Interconnect Performance Validation
Open-Source Mini-Clusters
CentOS8.3 Base OS, OpenHPC (v2.3)
xCAT/SLURM Edition for Linux* (x86_64)

Isayah Reed

November 2021

Revision 1.4

Table of Contents

Version	Description
v1.0	Initial draft. OpenMPI and IMPI installation. OSU and IMP setup. 1GbE validation
v1.1	Added 40GbE configuration and IMPI validation
v1.2	Added Omni-Path IFS configuration and IMPI validation
v1.3	Edited OPA configuration steps
v1.4	Added InfiniBand configuration and validation, using open-source/centOS tools and drivers

Introduction

The top-level instructions include information for the installation and functional validation of OpenMPI, with installation on the head and compute nodes.

```
$> dnf -y install openmpi openmpi-devel gcc-c++
$> dnf -y --installroot=$CHROOT install openmpi openmpi-devel gcc-c++
$> packimage centos8-x86_64-netboot-compute
$> pdsh -w c[1-2] reboot
```

Successful installation is confirmed with functional validation testing. There were basic connectivity tests by pinging to ensure the compute nodes can reach the head node, and using a sample MPI script to demonstrate that the compute nodes can communicate with each other. In this document we will move beyond functional testing and validate network performance to ensure it meets expectations. Here, performance validation is used to determine the speed of network communications over a given interface. In other words, this document will validate that the communication network(s) are performing at an acceptable level. Performance will be measured using the using the OSU Micro-Benchmark suite, executed with the open-source OpenMPI, in addition to the IMB benchmarks from Intel OneAPI.

Ethernet

This section will validate the performance of the standard Ethernet connection using OpenMPI. It assumes a head node connected to two compute nodes - c1 and c2 - in a minimal cluster setup using the instructions in the top-level cluster_setup documentation, with no other connections on the cluster such as Infiniband or high-speed Ethernet connection. Using root is not recommended unless otherwise noted.

OpenMPI

Install the OSU Micro-Benchmarks that will be used for performance testing. The following commands install the OSU benchmarks in the user's home directory.

```
$> wget https://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.8.tgz
$> tar -xzf osu-micro-benchmarks-5.8.tgz
$> cd osu-micro-benchmarks-5.8/
$> ./configure CC=/usr/lib64/openmpi/bin/mpicc CXX=/usr/lib64/openmpi/bin/mpicxx
$> make
$> make install exec_prefix=~/osu_benchmarks_openmpi
```

We will validate performance by using one-sided (RMA) communication for lower overhead. The osu_put_bw test is adequate for this purpose. Use the following sample script to execute the osu_put_bw test on two compute nodes:

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -J perf_test
#SBATCH -p normal
#SBATCH -t 20
#SBATCH -o osu_perf_test.out
#SBATCH -e osu_perf_test.err

export PATH=/usr/lib64/openmpi/bin:$PATH
mpirun -n 2 -N 1 -mca btl self,tcp
~/osu_benchmarks/libexec/osu-micro-benchmarks/mpi/one-sided/osu_put_bw
```

Submit the script on the head node with sbatch. A sample output from the osu_put_bw benchmark is included below:

```
# OSU MPI_Put Bandwidth Test v5.8
# Window creation: MPI_Win_allocate
# Synchronization: MPI_Win_flush
# Size      Bandwidth (MB/s)
   1                 0.20
   2                 0.40
   4                 0.81
   8                 1.58
   16                2.89
   32                5.67
   64               10.22
   128              21.14
   256              40.65
   512              65.06
   1024             84.70
   2048             98.46
   4096            106.42
   8192            110.92
   16384           113.80
   32768           115.32
   65536           116.06
   131072          116.53
   262144          116.76
   524288          116.88
   1048576         116.92
   2097152         116.95
   4194304         116.96
```

The output from the benchmark is listed in MB/s. For easier comparison we will convert to Gb/s. Using the highest listed bandwidth output, convert as followed:

$$\frac{116.96\,MB}{1\,s} \text{ x } \frac{1\,GB}{1000\,MB} \text{ x } \frac{8\,Gb}{1\,GB} = \frac{.9357\,Gb}{1\,s}$$

The calculated .93Gb/s is close to the theoretical peak rate of 1Gb/s for the ethernet connection.

Intel MPI

The previous section validates performance of the default Ethernet connection using OpenMPI. Alternatively, the Intel OneAPI Toolkit can be used for the same purposes. Performance validation with the OneAPI Toolkit uses Intel MPI with IMB (Intel MPI Benchmarks), instead of openMPI with OSU benchmarks.

Install Intel MPI from the OneAPI Toolkit:

```
$> dnf config-manager --add-repo https://yum.repos.intel.com/oneapi
$> rpm --import https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
$> dnf install intel-oneapi-mpi-devel
```

The default installation for the MPI executables will be /opt/intel/oneapi/mpi/latest/bin, with the IMB benchmarks located at /opt/intel/oneapi/mpi/latest/benchmarks/imb. To save space on the diskless compute nodes, the Intel MPI folders on the head node will be shared with the compute nodes through NFS instead of being installed into the compute image.

