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 1.1

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

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:

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

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:

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

.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 to automatically configure this network device after 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. Using only one of the interfaces requires additional arguments to mpirun. An example of this will be demonstrated in the next section using OpenMPI.