5.1 Testing the CN5000 Fabric

To start with the validation of our setup, we first review the fabric as presented by the OPA fabric manager to ensure all links are detected correctly. If a node has been detected by the fabric manager we expect to see on the node that the PortState is active and had a high Link Quality e.g.:

```
$ opainfo  -p 2
hfi1_0:2                          PortGID:0xfe80000000000000:d0066a0201000a8b
   PortState:     Active
   LinkSpeed      Act: 100Gb        En: 25Gb,100Gb
   LinkWidth      Act: 4            En: 1,2,3,4
   LinkWidthDnGrd ActTx: 4  Rx: 4   En: 3,4
   LCRC           Act: 16-bit       En: 16-bit         Mgmt: True
   LID: 0x00000001-0x00000001       SM LID: 0x00000001 SL: 2
         QSFP+ Copper,      2.0mAmphenol          P/N NJAAK3-CN02       Rev A
   Xmit Data:          419938602 MB Pkts:         117411660396
   Recv Data:          153676351 MB Pkts:          84503384523
   Link Quality: 5 (Excellent)
```

On the fabric manager node, we should see the nodes detected as part of a sweep, e.g.

```
$journalctl -u opafm.service
[…]
topology_main: TT: DISCOVERY CYCLE START - REASON: Scheduled sweep interval
topology_main: TT: DISCOVERY CYCLE END. 1 SWs, 16 HFIs, 16 end ports, 33 total ports, 1 SM(s), 1030
packets, 0 retries, 0.126 sec sweep
```

If a node has only recently been detected, you may need to wait for a discovery cycle to start before it is detected. To review

all of the devices on the fabric we can use `opafabricinfo`, e.g.

```
[root@gnr4317 ~]# opafabricinfo

Fabric 0:0 Information:

SM: gnr4317 hfi1_0 Guid: 0xd0066a0201000a8b State: Master

Number of HFIs: 16

Number of Switches: 1

Number of Links: 16

Number of HFI Links: 16             (Internal: 0   External: 16)

Number of ISLs: 0                   (Internal: 0   External: 0)

Number of Degraded Links: 0         (HFI Links: 0   ISLs: 0)

Number of Omitted Links: 0          (HFI Links: 0   ISLs: 0)

--------------------------------------------------------------------------------
```

In addition, if we setup out cluster with an explicit fabric topology file, we can check the current fabric versus the topology file using the command `opareport -o verifyall` as detailed in section 6 of the CN5000 Fabric installation guide [1], but since we only had a single switch in our test cluster we skipped this step.

When the fabric is up, a useful command to have running is `opatop` which can be used to review stats on the current traffic going through the fabric.

```
opatop: Img: 10s @ Fri May  8 15:59:55 2026, Live
Summary:  SW:     1 Ports: SW:    17  HFI:    16       Link:    16
          SM:     1 Node NRsp:     0 Skip:     0 Port NRsp:     0 Skip:     0
                    AvgMBps   MinMBps   MaxMBps   AvgKPps   MinKPps   MaxKPps
0 All         Int       743         0     12258       101         0      1540
      Integ:min  Congst:OVER SmaCong:min  Bubble:min  Secure:min  Routing:min
1 HFIs        Snd       766         0     12258       104         0      1540
              Rcv       766         0     12258       104         0      1540
      Integ:min  Congst:OVER SmaCong:min  Bubble:min  Secure:min  Routing:min
2 SWs         Int         0         0         0         0         0         0
              Snd       766         0     12258       104         0      1540
              Rcv       766         0     12258       104         0      1540
      Integ:min  Congst:OVER SmaCong:min  Bubble:min  Secure:min  Routing:min

    Master-SM: LID: 0x0001 Port: 2   Priority: 0  State: Master
               Name: gnr4317 hfi1_0
               PortGUID: 0xD0066A0201000A8B
 Secondary-SM: none



Quit up Live/rRev/fFwd/time/bookmrked Bookmrk Unbookmrk ?help |
sS Pmcfg Imginfo View 0-n:
```

Figure 9: Showcase of opatop running on a node with CN5000 SuperNIC installed whilst a benchmark is running.

