Mellanox GPUDirect RDMA User Manual

Rev 1.0

www.mellanox.com

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

Table of Contents

Table of Contents 3
List Of Tables 4
Chapter 1 Overview 5
1.1 System Requirements 5
1.2 Important Notes 5
Chapter 2 Installing GPUDirect RDMA 6
Chapter 3 Benchmark Tests 7
3.1 Testing GPUDirect RDMA with CUDA-Enabled Benchmark 7
3.2 Running GPUDirect RDMA with MVAPICH-GDR 2.0b 7
3.3 Running GPUDirect RDMA with OpenMPI 1.7.4 8

List Of Tables

Table 1: GPUDirect RDMA System Requirements 5

1 Overview

GPUDirect RDMA is an API between IB CORE and peer memory clients, such as NVIDIA Kepler class GPU's. It provides access for the HCA to read/write peer memory data buffers, as a result it allows RDMA-based applications to use the peer device computing power with the RDMA interconnect without the need to copy data to host memory. This capability is supported with Mellanox ConnectX®-3 VPI or Connect-IB® InfiniBand adapters. It will also work seamlessly using RoCE technology with the Mellanox ConnectX®-3 VPI adapters.

1.1 System Requirements

The platform and server requirements for GPUDirect RDMA are detailed in the following table:

Table 1 - GPUDirect RDMA System Requirements

Platform	Type and Version
HCAs	• Mellanox ConnectX®-3 • Mellanox ConnectX®-3 Pro • Mellanox Connect-IB® • NVIDIA® Tesla™ K-Series (K10, K20, K40)
Software/Plugins	• MLNX_OFED v2.1-x.x.x or later www.mellanox.com -> Products -> Software - > InfiniBand/VPI Drivers -> Linux SW/ Drivers • Plugin module to enable GPUDirect RDMA www.mellanox.com -> Products -> Software - > InfiniBand/VPI Drivers -> GPUDirect RDMA • NVIDIA Driver 331.20 or later http://www.nvidia.com/Download/index.aspx?lang=en-us • NVIDIA CUDA Runtime and Toolkit 6.0 https://developer.nvidia.com/cuda-downloadsservice

1.2 Important Notes

- Once the hardware and software components are installed, it is important to check that the GPUDirect kernel module is properly loaded on each of the compute systems where you plan to run the job that requires the GPUDirect RDMA feature.

  To check:

  ```
  service nv_peer_mem status
  ```

  Or for some other flavors of Linux:

  ```
  lsmod | grep nv_peer_mem
  ```

  Usually this kernel module is set to load by default by the system startup service. If not loaded, GPUDirect RDMA would not work, which would result in very high latency for message communications.

  One you start the module by either:

  ```
  service nv_peer_mem start
  ```

  Or for some other flavors of Linux:

  ```
  modprobe nv_peer_mem
  ```

- To achieve the best performance for GPUDirect RDMA, it is required that both the HCA and the GPU be physically located on the same PCIe IO root complex.

  To find out about the system architecture, either review the system manual, or run "lspci -tv".

2 Installing GPUDirect RDMA

- ***To install GPUDirect RDMA (excluding ubuntu):***

```
rpmbuild --rebuild <path to srpm>
rpm -ivh <path to generated binary rpm file>
```

Note: On SLES OSes add "`--nodeps`".

- ***To install GPUDirect RDMA on Ubuntu:***

Copy the tarball to a temporary directory.

```
tar xzf <tarball>
cd <extracted directory>
dpkg-buildpackage -us -uc
dpkg -i <path to generated deb files>
```

Example:

```
dpkg -i nvidia-peer-memory_1.0-0_all.deb
dpkg -i nvidia-peer-memory-dkms_1.0-0_all.deb
```

Please make sure this kernel module is installed and loaded on each GPU InfiniBand compute nodes.

3 Benchmark Tests

3.1 Testing GPUDirect RDMA with CUDA-Enabled Benchmark

GPUDirect RDMA can be tested by running the micro-benchmarks from Ohio State University (OSU). The OSU benchmarks 4 and above are CUDA-enabled benchmarks that can downloaded from: http://mvapich.cse.ohio-state.edu/benchmarks/

When building the OSU benchmarks, you must verify that the proper flags are set to enable the CUDA part of the tests, otherwise the tests will only run using the host memory instead which is the default.

```
./configure CC=/path/to/mpicc \
--enable-cuda \
--with-cuda-include=/path/to/cuda/include \
--with-cuda-libpath=/path/to/cuda/lib
make
make install
```

3.2 Running GPUDirect RDMA with MVAPICH-GDR 2.0b

MVAPICH2 that takes advantage of the new GPUDirect RDMA technology for inter-node data movement on NVIDIA GPUs clusters with Mellanox InfiniBand interconnect.