```
$> echo "10.10.1.10:/opt/intel /opt/intel nfs nfsvers=3,nodev,nosuid 0 0" >> $CHROOT/etc/fstab
$> echo "/opt/intel *(ro,no_subtree_check,fsid=13)" >> /etc/exports
$> systemctl restart nfs-server
$> packimage centos8-x86_64-netboot-compute
$> pdsh -w c[1-2] reboot
```

Note the fsid=13. This number may need to be changed, depending on other folders shared. Check /etc/exports to see if FSID 13 has been reserved for a different folder. If it is, then change to the lowest number that is not being used.

Use the following sample script to execute one of the IMB RMA tests - PingPong - on two compute nodes:

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -J perf_test
#SBATCH -p normal
#SBATCH -t 20
#SBATCH -o imb_perf_test.out
#SBATCH -e imb_perf_test.err

source /opt/intel/oneapi/mpi/latest/env/vars.sh
mpirun -np 2 -ppn 1 IMB-P2P PingPong
```

After submitting the script with sbatch, a sample output is shown below:

```
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec  Msg/sec
            0       100000        49.81         0.00    20077
            1       100000        48.05         0.02    20813
            2       100000        48.35         0.04    20681
            4       100000        54.19         0.07    18454
            8       100000        56.44         0.14    17717
           16       100000        59.67         0.27    16759
           32       100000        60.38         0.53    16561
           64       100000        72.15         0.89    13860
          128       100000        88.67         1.44    11278
          256       100000        63.22         4.05    15818
          512       100000        62.63         8.18    15967
         1024       100000        62.91        16.28    15895
         2048       100000        69.48        29.48    14393
         4096       100000       126.80        32.30     7887
         8192       100000       189.69        43.19     5272
        16384        51200       231.20        70.87     4325
        32768        25600       388.93        84.25     2571
        65536        12800       625.94       104.70     1598
       131072         6400      1205.68       108.71      829
       262144         3200      2457.57       106.67      407
       524288         1600      4687.17       111.86      213
      1048576          800      9232.21       113.58      108
      2097152          400     18220.72       115.10       55
      4194304          200     36276.94       115.62       28
```

IMB output gives bandwidth in million bytes per second, which we can loosely convert as followed:

$$\frac{115.62\,MB}{1\,s} \;\text{x}\; \frac{1\,GB}{1000\,MB} \;\text{x}\; \frac{8\,Gb}{1\,GB} = \frac{.925\,Gb}{1\,s}$$

.925Gb/s is close to the theoretical peak rate of 1Gb/s for the ethernet connection.

High-Speed Ethernet

This section demonstrates performance validation using an add-in high-speed Ethernet adapter. The example uses the Intel XL710 40Gb QSFP+ Ethernet card, and assumes one card on each of the two compute nodes. A 40Gb card is not needed on the head node.

Configuration

Before validating the performance of high-speed Ethernet adapters, the adapters must be configured appropriately and a separate network must be created. Driver installation and device configuration must be performed as root.

Intel provides a high-level command-line utility that allows users to modify the configurations of installed QSFP+ adapters. Download the Intel QSFP+ Configuration Utility (QCU) to configure the 40Gb Ethernet card

```
$> wget https://downloadmirror.intel.com/25849/eng/QCU.zip
$> unzip ./QCU.zip
```

Before changing the configurations, we need to determine the location of the 40Gb Ethernet adapter – meaning, the name that the OS has assigned to the port(s). Note that the adapter may have more than one port assigned to it, depending on the default adapter configurations. More on this later. Use the OS lshw command to list all of the network devices:

```
$> pdsh -w c1 lshw -class network
```

Look for the 40GbE entry. The logical name for the interface will be listed.

```
*-network
     description: Ethernet interface
     product: Ethernet Controller XL710 for 40GbE QSFP+
     vendor: Intel Corporation
     physical id: 0
     bus info: pci@0000:61:00.0
     logical name: enp97s0
     version: 02
     serial: 3c:fd:fe:a0:3f:d0
     width: 64 bits
     clock: 33MHz
     capabilities: pm msi msix pciexpress vpd bus_master cap_list rom ethernet physical fibre
          autonegotiation
     configuration: autonegotiation=on broadcast=yes driver=i40e
          driverversion=4.18.0-305.19.1.el8_4.x86_64 duplex=full firmware=5.04 0x8000253b 1.1313.0
          latency=0 link=yes multicast=yes
     resources: irq:274 memory:c4800000-c4ffffff memory:c5000000-c5007fff memory:c5e00000-c5e7ffff
```

In this example, the 40Gb Ethernet interface is on device enp97s0. A device is not guaranteed to have the same name on similar HW, even when using the same PCIe slot on similarly configured systems. Therefore, we check the interface on the second compute node.

```
*-network
     description: Ethernet interface
     product: Ethernet Controller XL710 for 40GbE QSFP+
     vendor: Intel Corporation
     physical id: 0
     bus info: pci@0000:b6:00.0
     logical name: enp182s0
     version: 02
     serial: 3c:fd:fe:a0:43:b8
     width: 64 bits
     clock: 33MHz
     capabilities: pm msi msix pciexpress vpd bus_master cap_list rom ethernet physical fibre
          autonegotiation
     configuration: autonegotiation=on broadcast=yes driver=i40e
          driverversion=4.18.0-305.19.1.el8_4.x86_64 duplex=full firmware=5.04 0x8000253b 1.1313.0
          latency=0 link=yes multicast=yes
     resources: irq:188 memory:fa800000-faffffff memory:fb000000-fb007fff memory:fbe00000-fbe7ffff
```

Despite being installed in the same PCIe slot, the 40Gb Ethernet card has a different interface name on the second compute node. These logical names will be needed for xCAT configuration later. Fortunately, the QCU tool does not require the logical name for the device.

