Dual-DAC Interconnect Configuration Guide for DGX Spark Systems

Introduction

This document provides a reference configuration for establishing a redundant direct-attached interconnect between two NVIDIA DGX Spark systems using **dual DAC cables**.

The procedure includes hardware topology considerations, software environment preparation, NCCL compilation, interface selection, IP configuration, and communication validation.

DGX Spark systems support direct, point-to-point deployment without data-center infrastructure such as switches or rack enclosures. While suitable for flexible office environments, such environments may introduce physical-layer risks including dust exposure, connector wear caused by repeated insertion/removal, and cable bending. These factors can potentially reduce link reliability.

1. Interconnect Architecture

1.1. Recommended Baseline Topology

NVIDIA documentation specifies that a **single DAC cable** provides sufficient bandwidth for direct two-node DGX Spark connectivity.

1.2. Rationale for Dual-DAC Deployment

To enhance physical-layer availability, a second DAC cable of identical specification may be installed.

- The redundant cable: Does **not** modify functional bandwidth
- Supplements the primary data path
- Improves overall link robustness
- Reduces risk of communication failure caused by physical-layer degradation

1.3. Scope of This Document

This guide demonstrates an example dual-DAC configuration and includes:

- NCCL build procedure with Blackwell architecture support
- NCCL performance validation commands
- Network interface selection criteria
- IP addressing practices
- End-to-end communication verification

2. NCCL Software Environment Setup

2.1. Building NCCL with Blackwell Architecture Support

Execute the following commands on both nodes to build NCCL from source with support for Blackwell architecture:

```
# Install dependencies and build NCCL
sudo apt-get update && sudo apt-get install -y libopenmpi-dev
git clone -b v2.28.3-1 https://github.com/NVIDIA/nccl.git ~/nccl/
cd ~/nccl/
make -j src.build NVCC_GENCODE="-gencode=arch=compute_121,code=sm_121"
```

2.2. Setting Environment Variables

```
# Set environment variables
export CUDA_HOME="/usr/local/cuda"
export MPI_HOME="/usr/lib/aarch64-linux-gnu/openmpi"
export NCCL_HOME="$HOME/nccl/build/"
export LD_LIBRARY_PATH="$NCCL_HOME/lib:$CUDA_HOME/lib64/:$MPI_HOME/lib:$LD_LIBRARY_PATH"
```

3. Building the NCCL Test Suite

```
git clone https://github.com/NVIDIA/nccl-tests.git ~/nccl-tests/
cd ~/nccl-tests/
make MPI=1
```

4. Network Interface Configuration

4.1. Identifying Active Interfaces

Run the following command to list active network interfaces:

```
# Check network port status
ibdev2netdev
```

Example output:

```
rocep1s0f0 port 1 ==> enp1s0f0np0 (Up)
rocep1s0f1 port 1 ==> enp1s0f1np1 (Up)
roceP2p1s0f0 port 1 ==> enP2p1s0f0np0 (Up)
roceP2p1s0f1 port 1 ==> enP2p1s0f1np1 (Up)
```

4.2. CPU Group Selection Criteria

Note:

Select interfaces mapped to different CPU groups to avoid resource contention and maintain optimal communication throughput.

In this example:

- **rocep1s0f0** and **rocep1s0f1** map to the same CPU group
- **roceP2p1s0f0** and **roceP2p1s0f1** map to a different CPU group
- Accordingly, **enp1s0f0np0** and **enP2p1s0f1np1** are selected for configuration

4.3. Obtain IP Address Assignments

Retrieve IP addresses from the selected interfaces:

```
ip addr show enp1s0f0np0
ip addr show enP2p1s0f1np1
```

Example output:

```
3: enp1s0f0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 4c:bb:47:2f:bc:b3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.9/30 scope global enp1s0f0np0
       valid_lft forever preferred_lft forever

6: enP2p1s0f1np1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 4c:bb:47:2f:bc:b8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.102.9/30 scope global enP2p1s0f1np1
       valid_lft forever preferred_lft forever
```

Ensure the interfaces reside on **distinct IP subnets**.
Repeat the same process on the second node.

5. Communication Validation

5.1. Execute the NCCL Communication Test

Run the following command to validate communication between nodes.
Replace the interface names and IP addresses with the values obtained earlier:

```
mpirun --allow-run-as-root -np 2 -H <IP for Node 1>:1,<IP for Node 2>:1 \
--mca btl_tcp_if_include enP7s7 \
--mca plm_rsh_agent "ssh -x -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no" \
-x LD_LIBRARY_PATH=$LD_LIBRARY_PATH \
-x UCX_NET_DEVICES=enp1s0f0np0,enP2p1s0f1np1 \
-x NCCL_SOCKET_IFNAME=enp1s0f0np0,enP2p1s0f1np1 \
-x NCCL_SOCKET_FAMILY=AF_INET \
-x NCCL_IB_HCA=rocep1s0f0,roceP2p1s0f1 \
-x NCCL_IB_TC=3 \
-x NCCL_IB_MERGE_NICS=1 \
-x NCCL_MAX_NCHANNELS=16 \
-x NCCL_MIN_NCHANNELS=16 \
-x NCCL_ALGO=Ring \
$HOME/nccl-tests/build/all_gather_perf -b 512M -e 8G -f 2
```

5.2. Expected Output

```
# nccl-tests version 2.17.6 nccl-headers=22803 nccl-library=22803
# Collective test starting: all_gather_perf
# nThread 1 nGpus 1 minBytes 536870912 maxBytes 8589934592
# step: 2(factor) warmup iters: 1 iters: 20 agg iters: 1 validation: 1 graph: 0
#
# Using devices
#  Rank  0 Group  0 Pid  30868 on spark-bcb2 device  0 [000f:01:00] NVIDIA GB10
#  Rank  1 Group  0 Pid  36414 on spark-c77a device  0 [000f:01:00] NVIDIA GB10
#
#                                                     out-of-place                  in-place
#      size       count     type   redop    root    time   algbw   busbw  #wrong    time   algbw   busbw  #wrong
#       (B)    (elements)                               (us)  (GB/s)  (GB/s)           (us)  (GB/s)  (GB/s)
   536870912     67108864    float    none      -1  16769.5   32.01   16.01       0  15822.9   33.93   16.96       0
  1073741824    134217728    float    none      -1  31030.2   34.60   17.30       0  29206.3   36.76   18.38       0
  2147483648    268435456    float    none      -1  58664.8   36.61
18.30       0  54589.3   39.34   19.67       0
  4294967296    536870912    float    none      -1   111552   38.50   19.25       0   104803   40.98   20.49       0
  8589934592   1073741824    float    none      -1   218506   39.31   19.66       0   207334   41.43   20.72       0
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 18.6741
#
# Collective test concluded: all_gather_perf
```

6. Next Steps

Your NCCL environment is now ready for multi-node distributed training on DGX Spark systems. You may proceed with running larger distributed workloads, such as TRT-LLM or vLLM inference, to validate end-to-end operation.