Technical Advisory

Omni-Path Software

Issue: Optimizing NCCL for Use with Omni-Path

Rev. 1.0

February 2021

Distribution: All Users

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Cornelis Networks products described herein. You agree to grant Cornelis Networks a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cornelis Networks technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Cornelis Networks and the Cornelis Networks logo belong to Cornelis Networks, Inc. Other names and brands may be claimed as the property of others.

Contents

1 Description

1.1 Issue

The NVIDIA* Collectives Communications Library (NCCL*) does not come with an optimized transport for Omni-Path. Use of the Verbs transport does not take maximum advantage of the OPA HFI ASIC and GPUDirect* support that is available with Omni-Path Architecture (OPA) PSM2 and hfi1.

1.2 Implication

NCCL is a library from NVIDIA for collectives-heavy workloads on GPU-equipped servers. A NCCL job can be run through an MPI such as OpenMPI. The job can be run across multiple servers, in which case NCCL will use a network transport for communicating between servers.

NCCL workloads on OPA fabrics may not achieve maximum performance using only OFI or Verbs.

2 Corrective Action/Resolution

Install the Cornelis Networks PSM2-NCCL to achieve maximum performance.

For additional information, refer to the *Omni-Path Software Release Notes*.

2.1 Known Limitations

- This beta should not be used for jobs with more than 64 ranks at this time. Jobs with more than 64 ranks may abort during initialization. See Section 2.7 Setting User Contexts Per Node.
- For correct operation, the following PSM2 configuration variables should be set as indicated:

  PSM2_CUDA=1

  PSM2_GPUDIRECT=1

  PSM2_MULTI_EP=1

  PSM2_GDRCOPY=0

2.2 Prerequisites

1. RHEL* 8.2 and onward
2. An OPA fabric with NVIDIA GPUs in 2 or more nodes
3. IFS 10.10.3.1.1 or later, installed with GPUDirect components
4. PSM2 source code with NCCL support (https://github.com/cornelisnetworks/opa-psm2/tree/PSM2_11.2.NCCL)
5. CUDA 10.2 with NVIDIA driver installed on all GPU-equipped nodes
6. NCCL source code (https://github.com/NVIDIA/nccl. This beta has been tested against NCCL 2.8.3-1 https://github.com/NVIDIA/nccl/tree/v2.8.3-1/src)
7. OpenMPI 4.0.3 or later

   You may use OpenMPI with or without CUDA support. Refer to the following NVIDIA advisory about mixing NCCL and MPI operations:

   https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/mpi.html#inter-gpu-communication-with-cuda-aware-mpi
8. psm2-nccl code (https://github.com/cornelisnetworks/psm2-nccl)
9. (Optional) nccl-tests (https://github.com/NVIDIA/nccl-tests) for verifying NCCL and PSM2-NCCL plugin are working after installation

2.3 Building

1. Build PSM2 with changes (see Prerequisite #4) to support PSM2-NCCL from source and install on all applicable GPU nodes. Ensure that you build libpsm2 with CUDA and GPUDirect support.

   Refer to the PSM2 README for build instructions.

2. Clone the nccl source code from GitHub (see Prerequisites #6) to a host with PSM2, CUDA installed.

3. Change directory (`cd`) to your nccl clone.

4. Build NCCL rpms using:

   ```
   make -j $(nproc) pkg.redhat.build
   ```

5. Install rpms found under the nccl clone on all applicable GPU nodes.

6. Clone psm2-nccl code (see Prerequisites #8) and place it in same directory as your nccl clone.

   You should put your psm2-nccl clone in the same parent directory as your nccl clone because psm2-nccl includes header files from nccl that are not included in the NCCL rpms.

   If your nccl and psm2-nccl clones are not in the same directory, set the value of *`NCCL_SRC_DIR`* in the psm2-nccl/Makefile to point to your nccl clone.

7. Change directory (`cd`) into your psm2-nccl clone.

8. Edit the Makefile to change *`BUILDDIR`* (if desired) and then execute `make`.

2.4 Deploying

1. Copy `libnccl-net.so` produced in the previous section to same location on all nodes in the fabric or shared mount that all ranks in NCCL job will be able to access.