Use the QCU tool to check the current link configuration of the 40Gb Ethernet devices with the following command:

```
$> chmod +x ./QCU/Linux_x64/qcu64e
$> pdsh -w c[1-2] ./QCU/Linux_x64/qcu64e /NIC=1 /INFO
```

Sample output from the command is below:

```
Intel(R) QSFP+ Configuration Utility

QCU version: v2.34.17.03
Copyright(C) 2014 - 2019 by Intel Corporation.
Software released under Intel Proprietary License.

Adapter supports QSFP+ Configuration modification.
Current Configuration: 1x40

Supported Configurations:
 1x40
 4x10
```

For performance validation purposes, we want the configuration to be 1x40 - meaning one port is used, which performs at 40Gb/s. The 4X10 configuration sets 4 different QSFP+ ports (i.e 4 IP addresses) with each performing at 10Gb/s. In this example, the default use case is 1x40. If it shows 4x10 then change the operational mode to 1x40 using the QCU script:

```
$> ./Linux_x64/qcu64e /NIC=1 /SET 1x40
```

The 40Gb Ethernet devices are now configured. The next step is to make an entry in xCAT database for the network to be used by this Ethernet connection.

```
$> chdef -t network net40 net=10.10.40.0 mask=255.0.0.0
$> chdef c1 nicips.enp97s0="10.10.40.21" nicnetworks.enp97s0="net40" \
     nictypes.enp97s0="Ethernet" nichostnamesuffixes.enp97s0=-eth40
$> chdef c2 nicips.enp182s0="10.10.40.22" nicnetworks.enp182s0="net40" \
     nictypes.enp182s0="Ethernet" nichostnamesuffixes.enp182s0=-eth40
$> make networks
$> makehosts
```

Set the compute nodes to automatically configure this network device during OS booting.

```
$> chdef c1 -p postscripts="confignetwork"
$> chdef c2 -p postscripts="confignetwork"
```

To prevent the need to reboot nodes for settings to take effect, dynamically update the node network information with the following command:

```
$> updatenode c[1-2] -P "confignetwork -s"
```

Validate that the compute nodes can communicate with each other through the 40Gb Ethernet adapters by performing a ping command between the two compute nodes on the newly configured 40Gb network connection:

```
$> pdsh -w c1 ping -c 3 10.10.40.22
```

Successful pinging confirms the 40GbE connection is now ready for use.

Validation: Intel MPI

Validate the performance of the 40Gb Ethernet network using Intel MPI from the Intel OneAPI Toolkit. Setup instructions for OneAPI and Intel MPI are in the previous section on 1Gb Ethernet.

The same sample script to execute the IMB RMA PingPong test on two compute nodes is the same as the one used for 1Gb Ethernet:

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -J perf_test
#SBATCH -p normal
#SBATCH -t 20
#SBATCH -o imb_perf_test.out

source /opt/intel/oneapi/mpi/latest/env/vars.sh
mpirun -np 2 -ppn 1 IMB-P2P PingPong
```

Sample output is listed below:

```
#----------------------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#----------------------------------------------------------------
     #bytes #repetitions    t[usec]   Mbytes/sec    Msg/sec
          0       100000       0.44         0.00    2288056
          1       100000       0.46         2.16    2156700
          2       100000       0.47         4.21    2106393
          4       100000       0.47         8.45    2112387
          8       100000       0.47        17.13    2141056
         16       100000       0.47        34.22    2138933
         32       100000       0.50        63.94    1998005
         64       100000       0.49       130.97    2046425
        128       100000       0.60       213.86    1670760
        256       100000       0.64       398.45    1556442
        512       100000       0.88       578.80    1130465
       1024       100000       0.96      1070.76    1045659
       2048       100000       1.13      1817.04     887225
       4096       100000       1.58      2585.45     631214
       8192       100000       2.14      3827.98     467283
      16384        51200       3.47      4716.45     287869
      32768        25600       7.96      4117.18     125646
      65536        12800      14.55      4502.68      68705
     131072         6400      27.19      4820.95      36781
     262144         3200      52.57      4986.11      19021
     524288         1600     106.64      4916.06       9377
    1048576          800     251.05      4176.51       3983
    2097152          400     534.80      3920.91       1870
    4194304          200    1061.48      3950.47        942
```

IMB output gives bandwidth in million bytes per second, which we can loosely convert as followed:

$$\frac{4986.11\,MB}{1\,s} \times \frac{1\,GB}{1000\,MB} \times \frac{8\,Gb}{1\,GB} = \frac{39.89\,Gb}{1\,s}$$

While 39.89Gbs is near the theoretical peak of 40Gb/s, this is still a high number. The previous sections on 1Gb Ethernet shows the benchmark outputs reaching ~93% of the theoretical peak, as opposed to the 99% listed here. This is because MPI uses all IP networks that it finds, unless otherwise specified. In our scenario, both the 40Gb and 1Gb Ethernet connections were used to run the benchmark.

Omni-path

This section demonstrates installation and performance validation of the Omni-Path adapter. The reference cluster system used for these instructions assumes one Omni-Path 100HFA016 x16 host fabric interface 100 series (OPA100) adapter on two compute nodes with no other interconnect present besides the standard 1GbE connection for node management. The adapters use a point-to-point physical connection with no edge switch. The reference system also has a 100HFA016LS card on the head node to assist with visualzation of the installation and configuration instruction, but it is not required for successful installation or validation. In fact, there is no Omni-Path connection between the head and compute nodes.

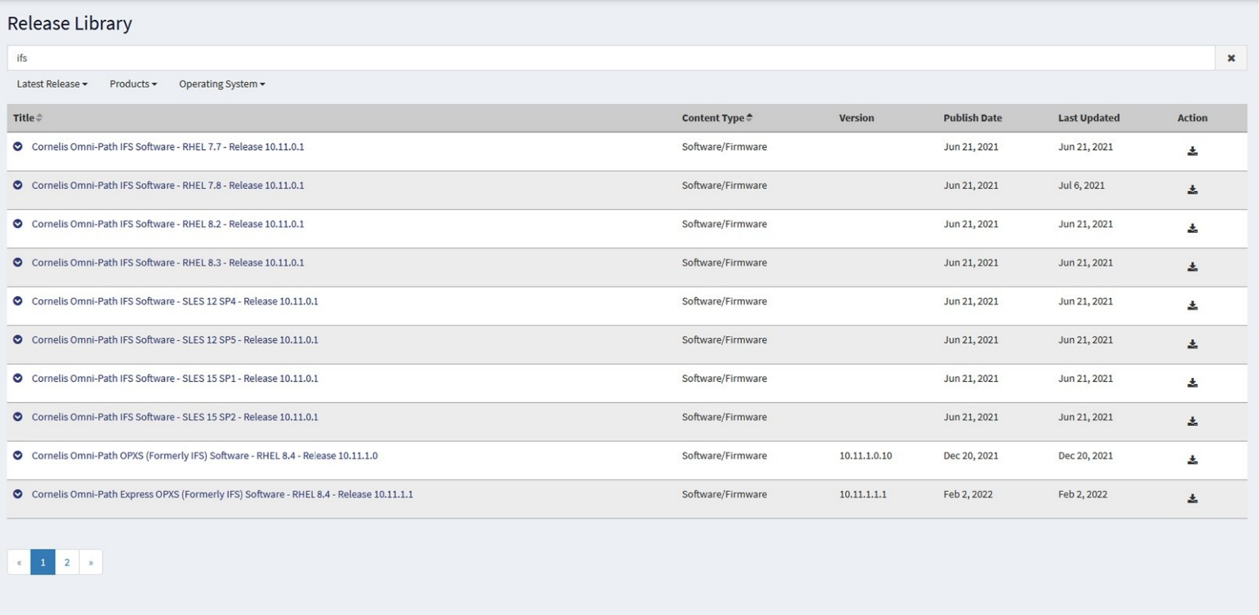
Instructions are for driver and software installation, and validate that the software and subnet manager is active. Performance tuning is beyond the scope of this document.

Configuration

There are multiple ways to install and configure the Omni-Path drivers. CentOS8.3 includes Omni-Path drivers and configuration tools, but the simplest method is through the Intel Fabric Suite (IFS). Unfortunately, this method is much harder to automate through scripting.

After Intel transferred ownership of the Omni-path product line to Cornelius Networks, IFS is no longer available for download on the Intel website. Acquiring IFS requires creating an account with Cornelius Networks and downloading from https://customercenter.cornelisnetworks.com/#/customer/assets/software-anddocumentation/release.

Newer CentOS releases - v8.4 and later - use the Cornelius Omni-Path Express Accelerated Host Fabric Suite (OPXS). For CentOS8.3, search for the legacy IFS (Intel Fabric Suite) and download the release for RHEL8.3.



Release Library

ifs ✖

Latest Release ▾ Products ▾ Operating System ▾

Title	Content Type	Version	Publish Date	Last Updated	Action
Cornelis Omni-Path IFS Software - RHEL 7.7 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path IFS Software - RHEL 7.8 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jul 6, 2021	⭳
Cornelis Omni-Path IFS Software - RHEL 8.2 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path IFS Software - RHEL 8.3 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path IFS Software - SLES 12 SP4 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path IFS Software - SLES 12 SP5 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path IFS Software - SLES 15 SP1 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path IFS Software - SLES 15 SP2 - Release 10.11.0.1	Software/Firmware		Jun 21, 2021	Jun 21, 2021	⭳
Cornelis Omni-Path OPXS (Formerly IFS) Software - RHEL 8.4 - Release 10.11.1.0	Software/Firmware	10.11.1.0.10	Dec 20, 2021	Dec 20, 2021	⭳
Cornelis Omni-Path Express OPXS (Formerly IFS) Software - RHEL 8.4 - Release 10.11.1.1	Software/Firmware	10.11.1.1.1	Feb 2, 2022	Feb 2, 2022	⭳

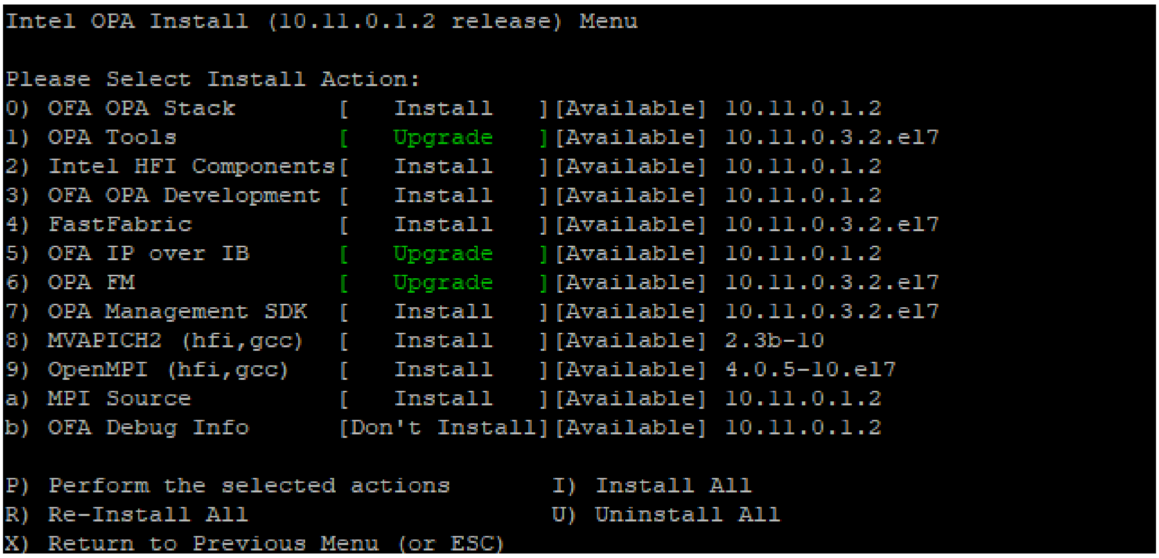
Before using the IFS, install packages required for driver support.