MVAPICH-GDR 2.0b, can be downloaded from:
http://mvapich.cse.ohio-state.edu/download/mvapich2gdr/

Below is an example of running one of the OSU benchmark which enables GPUDirect RDMA.

```
[gdr@ops001 ~]$ mpirun_rsh -np 2 ops001 ops002 MV2_USE_CUDA=1 MV2_USE_GPUDIRECT=1 /home/gdr/
osu-micro-benchmarks-4.2-mvapich2/mpi/pt2pt/osu_bw -d cuda D D
# OSU MPI-CUDA Bandwidth Test v4.2
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size      Bandwidth (MB/s)
...
2097152              6372.60
4194304              6388.63
```

The MV2_GPUDIRECT_LIMIT is a tunable parameter which controls the buffer size that it starts to use.

Here is a list of runtime parameters that can be used for process-to-rail binding in case the system has multi-rail configuration:

```
export MV2_USE_CUDA=1
export MV2_USE_GPUDIRECT=1
export MV2_RAIL_SHARING_POLICY=FIXED_MAPPING
export MV2_PROCESS_TO_RAIL_MAPPING=mlx5_0:mlx5_1
export MV2_RAIL_SHARING_LARGE_MSG_THRESHOLD=1G
export MV2_CPU_BINDING_LEVEL=SOCKET
export MV2_CPU_BINDING_POLICY=SCATTER
```

Additional tuning parameters related to CUDA and GPUDirect RDMA (such as MV2_CUDA_BLOCK_SIZE) can be found in the README installed on the node:

/opt/mvapich2/gdr/2.0/gnu/share/doc/mvapich2-gdr-gnu-2.0/README-GDR

3.3 Running GPUDirect RDMA with OpenMPI 1.7.4

The GPUDirect RDMA support is available on OpenMPI 1.7.4rc1. Unlike MVAPICH2-GDR which is available in the RPM format, one can download the source code for OpenMPI and compile using flags below to enable GPUDirect RDMA support:

```
[co-mell1@login-sand8 ~]$ ../configure --prefix=/path/to/openmpi-1.7.4rc1/install \
--with-wrapper-ldflags=-Wl,-rpath,/lib --disable-vt --enable-orterun-prefix-by-default -dis-
able-io-romio --enable-picky \
--with-cuda=/usr/local/cuda-5.5 \
--with-cuda-include=/usr/local/cuda-6.0/include \
--with-cuda-libpath=/usr/local/cuda-6.0/lib64
[co-mell1@login-sand8 ~]$ make; make install
```

To run the OpenMPI that uses the flag that enables GPUDirect RDMA:

```
[gdr@jupiter001 ~]$ mpirun -mca btl_openib_want_cuda_gdr 1 -np 2 -npernode 1 -x
LD_LIBRARY_PATH -mca btl_openib_if_include mlx5_0:1 -bind-to-core -report-bindings -mca
coll_fca_enable 0 -x CUDA_VISIBLE_DEVICES=0 /home/co-mell1/scratch/osu-micro-benchmarks-4.2/
install/libexec/osu-micro-benchmarks/mpi/pt2pt/osu_latency -d cuda D D
# OSU MPI-CUDA Latency Test v4.2
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size          Latency (us)
0                       1.08
1                       3.83
2                       3.83
4                       3.84
8                       3.83
16                      3.83
32                      3.82
64                      3.80
...
```

If the flag for GPUDirect RDMA is not enabled, it would result in much higher latency for the above.

By default in OpenMPI 1.7.4, the GPUDirect RDMA will work for message sizes between 0 to 30KB. For messages above that limit, it will be switched to use asynchronous copies through the host memory instead. Sometimes, better application performance can be seen by adjusting that limit. Here is an example of increasing to adjust the switch over point to above 64KB:

```
-mca btl_openib_cuda_rdma_limit 65537
```