2. If the location you chose is not a standard library directory, ensure that *`LD_LIBRARY_PATH`* is updated correctly.

2.5 Testing

1. Clone nccl-tests from GitHub (see Prerequisites #9).

2. Source OpenMPI `bin/mpivars.sh` file.

3. Change directory (`cd`) into nccl-tests repo.

4. Enter:

   ```
   make MPI=1 MPI_HOME=${MPI_ROOT} CUDA_HOME=/usr/local/cuda
   ```

nccl-tests test-programs will be under `nccl-tests/build/`

5. Run a test program (such as *`all_reduce_perf`*) and verify that the psm2_nccl plugin is being used. For example:

```
mpirun -np 2 --map-by node -host host1,host2 --timeout 90 -x
NCCL_DEBUG=INFO -x PSM2_GDRCOPY=0 -x PSM2_CUDA=1 -x PSM2_GPUDIRECT=1
-x PSM2_MULTI_EP=1  -x LD_LIBRARY_PATH nccl-
tests/build/all_reduce_perf
```

   The output should include `Using network psm2-nccl` for each rank. For example:

```
host1:#####:##### [0] NCCL INFO Using network psm2-nccl
host2:#####:##### [0] NCCL INFO Using network psm2-nccl
```

6. If performance is not improved, NCCL may not be enabling GPUDirect for PSM2-NCCL. Add the following to your `mpirun` line to force NCCL to use GPUDirect regardless of GPU-HFI distance:

```
-x NCCL_NET_GDR_LEVEL=5
```

2.6 Running

This task provides instructions for running a NCCL application in OpenMPI.

1. Add the path where `libnccl-net.so` was deployed to *`LD_LIBRARY_PATH`* environment variable.

   For example:

```
export LD_LIBRARY_PATH=${USER}/install/lib:${LD_LIBRARY_PATH}
```

2. Add the following line to your OpenMPI `mpirun` command line:

```
-x LD_LIBRARY_PATH -x PSM2_GDRCOPY=0 -x PSM2_CUDA=1 -x
PSM2_GPUDIRECT=1 -x PSM2_MULTI_EP=1
```

3. Run your application with OpenMPI `mpirun`.

2.7 Setting User Contexts Per Node

Note: The size of a NCCL job (in ranks) is limited to no more than the number of PSM2 endpoints divided by 2. The maximum number of PSM2 endpoints is determined by the number of hfi1 contexts. The number of hfi1 contexts is typically equal to the number of physical CPU cores on server but can be set up to 160.

1. To set the number of user contexts per node:

   a. Edit `/etc/modprobe.d/hfi1.conf`.

b. Add the following hfi1 driver parameter settings to the "options hfi1" line.

```
num_user_contexts=X
```

where *X* can be one of the following:

X=16 – This value provides the best performance when running a few MPI ranks per node (16 or fewer), such as when using a Hybrid programming model with many more threads than MPI ranks.

X="Number of physical cores per node" (Default setting) – This value is used if you do not plan on running more MPI ranks than cores.

X=2x "Number of physical cores per node" – This value is used if you plan on running an MPI rank-count up to 2x the number of cores on a node (assuming Hyper-Threading is turned on).

2. Determine if dracut needs to be run:

   If the following sequence happens, run the dracut command as described in Step 3.

   a. At the start of boot, initramfs is all that is visible.

   b. The hfi1 driver is loaded while only the initramfs is visible.

   c. The hfi1.conf file within the initramfs is used.

      If one of the following happens, then the dracut command is not needed. Skip to Step 4.

      — If you reload the driver while the OS is running, the initramfs is not used.

      — If the hfi1 driver is loaded after the initramfs stops being used, then the initramfs is not used.

3. Run the /usr/bin/dracut -f command to force `/etc/modprobe.d/hf1.conf` into the initramfs image.

```
$ /usr/bin/dracut -f
```

4. Reboot the system.