```
$> dnf install opensm-libs librdmacm libquadmath infinipath-psm libgfortran gcc-gfortran
$> dnf install rdma-core-devel atlas libatomic ibacm
```

Next, change the maximum amount of memory that a user process can lock. The default memory may be too low for HPC applications that use stateless nodes.

```
$> export CHROOT=/install/netboot/centos8.3/x86_64/compute/rootimg/
$> perl -pi -e 's/# End of file/\* soft memlock unlimited\n$&/s' /etc/security/limits.conf
$> perl -pi -e 's/# End of file/\* hard memlock unlimited\n$&/s' /etc/security/limits.conf
$> perl -pi -e 's/# End of file/\* soft memlock unlimited\n$&/s' $CHROOT/etc/security/limits.conf
$> perl -pi -e 's/# End of file/\* hard memlock unlimited\n$&/s' $CHROOT/etc/security/limits.conf
```

The IFS software is now ready to be installed on the head node. Un-tar the downloaded IFS file, switch to the un-tarred directory, then execute $> ./INSTALL as root



```
Intel OPA Install (10.11.0.1.2 release) Menu

Please Select Install Action:
0) OFA OPA Stack        [   Install   ][Available] 10.11.0.1.2
1) OPA Tools            [   Upgrade   ][Available] 10.11.0.3.2.el7
2) Intel HFI Components[   Install   ][Available] 10.11.0.1.2
3) OFA OPA Development [   Install   ][Available] 10.11.0.1.2
4) FastFabric           [   Install   ][Available] 10.11.0.3.2.el7
5) OFA IP over IB       [   Upgrade   ][Available] 10.11.0.1.2
6) OPA FM               [   Upgrade   ][Available] 10.11.0.3.2.el7
7) OPA Management SDK  [   Install   ][Available] 10.11.0.3.2.el7
8) MVAPICH2 (hfi,gcc)  [   Install   ][Available] 2.3b-10
9) OpenMPI (hfi,gcc)   [   Install   ][Available] 4.0.5-10.el7
a) MPI Source           [   Install   ][Available] 10.11.0.1.2
b) OFA Debug Info       [Don't Install][Available] 10.11.0.1.2

P) Perform the selected actions       I) Install All
R) Re-Install All                     U) Uninstall All
X) Return to Previous Menu (or ESC)
```

Choose to **Install All**, then **Perform the selected actions**.

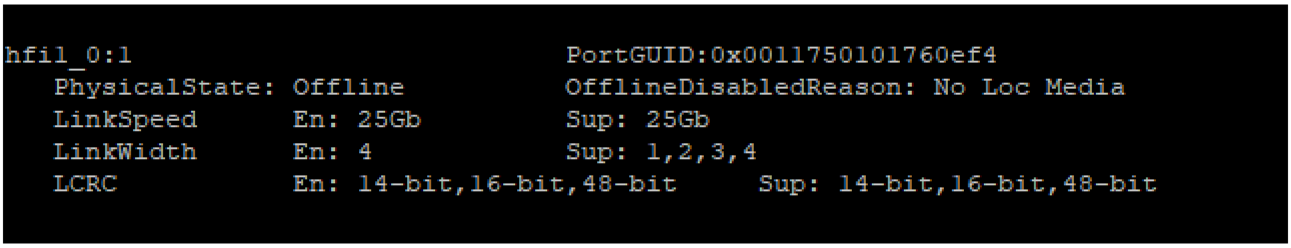
Set opafm (OPA Fabric Manager) to automatically start. This is service is required for systems that do not use an Omni-path edge switch. Even if a switch is part of the configuration, activating opafm allows the head node to act as a backup manager to the switch.

An alternative method to the previous steps of installing IFS through the ./INSTALL script is to install IFS as a repository using ./opacreaterepo script that comes with IFS, followed by using dnf install to choose which components are installed. This method in not chosen for this tutorial because it is more difficult for the user to understand which components are installed and why they are needed.

Verify the driver is installed and working.

```
$> lsmod | grep hfi
$> opainfo
```

The default performance setting should be a LinkSpeed of 25Gb with a LinkWidth of 4, giving an overall speed of 100Gb. This can be verified from output of the opainfo command:



```
hfil_0:1                          PortGUID:0x0011750101760ef4
   PhysicalState: Offline         OfflineDisabledReason: No Loc Media
   LinkSpeed      En: 25Gb        Sup: 25Gb
   LinkWidth      En: 4           Sup: 1,2,3,4
   LCRC           En: 14-bit,16-bit,48-bit     Sup: 14-bit,16-bit,48-bit
```

Recall that the system includes a 100HFA016LS card on the head node to assist with visualization of the installation instructions. Without the card, opainfo would have no output.

Enable opafm and the rdma service to start during system boot, then reboot.