5.2 Building Modules for Toolchains

In order to perform tests on the fabric with MPI benchmarks, we first need to build some MPI modules which will include all the compilers, libraries and environment variables required to build and run the benchmarks on the system. To provide two

options we build a module with Intel One-API[6] and one with OpenMPI[7] which we will build according to the CN5000 Performance Tuning Guide [1].

For Intel MPI, we download the offline installer from the official webpage and run it to installed a shared installation on our system.

```
./intel-oneapi-hpc-toolkit-2025.2.1.44_offline.sh -a -s --eula=accept\
      --install-dir=/hpc/xproj/cn5000_opa/software/intel-oneapi --instance=cn5k-hpc
```

This will create an installation of OneAPI which we can source with the created `/env/vars.sh` script. At the same time as sourcing OneAPI, there are some environment variables we want to set globally for all users on the system. These are:

- `I_MPI_OFI_LIBRARY_INTERNAL=0` to disable the libfabric that comes with OneAPI
- `I_MPI_FABRICS=shm:ofi` to set OFI as the fabric used
- `FI_PROVIDER=opx` to use the OPX provider
- `I_MPI_DEBUG=5` to set debug level of Intel MPI (not required but useful)
- `FI_OPX_CONTEXT_SHARING=1` to enable the use of shared contexts [Required for our choice of CPU SKU]
- `FI_OPX_ENDPOINTS_PER_HFI_CONTEXT=2` to set 2 MPI end points per context [Sufficient for our choice of SKU]

For convenience we can set this up as a module file or as a shell script which can be sourced. The Lua module used for our setup is included in Appendix A1.

For OpenMPI, after we have obtained a release tar bundle from the official webpage, we build it with a GCC installed already on our system via EasyBuild[8] (GCC v15.2.0).

```
INSTALL=/hpc/xproj/cn5000_opa/software/OMPI_5.0.10


module load eb-env
module load GCC
tar -xvzf openmpi-5.0.10.tar.gz
cd openmpi-5.0.10
./configure --prefix=$INSTALL --with-ofi=/usr --enable-orterun-prefix-by-default
make -j 128
make install
```

A module file to source this installation and set the required environment variables from the list above relevant for OpenMPI is provided again in Appendix A1.

---

[6] Intel One-API publically available from https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html

[7] OpenMPI publically available from https://www.open-mpi.org/

[8] Easybuild available at https://easybuild.io/

5.3 Latency Testing

Latency testing is going to strongly depend on the exact hardware used with options such as cable length, switch ports used and number of switch hops all adding latency to the transmitted signals. In our benchmark centre we have sixteen units of 2m DAC cables connected all to a single CN5000 Switch. On the CN5000 switch there are a couple of different port configurations that can be used.

According to the cabling guidance documentation [2], only DAC cables of up to 1.5m can make use of LLR (Log-Likelihood Ratio) technology for latency-optimized communication between the CN5000 switch and SuperNICs. For latency-optimized communication with longer DAC cables, there are 7 ports available on the switch with support for up to 2.0m links and 11 ports available on the switch with support for up to 2.5m links.

These ports are highlighted in the below figure, with ports 5, 6, 8, 41, 42, 44, and 48 allowing for LLR to be used on 2.0m DAC links and ports 19 to 29 supporting LLR on 2.5m DAC links.

