NVIDIA COLLECTIVE COMMUNICATION LIBRARY (NCCL)

RN-08645-210_v01 | December 2017

Release Notes

TABLE OF CONTENTS

Chapter 1. NCCL OVERVIEW

The NVIDIA® Collective Communications Library ™ (NCCL) (pronounced "Nickel") is a library of multi-GPU collective communication primitives that are topology-aware and can be easily integrated into applications.

Collective communication algorithms employ many processors working in concert to aggregate data. NCCL is not a full-blown parallel programming framework; rather, it is a library focused on accelerating collective communication primitives. The following collective operations are currently supported:

- **`AllReduce`**
- **`Broadcast`**
- **`Reduce`**
- **`AllGather`**
- **`ReduceScatter`**

Tight synchronization between communicating processors is a key aspect of collective communication. Compute Unified Device Architecture® (CUDA) based collectives would traditionally be realized through a combination of CUDA memory copy operations and CUDA kernels for local reductions. NCCL, on the other hand, implements each collective in a single kernel handling both communication and computation operations. This allows for fast synchronization and minimizes the resources needed to reach peak bandwidth.

NCCL conveniently removes the need for developers to optimize their applications for specific machines. NCCL provides fast collectives over multiple GPUs both within and across nodes. It supports a variety of interconnect technologies including PCIe, NVLink™ , InfiniBand Verbs, and IP sockets. NCCL also automatically patterns its communication strategy to match the system's underlying GPU interconnect topology.

Next to performance, ease of programming was the primary consideration in the design of NCCL. NCCL uses a simple C API, which can be easily accessed from a variety of programming languages. NCCL closely follows the popular collectives API defined by MPI (Message Passing Interface). Anyone familiar with MPI will thus find NCCL API very natural to use. In a minor departure from MPI, NCCL collectives take a "stream" argument which provides direct integration with the CUDA programming model.

Finally, NCCL is compatible with virtually any multi-GPU parallelization model, for example:

- single-threaded
- multi-threaded, for example, using one thread per GPU
- multi-process, for example, MPI combined with multi-threaded operation on GPUs

NCCL has found great application in deep learning frameworks, where the **`AllReduce`** collective is heavily used for neural network training. Efficient scaling of neural network training is possible with the multi-GPU and multi node communication provided by NCCL.

Chapter 2. NCCL RELEASE 2.1.2

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- New algorithms for improved latency communication
- RoCE support

Using NCCL 2.1.2

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.x API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.

Known Issues

- If NCCL returns an error code, set the environment variable **`NCCL_DEBUG to WARN`** to receive an explicit error message.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause **`ncclCommInitRank`** to fail while establishing IPCs (**`cudaIpcOpenMemHandle`**). This problem does not appear when using only processes or only threads.
- NCCL uses CUDA 9 cooperative group launch by default, which may induce increased latencies in multi-threaded programs. See the **`NCCL_LAUNCH_MODE`** knob in the NCCL Developer Guide to restore the original behavior.

Fixed Issues

- NCCL now uses CUDA 9 cooperative groups to launch the CUDA kernels, fixing a long-standing issue with CUDA and NCCL operations being interleaved and potentially causing hangs.

Chapter 3.
NCCL RELEASE 2.0.5

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- NCCL 2.0.5 provides support for intra-node and inter-node communication.
- NCCL optimizes intra-node communication using NVLink, PCI express, and shared memory.
- Between nodes, NCCL implements fast transfers over sockets or InfiniBand verbs.
- GPU-to-GPU and GPU-to-Network direct transfers, using the GPU Direct technology, are extensively used when the hardware topology permits it.

Using NCCL 2.0.5

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.0 API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- NCCL 2.0.5 has the new configuration file support. The NCCL environment variables can now be set in **`~/.nccl.conf`** and **`/etc/nccl.conf`**.
- Values defined in **`~/.nccl.conf`** take precedence over ones in **`/etc/nccl.conf`**.
- The syntax for each line of the NCCL configuration file is **`<NCCL_VAR_NAME>=<VALUE>`**.

Known Issues

- If NCCL returns any error code, set the environment variable `NCCL_DEBUG` to **`WARN`** to receive an explicit error message.
- RoCE support is experimental.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while

establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.

Chapter 4. NCCL RELEASE 2.0.4

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- NCCL 2.0.4 provides support for intra-node and inter-node communication.
- NCCL optimizes intra-node communication using NVLink, PCI express, and shared memory.
- Between nodes, NCCL implements fast transfers over sockets or InfiniBand verbs.
- GPU-to-GPU and GPU-to-Network direct transfers, using the GPU Direct technology, are extensively used when the hardware topology permits it.

Using NCCL 2.0.4

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.0 API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- NCCL 2.0.4 has the new configuration file support. The NCCL environment variables can now be set in **`~/.nccl.conf`** and **`/etc/nccl.conf`**.
- Values defined in **`~/.nccl.conf`** take precedence over ones in **`/etc/nccl.conf`**.
- The syntax for each line of the NCCL configuration file is **`<NCCL_VAR_NAME>=<VALUE>`**.

Known Issues

- If NCCL returns any error code, set the environment variable `NCCL_DEBUG` to **`WARN`** to receive an explicit error message.
- RoCE is not supported.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while

establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.

Chapter 5. NCCL RELEASE 2.0.2

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- NCCL 2.0.2 provides support for intra-node and inter-node communication.
- NCCL optimizes intra-node communication using NVLink, PCI express, and shared memory.
- Between nodes, NCCL implements fast transfers over sockets or InfiniBand verbs.
- GPU-to-GPU and GPU-to-Network direct transfers, using the GPU Direct technology, is extensively used when the hardware topology permits it.

Using NCCL 2.0.2

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.0 API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.

Known Issues

- NCCL 2.0.2 is known to not work with CUDA driver 384.40 and later.
- If NCCL returns any error code, set the environment variable `NCCL_DEBUG` to **`WARN`** to receive an explicit error message.
- NCCL 2.0.2 does not support RoCE, that is, InfiniBand cards using Ethernet as link layer. The presence of an RoCE card on a node will make NCCL fail even when run within the node.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.

Notice

Trademarks

Copyright

www.nvidia.com