```
$> opaconfig -E opafm
$> systemctl enable rdma
$> reboot
```

Create the Omni-path subnet and add network entries for each compute node to the xCAT networks table, then set the compute nodes to automatically configure the OPA network device during each boot.

```
$> chdef -t network -o ib0 mask=255.0.0.0 net=10.10.100.0
$> makenetworks
$> chdef c1 nicips.ib0=10.10.100.21 nictypes.ib0='InfiniBand' nicnetworks.ib0=ib0
$> chdef c2 nicips.ib0=10.10.100.22 nictypes.ib0='InfiniBand' nicnetworks.ib0=ib0
$> chdef c1 -p postscripts="confignetwork"
$> chdef c2 -p postscripts="confignetwork"
```

Add <hostname>-ib0 to /etc/hosts and update DNS records.

```
$> echo "10.10.100.10 quartz-ib0 quartz-ib0.aau" >> /etc/hosts
$> makehosts
$> makedns -n
```

Update the newly edited system configuration on the compute nodes, then reboot to confirm that the software is active.

```
$> packimage centos8-x86_64-netboot-compute
$> pdsh -w c1,c2 reboot
```

Finally, ensure the compute nodes can communicate with each other through the Omni-path subnet. Recall that the head node has no Omni-path connection to the compute nodes. This means verification of the Omni-path subnet must be done by remotely sending a ping command to each of the compute nodes.

```
$> pdsh -w c1 ping -c 3 10.10.100.22
$> pdsh -w c2 ping -c 3 10.10.100.21
```

A successful ping from both nodes means the Omni-path network is ready for use.

Validation

Validate the performance of the Omni-Path network using Intel MPI from the Intel OneAPI Toolkit. Setup instructions for OneAPI and Intel MPI are in the previous section on 1Gb Ethernet. Use the following sample script to execute the IMB PingPong test on two compute nodes using the Omni-path connection:

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -J perf_test
#SBATCH -p normal
#SBATCH -t 10
#SBATCH -o imb_perf_test.out
#SBATCH -e imb_perf_test.err

source /opt/intel/oneapi/mpi/latest/env/vars.sh
mpirun -genv I_MPI_FABRICS shm:tmi -genv I_MPI_TMI_PROVIDER psm2 -np 2 -ppn 1 IMB-P2P PingPong
```

By default, IMPI selects the fastest fabric available. Therefore, I_MPI_FABRICS is not needed because the sample system contains only an Omni-Path interconnect. The arguments for mpirun explicitly set the fabric in the sample script above. After submitting the script with sbatch, a sample output is shown below:

```
#-----------------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#-----------------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec      Msg/sec
            0       100000         0.20         0.00      5050899
            1       100000         0.32         3.16      3155241
            2       100000         0.33         6.09      3044198
            4       100000         0.26        15.43      3856682
            8       100000         0.32        25.20      3149664
           16       100000         0.32        49.40      3087682
           32       100000         0.28       114.16      3567410
           64       100000         0.32       197.44      3084945
          128       100000         0.33       389.46      3042624
          256       100000         0.31       836.69      3268327
          512       100000         0.41      1244.55      2430763
         1024       100000         0.47      2162.80      2112109
         2048       100000         0.59      3479.44      1698947
         4096       100000         0.85      4839.55      1181531
         8192       100000         1.45      5668.69       691979
        16384        51200         2.30      7123.85       434805
        32768        25600         3.79      8656.41       264173
        65536        12800         6.64      9863.68       150508
       131072         6400        14.24      9203.05        70214
       262144         3200        27.18      9645.47        36795
       524288         1600        56.50      9278.72        17698
      1048576          800       112.05      9357.37         8924
      2097152          400       316.20      6631.69         3162
      4194304          200       753.04      5567.74         1327
```

IMB output gives bandwidth in million bytes per second, which we can convert using following formula:

$$\frac{9863.68\,MB}{1\,s} \times \frac{1\,GB}{1000\,MB} \times \frac{8\,Gb}{1\,GB} = \frac{78.91\,Gb}{1\,s}$$

78.91Gb/s is much less than the theoretical peak rate of 100Gb/s for the Omni-Path connection. This is because, as mentioned earlier, performance optimizations such as fabric tuning and process binding is beyond the scope of this document.

InfiniBand

This section demonstrates configuration and performance validation of a point-to-point InfiniBand network. All instructions are based on the Mellanox ConnectX-4 CX455A 100Gb/s EDR InfiniBand card and assumes one InfiniBand card on each of the two compute nodes with no other high-speed interconnects on the system – specifically, Omni-Path. For this tutorial, no InfiniBand card is used on the head node. The compute nodes are connected as point-to-point with no InfiniBand switch. All configuration steps are perform as root.

Configuration

Configuration will use open-source InfiniBand drivers and tools available in the CentOS distribution source, instead of OFED drivers or other proprietary modules. Driver installation and device configuration must be performed as root.

First, install the CentOS InfiniBand package, which provides the tools and software for supporting RDMA-based InfiniBand fabrics.

```
$> dnf -y groupinstall "Infiniband Support"
$> yum-config-manager --installroot=$CHROOT --enable base
$> dnf -y --installroot=$CHROOT groupinstall "Infiniband Support"
```

This package includes not only the necessary modules for functionality such as the user space library (libibverbs), but also supplementary tools for several InfiniBand functions such as performance tests (perftest),and checking the health of the fabric (inifiniband-diags).