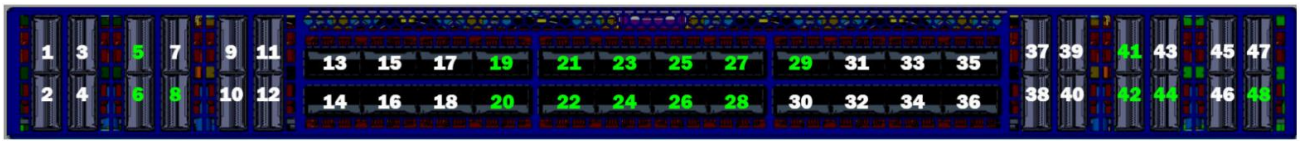

Figure 10: CN5000 Switch with "Extended reach ports" highlighted in green. Taken from Cornelis Network CN5000 cabling guidance document [2].

With the current firmware released, FEC (Forward Error Correction) is enabled on all of the switch ports and disabling it to use LLR for lower latency is not an available option at this time but may be available again in future.

For our testing in this paper, we measured latency without using the latency optimised ports since this will reflect the worst case latency measured if the switch were to be fully populated, and we also used this configuration for bandwidth testing presented later in this section and for the performance tuning tests showcased in the section which follows.

5.3.1 Testing Methodology

To measure node to node latency, we use the micro benchmarks provided by Ohio State University (OSU) [3] built with the OpenMPI module we created in the previous subsection.

```
./configure --prefix=/hpc/xproj/cn5000_opa/software/OSU_7.5.2/
make -j 64
make install
```

We then use the following commands to measure

- Single MPI rank <-> Single MPI rank latency using `osu_latency`
- Multiple MPI rank <-> Multiple MPI rank latency using `osu_multi_lat`

5.3.2 Latency Measurements with standard switch port configuration

With the `osu_latency` test we measure the following performance for an example pair of nodes using a core that is local to the adapter:

```
$ mpirun -x FI_PROVIDER=opx  -mca mtl ofi -mca btl self,vader --bind-to core -n 2 --host gnr4318-
opa1:1,gnr4319-opa1:1 taskset -c 1 osu_latency

# OSU MPI Latency Test v7.5.2
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                       1.19
2                       1.19
4                       1.19
8                       1.19
16                      1.19
32                      1.26
64                      1.25
128                     1.27
256                     1.29
512                     1.34
1024                    1.46
2048                    1.64
4096                    1.97
8192                    2.70
16384                   3.84
32768                  13.88
65536                  16.99
131072                 21.31
262144                 26.24
524288                 32.25
1048576                45.83
2097152                73.58
4194304               135.06
```

And for an core than is on the opposite socket to the adapter:

```
$ mpirun -x FI_PROVIDER=opx  -mca mtl ofi -mca btl self,vader --bind-to core -n 2 --host gnr4318-
opa1:1,gnr4319-opa1:1 taskset -c 128 osu_latency

# OSU MPI Latency Test v7.5.2
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                       1.59
```

```
2                       1.58
4                       1.58
8                       1.57
16                      1.57
32                      1.85
64                      1.87
128                     1.89
256                     1.93
512                     1.98
1024                    2.22
2048                    2.38
4096                    2.87
8192                    4.11
16384                   5.92
32768                  19.56
65536                  24.64
131072                 28.38
262144                 36.25
524288                 55.62
1048576                90.73
2097152               155.10
4194304               287.40
```

With multiple MPI ranks per node, we measure the following latency for an example pair of servers:

```
$ mpirun -x FI_PROVIDER=opx  -mca mtl ofi -mca btl self,vader --bind-to core -n 64 --host gnr4318-
opa1:32,gnr4319-opa1:32 osu_multi_lat


# OSU MPI Multi Latency Test v7.5.2
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                       1.11
2                       1.10
4                       1.10
8                       1.09
16                      1.09
32                      1.28
64                      1.17
128                     1.18
```

```
256                    1.22
512                    1.37
1024                   1.49
2048                   1.91
4096                   2.98
8192                   5.64
16384                 12.18
32768                 20.84
65536                 27.36
131072                64.43
262144               160.17
524288               323.79
1048576              511.35
2097152             1062.42
4194304             2242.83
```

We repeated this test for all node pairs in the setup (including the server running the opa fabric manager), and observed that the minimum reported latency from the osu test was between 1.09 ns and 1.11 ns and averaging at 1.10 ns. These results match what was expected for this setup by the Cornelis Engineers.