Some installed packages are not needed and should be removed to prevent potential conflicts. One example is the OPA-related packages.

```
$> dnf remove opa-address-resolution opa-libopamgt opa-fastfabric
$> dnf --installroot=$CHROOT remove opa-address-resolution opa-libopamgt opa-fastfabric
```

InfiniBand networks require a subnet manager. This usually comes with the InfiniBand switch. Our network uses point-to-point connections with no InfiniBand switch, therefore requiring a 3rd party subnet manager. The CentOS ‘Infiniband Support’ package includes an open-source tool for switch-less subnet management: openSM. Enable the rdma service and the switch management service on both the head node and compute nodes.

```
$> systemctl enable opensm
$> systemctl enable rdma
$> systemctl restart opensm
$> systemctl restart rdma
$> chroot $CHROOT systemctl enable opensm
$> chroot $CHROOT systemctl enable rdma
```

The xCAT database needs to recognize and configure the InfiniBand cards on the compute nodes. Similar to high-speed Ethernet, the InfiniBand adapters must be configured appropriately and a separate network must be created in the cluster environment.

```
$> chdef -t network -o ib0 mask=255.0.0.0 net=10.10.100.0 mgtifname=ib0
$> chdef c1 nicips.ib0=10.10.100.21 nictypes.ib0="Infiniband" nicnetworks.ib0=ib0
$> chdef c2 nicips.ib0=10.10.100.22 nictypes.ib0="Infiniband" nicnetworks.ib0=ib0
$> echo 10.10.100.10 quartz-ib0 quartz-ib0.aau >> /etc/hosts
$> makehosts
$> makenetworks
$> makedhcp -n
$> makedns -n
```

xCAT includes a script - configiba - that configures IB during boot. Additionally, as with all external network cards, the confignics script is required. Set the compute nodes to execute these scripts during node bring-up.

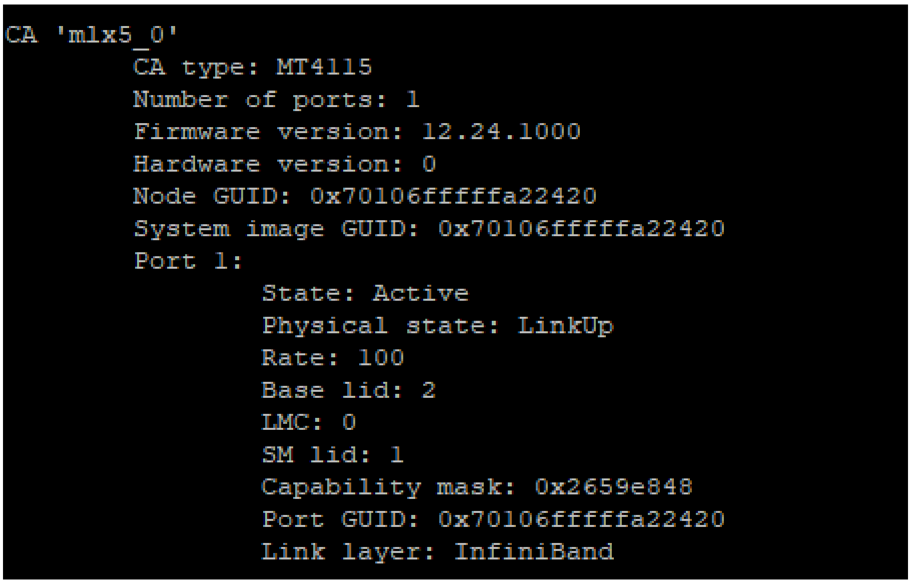
```
$> chdef c1 -p postbootscripts=confignics
$> chdef c2 -p postbootscripts=confignics
$> chdef c1 -p postscripts=configiba
$> chdef c2 -p postscripts=configiba
```

Initialize the node image, then reboot the nodes.

```
$> packimage centos8-x86_64-netboot-compute
$> pdsh -w c[1-2] reboot
```

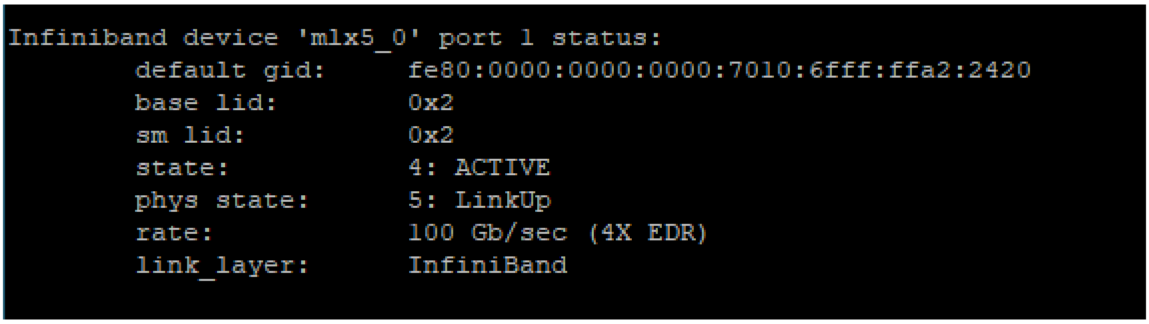
Now that the InfiniBand tools are installed on the head and compute nodes, we can use some of the built-in commands from the infiniband-diags package to gather more information about the InfiniBand hardware and network.

```
$> pdsh -w c1 ibstat
```