5.4 Bandwidth Testing

For bandwidth testing we made us of the OSU MPI multiple bandwidth test `osu_mbw_mr` to make sure we were using enough data to saturate the connection[9]. For an example pair of nodes we observed the following:

```
$ mpirun -x FI_PROVIDER=opx -mca mtl ofi -mca btl self,vader --bind-to core -n 6 --host gnr4318-opa1:3,gnr4319-opa1:3 osu_mbw_mr


# OSU MPI Multiple Bandwidth / Message Rate Test v7.5.2
# [ pairs: 3 ] [ window size: 64 ]
# Datatype: MPI_CHAR.
# Size                  MB/s        Messages/s
```

[9] A patch was needed to ensure the reported bandwidth was calculated correctly.

```
<       MPI_CHECK(MPI_Reduce(&t, &temp_t_reduce, 1, MPI_DOUBLE, MPI_SUM, 0,
>       MPI_CHECK(MPI_Reduce(&t, &temp_t_reduce, 1, MPI_DOUBLE, MPI_MAX, 0,
<       t = temp_t_reduce / num_pairs;
>       t = temp_t_reduce ;
```

Without this fix, the results calculated by OSU are higher than the actual value.

```
1                       24.14       24143836.91
2                       50.44       25218715.44
4                      107.91       26976673.61
8                      221.47       27683563.41
16                     431.96       26997270.74
32                     720.61       22519123.66
64                    1510.85       23606983.08
128                   2947.72       23029075.01
256                   5532.14       21609916.25
512                   9878.77       19294465.70
1024                 14080.93       13750908.97
2048                 18559.19        9062105.16
4096                 22639.42        5527202.52
8192                 26825.58        3274606.45
16384                27276.06        1664798.78
32768                36771.00        1122162.00
65536                44354.08         676789.61
131072               47555.11         362816.72
262144               48976.09         186828.97
524288               48238.39          92007.43
1048576              48664.27          46409.86
2097152              48055.80          22914.79
4194304              48322.09          11520.88
```

Using a transfer size of 4194304 bytes we observed across all node pairs a bandwidth of between 378 Gb/s and 391 Gb/s, with an average result of 385 Gb/s which is very close to the expected line rate of 400 Gb/s, confirming correct configuration.

6 Initial Performance Tuning

In this section we discuss some of the global settings needed to achieve a good baseline performance for applications based on studies we've done on the setup in the Lenovo benchmarking cluster. Some of these are general and will apply to most other setups using ThinkSystem and CN5000 and some are specific to the hardware we have in the benchmarking centre.

6.1 Settings and Tunables

6.1.1 MPI Contexts and Context Sharing

With MPI, the basic unit of communication space dictating the number of concurrent messages that can be sent and received at one time. Messages sent from one endpoint to another will use a context and a message sent in one context cannot be received in a different context. By default on CN5000 Omni-path, the maximum number of MPI ranks that is officially supported is limited by the number of available contexts at 208 [2]. At boot time, we can see the number of contexts available in the console e.g.:

```
[    6.893839] hfi1 0000:15:00.0: hfi1_0: rcv contexts: avail 236
[    6.907489] hfi1 0000:15:00.0: hfi1_0: RcvArray per-context groups 74, unused groups 26
[    6.911957] hfi1 0000:15:00.0: hfi1_0: unused send context blocks: 78
[    6.916304] hfi1 0000:15:00.0: hfi1_0: send contexts: avail 236, used 230 (kernel 8, ack 3, user 218, vl15 1)
```

When the node is booted, we can use the following to check the current number of free contexts available to the user:

```
$ grep . /sys/class/infiniband/hfi1_0/ports/2/num_freectxts
208
```

This means that out-of-the-box[10] we are limited to 208 MPI ranks per node in a multi node workload. We can however make use of MPI context sharing to allow multiple MPI ranks to share the same MPI context, raising the number of MPI ranks we can use per node. For our setup of nodes powered by two sockets of Intel Xeon 6980p 128C/256T processors, this is required to make use of all the available cores in multi-node workloads.

To enable context sharing, we can use two environment variables already showcased in the previous section. Without utilising the virtual cores available through hyperthreading it was sufficient to set FI_OPX_ENDPOINTS_PER_HFI_CONTEXT to 2, but with if we wanted to have MPI ranks on the virtual cores or if we have a higher core count SKU or even additional CPU sockets, this will need to be set higher.

To study the effects on communication latency when multiple endpoints are used, we ran an OSU All reduce benchmark using from 1 to 256 tasks per node. In Figure 11 Plot of OSU All Reduce Latency versus number of MPI ranks per node for 16 nodes. we plot the reported latency for an all-reduce operation on 16 nodes with an 8 byte message size using no shared contexts, 2 tasks per context, 4 tasks per context and 6 tasks per context to simulate using very high core count CPUs with

[10] This default behaviour may change in upcoming revisions

CN5000 Omni-path. Here we see there is a marked increase in latency reported when context sharing is enabled due to the blocking on messages using the same context.

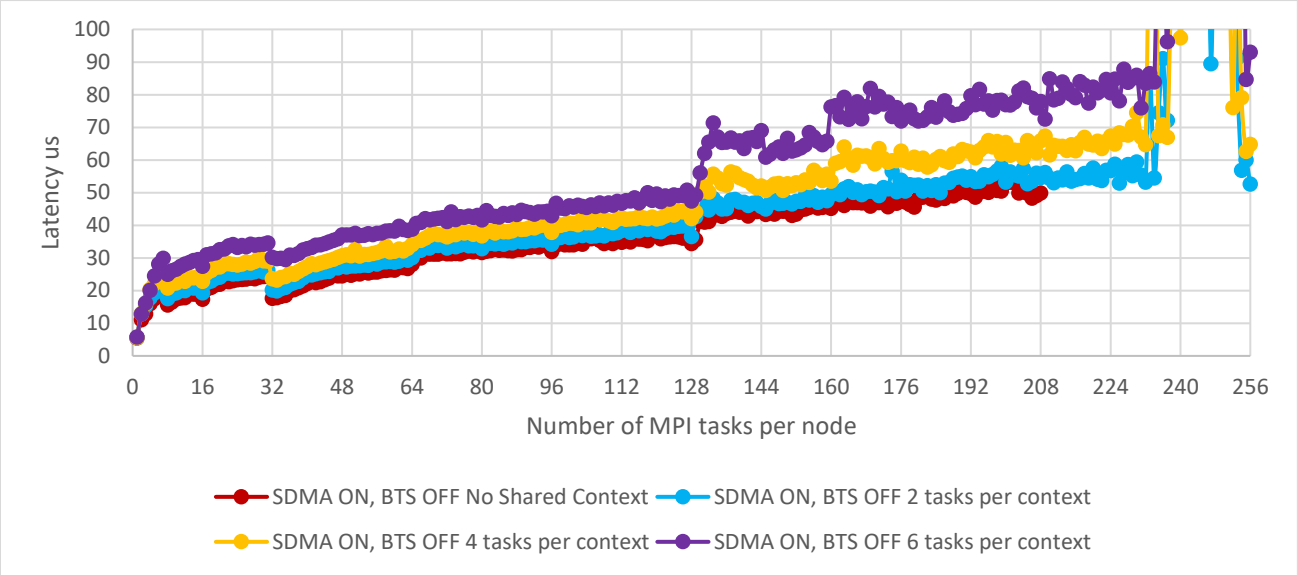

Figure 11 Plot of OSU All Reduce Latency versus number of MPI ranks per node for 16 nodes.