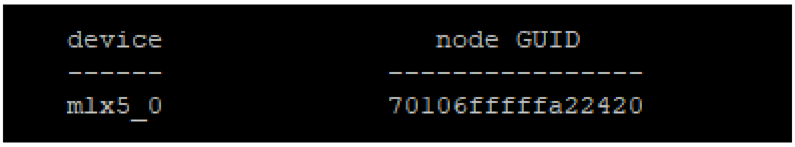
```
CA 'mlx5_0'
        CA type: MT4115
        Number of ports: 1
        Firmware version: 12.24.1000
        Hardware version: 0
        Node GUID: 0x70106fffffa22420
        System image GUID: 0x70106fffffa22420
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 100
                Base lid: 2
                LMC: 0
                SM lid: 1
                Capability mask: 0x2659e848
                Port GUID: 0x70106fffffa22420
                Link layer: InfiniBand
```

```
$> pdsh -w c1 ibstatus
```



```
Infiniband device 'mlx5_0' port 1 status:
        default gid:     fe80:0000:0000:0000:7010:6fff:ffa2:2420
        base lid:        0x2
        sm lid:          0x2
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            100 Gb/sec (4X EDR)
        link_layer:      InfiniBand
```

```
$> pdsh -w c1 ibv_devices
```



```
    device                 node GUID
    ------              ----------------
    mlx5_0              70106fffffa22420
```

As we see from the list of RDMA devices from the ibv_devices command, the InfiniBand device is identified by the name mlx5_0, the ibstatus command shows the InfinBand card is set at 100 Gb/s, and ibstat shows the InfiniBand device is set as 'Active'. The InfiniBand network is now ready for use.

Validation

Similar to high-speed Ethernet, the Infiniband adapters must be configured appropriately and a separate network must be created. Driver installation and device configuration must be performed as root.

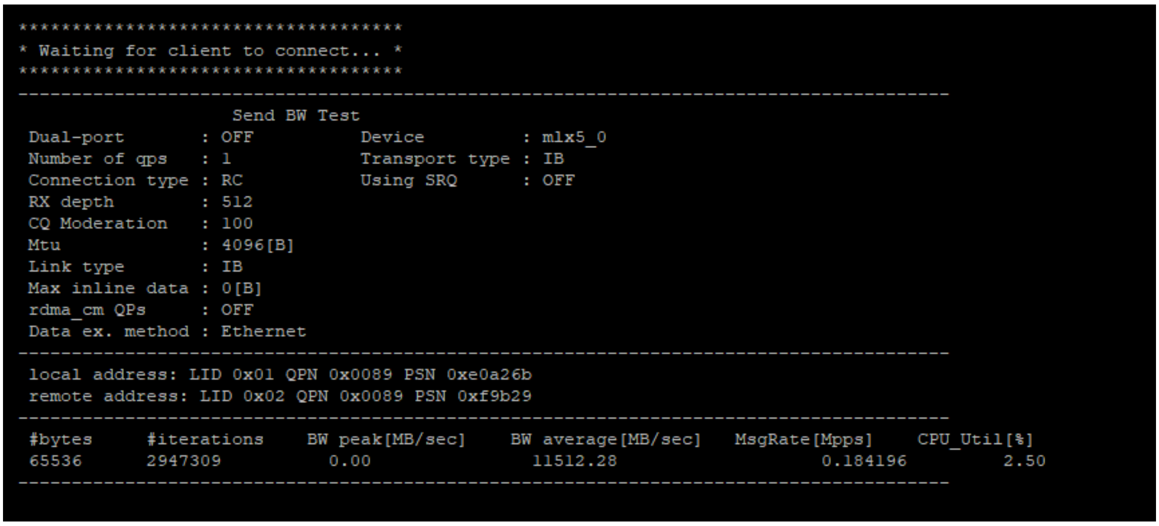
The previously installed InfiniBand Support packages includes perftest, a collection of performance micro-benchmarks. This section will demonstrate validation using perftest benchmarks instead of OSU or Intel benchmarks from previous sections.

Run the uni-directional bandwidth test for send.

```
$> pdsh -w c1 ib_send_bw -d mlx5_0 -D 30 –cpu_util
$> pdsh -w c2 ib_send_bw -d mlx5_0 -D 30 –cpu_util c1
```

The -D flag is duration, in second; 30 seconds here. The flag -d tells the network device to use. As seen in the previous section, the InfiniBand device is listed as mlx5_0.

To perform this test, the command is send to both compute nodes. Here, node c1 waits for node c2 to connect. The test begins when both bodes are ready to connect, with results displayed on both compute nodes.



```
************************************
* Waiting for client to connect... *
************************************
---------------------------------------------------------------------------------------
                    Send BW Test
 Dual-port       : OFF          Device         : mlx5_0
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 RX depth        : 512
 CQ Moderation   : 100
 Mtu             : 4096[B]
 Link type       : IB
 Max inline data : 0[B]
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0x01 QPN 0x0089 PSN 0xe0a26b
 remote address: LID 0x02 QPN 0x0089 PSN 0xf9b29
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[MB/sec]    BW average[MB/sec]   MsgRate[Mpps]    CPU_Util[%]
 65536      2947309          0.00               11512.28                   0.184196         2.50
---------------------------------------------------------------------------------------
```

Similar to the IMB benchmarks, bandwidth output is displayed as million bytes per second, which we can convert using following formula:

$$\frac{11512.28\,MB}{1\,s} \text{ x } \frac{1\,GB}{1000\,MB} \text{ x } \frac{8\,Gb}{1\,GB} = \frac{92.09\,Gb}{1\,s}$$

92.09Gb/s is within range of the theoretical peak rate for this 100Gb/s InfiniBand connection. As mentioned earlier, no performance optimizations were attempted.