In Figure 12 we show the difference between each of the lines in Figure 11 by plotting the latency normalised to the result without using context sharing. We see that using 2 tasks per context increases all-reduce time by around 0-20%, 4 tasks per contexts increased all-reduce time by up to 40% and using 6 tasks per context increased all-reduce time by up to 80%. For completeness we also tested using the other combinations of SDMA and BTS which are described in the proceeding section, but saw the same behaviour in those cases.

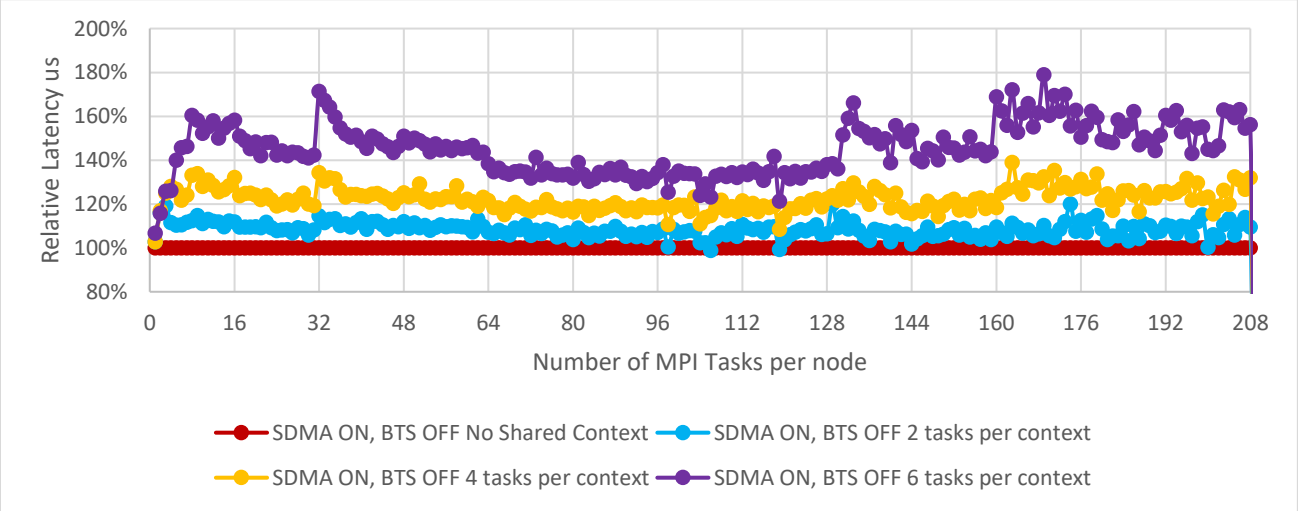

Figure 12 Plot of OSU All Reduce Latency versus number of MPI ranks per node for 16 nodes normalised to the result without context sharing.

The impact of sharing HFI contexts on real workloads is planned to be reviewed in a subsequent paper.

6.1.2 Bulk Transfer Service

Bulk transfer service (BTS) is a technology that reserves HFI contexts which are used on top of the user allocated contexts

in order to improve the communication bandwidth for some workloads. To enable BTS, we need to reserve some of the MPI contexts at boot time which will no longer be available to the user. Within Lenovo Confluent, we can do this by adapting the kernel arguments which are located in the `profile.yaml` file associated with our diskless OS image. As an example, the following is what was used in the Lenovo Benchmarking cluster as kernel arguments for the compute servers:

```
# cat /var/lib/confluent/public/os/rocky-9.6-cn5k-diskless-v1/profile.yaml
label: Rocky Linux 9.6 (Blue Onyx) (Diskless Boot)
kernelargs: clocksource=tsc tsc=reliable confluent_imagemethod=untethered \
hfi1.use_bulksvc=Y hfi1.num_sdma=8 hfi1.num_vls=4 hfi1.num_user_contexts=0,208
 initcall_blacklist=algif_aead_init
```

- To use BTS, we set `hfi1.use_bulksvc` to `Y`
- To ensure there are enough contexts available for BTS to start, we limit the number of user contexts to 208 using `hfi1.num_user_contexts=0,208` (it is 0,208 because we are using “port” 2)
- Additionally `num_sdma=8 num_vls=4` are set as optimal options for HFI driver as according to the CN5000 Performance Tuning Guide [2].

Then from within the booted, the user can choose to use BTS at runtime with the environment variable,

```
FI_OPX_HFISVC=1
```

Depending on the specific application and workload in question, this may end up improving or hindering the measured performance.

6.1.3 SDMA

System direct memory access (SDMA) is another option that can be toggled to improve the performance of certain applications and workloads. With `num_sdma=8` set in our kernel options, SDMA will be used. The user can choose to disable SDMA at run time by setting:

```
FI_OPX_SDMA_DISABLE=0
```

6.2 Future Publications and Studies

In a planned future study, we will review the performance of different applications and workloads when running over a CN5000 Omni-Path network to see how the different options presented here affect the scaling of the tests with number of nodes used.

7 Conclusions

In this paper, we showcased the setup and deployment of a CN5000 Omni-Path in the Lenovo Benchmarking Cluster aiming to include sufficient detail so that this serves as a guide for deploying a CN5000 Omni-Path network in future ThinkSystem based HPC clusters. During our deployment we learned a number of lessons on how to get the most out of the hardware which we discussed in order to aid customers, partners and Lenovo Professional services team.

During the scoping and sizing phase, care must be taken to ensure that you are using the best available switch port configuration for your cluster since certain ports support protocols which enable lower latency between pairs of CN5000 switches and between CN5000 switches and SuperNICs for certain cable types and lengths. To plan a topology for a large (greater than one switch) cluster, it is recommended to adapt the Excel fabric template files for all cables and endpoints so that when the cluster is deployed the fabric can be tested against the template ensuring everything is setup as desired. Care must also be taken with the proposed server type and CPU SKU since above 208 cores per node, context sharing must be enabled in order for the all cores in the CPU to be able to be utilised in multi-node workloads which may impact the performance of the workload.

During the deployment of a HPC cluster, this paper details how best to setup the hardware and software for CN5000 Omni-path in order for the setup to be fully functional and performant. Depending on your final workload of interest and the choice of server + CPU SKU, there are number of tunings and options that need to be considered for optimal performance of the network. Some of these options need to be set at the level of the OS image and will not be able to be changed by the regular end user, and we've shown how this can be done with Lenovo Confluent. Other tunings can be set globally for all users in the modules of the packages and toolchains, and some final tunings which are workload specific will need to be set by the end user.

In a subsequent article we plan to detail which tunings give optimal performance for a selection of real world HPC workloads when scaled out to multi-node tests.

8 References

[1] Cornelis Networks, “CORNELIS CN5000 OMNI-PATH Product Brief V1.0,” 2026. [Online]. Available: https://www.cornelis.com/site/documents/products/cn5000_switch_pb_a00922_v1.0.pdf.

[2] Cornelis Networks, “CN5000 Maintenance and Troubleshooting Guide V1.6,” 2026. [Online]. Available: https://customercenter.cornelis.com/.

[3] Cornelis Networks, “CN5000 Fabric Installation Guide V3.1,” [Online]. Available: https://customercenter.cornelis.com/.

[4] A. Thapa, A. Ortiz and J. Johnson, “Implementing Lenovo Confluent Management Software,” 29 July 2025. [Online]. Available: https://lenovopress.lenovo.com/lp2266-implementing-lenovo-confluent-management-software.

[5] Cornelis Networks, “CN5000 Performance Tuning Guide V3.0,” 2026. [Online]. Available: https://customercenter.cornelis.com/.

[6] Cornelis Networks, “Cornelis® CN5000 Omni-Path® Cabling V1.1,” 2026. [Online]. Available: https://customercenter.cornelis.com/.

[7] Ohio State University, “MPI Microbenchmarks V7.5.2,” [Online]. Available: https://mvapich.cse.ohio-state.edu/benchmarks/.

[8] Intel, “Intel One-API,” [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html.

9 Author

Conor Elrick is a HPC/AI Performance Engineer in the ISG Offerings Group at Lenovo working as part of the HPC/AI Solutions Architect Team. Conor does setup and performance benchmarking on compute servers, networking infrastructure and storage solutions to push hardware to its limits and get the best results for real world scientific workloads. He has been at Lenovo for over 2 years. Conor earned his PhD in Theoretical Particle Physics from the University of Edinburgh in 2024 and a Master's Degree and Bachelor's Degree in the same field before that, with a focus on physics simulations on compute systems.

We would like to thank:

- Kevin Dean for review and discussions
- Cornelis Networks team including Paul Stasurak, John Swinburne, Balaji Srinivasan, James Erwin, Pavel Dobrinskiy and Tyle Cruise for input throughout and for validating the setup

A1: Lua modules for sourcing toolchains

The module used for sourcing Intel One-API is as follows:

```
$ cat modules/intel-opa/1.0.0.lua
help([[
Intel oneAPI + Intel MPI OPX/OFI configuration
]])

whatis("Intel oneAPI with Intel MPI OPX fabric")

-- Ensure external OFI is used
setenv("I_MPI_OFI_LIBRARY_INTERNAL", "0")

-- Source Intel oneAPI environment
source_sh("bash", "/hpc/xproj/cn5000_opa/software/intel-oneapi/setvars.sh --force")

-- MPI / OFI settings
setenv("I_MPI_FABRICS", "shm:ofi")
setenv("FI_PROVIDER", "opx")
setenv("I_MPI_DEBUG", "5")
setenv("FI_OPX_CONTEXT_SHARING", "1")
setenv("FI_OPX_ENDPOINTS_PER_HFI_CONTEXT", "2")
```

The module used for sourcing OpenMPI is as follows:

```
$ cat modules/gompi-opa/1.0.0.lua
help([[
CN5000 Omni-Path OpenMPI 5.0.10 environment
]])

whatis("Name: CN5000 OpenMPI")
whatis("Version: 5.0.10")
whatis("Description: OpenMPI 5.0.10 built for Cornelis CN5000 Omni-Path")

-- Prerequisites
depends_on("eb-env")
depends_on("GCC")

-- Base path
local base = "/hpc/xproj/cn5000_opa/software/OMPI_5.0.10"
```

```
-- Paths
prepend_path("PATH", pathJoin(base, "bin"))
prepend_path("LD_LIBRARY_PATH", pathJoin(base, "lib"))
prepend_path("LIBRARY_PATH", pathJoin(base, "lib"))

-- Libfabric / OPX environment
setenv("FI_OPX_CONTEXT_SHARING", "1")
setenv("FI_OPX_ENDPOINTS_PER_HFI_CONTEXT", "2")
setenv("FI_PROVIDER", "opx")
```

Trademarks and special notices

References in this document to Lenovo products or services do not imply that Lenovo intends to make them available in every country.

The following terms are trademarks of Lenovo in the United States, other countries, or both:

Lenovo®
Neptune®
ThinkSystem®
XClarity®

The following terms are trademarks of other companies:

Intel® is a trademark of Intel Corporation or its subsidiaries.

Linux® is the trademark of Linus Torvalds in the U.S. and other countries.

IBM® is a trademark of IBM in the United States, other countries, or both.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used Lenovo products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-Lenovo products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by Lenovo. Sources for non-Lenovo list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. Lenovo has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-Lenovo products. Questions on the capability of non-Lenovo products should be addressed to the supplier of those products.

All statements regarding Lenovo future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local Lenovo office or Lenovo authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in Lenovo product announcements. The information is presented here to communicate Lenovo’s current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard Lenovo benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Any references in this information to non-Lenovo websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this Lenovo product and use of those websites is at your own risk.