OPEN MPI WITH RDMA SUPPORT AND CUDA

Rolf vandeVaart, NVIDIA
OVERVIEW

- What is CUDA-aware
- History of CUDA-aware support in Open MPI
- GPU Direct RDMA support
- Tuning parameters
- Application example
- Future work
CUDA-AWARE DEFINITION

Regular MPI

//MPI rank 0
cudaMemcpy(s_buf_h,s_buf_d,size,...);
MPI_Send(s_buf_h,size,...);

//MPI rank n-1
MPI_Recv(r_buf_h,size,...);
cudaMemcpy(r_buf_d,r_buf_h,size,...);

CUDA-aware MPI

//MPI rank 0
MPI_Send(s_buf_d,size,...);

//MPI rank n-1
MPI_Recv(r_buf_d,size,...);
cudaMemcpy(r_buf_d,r_buf_h,size,...);

CUDA-aware MPI makes MPI+CUDA easier.
CUDA-AWARE MPI IMPLEMENTATIONS

- Open MPI 1.7.4
  
  http://www.open-mpi.org

- MVAPICH2 2.0
  
  http://mvapich.cse.ohio-state.edu/overview/mvapich2

- IBM Platform MPI 9.1.2
  

- CRAY MPT
HISTORY OF CUDA-AWARE OPEN MPI

- Open MPI 1.7.0 released in April, 2013
- Pipelining with host memory staging for large messages (utilizing asynchronous copies) over verbs layer
- Dynamic CUDA IPC support added
- GPU Direct RDMA support
- Use Open MPI 1.7.4 to get all the latest features
MPI API SUPPORT

- Yes
  - All send and receive types
  - All non-arithmetic collectives

- No
  - Reduction type operations - MPI_Reduce, MPI_Allreduce, MPI_Scan
  - Non-blocking collectives
  - MPI-2 and MPI-3 (one sided) RMA

- FAQ will be updated as support changes
CUDA IPC SUPPORT IN OPEN MPI

NODE 0

INTEL CPU

RANK 0
GPU
CUDA IPC

RANK 1
GPU

QPI

INTEL CPU

RANK 2
GPU

NODE 1

INTEL CPU

RANK 3
GPU
CUDA IPC

RANK 4
CUDA IPC SUPPORT IN OPEN MPI

- Open MPI dynamically detects if CUDA IPC is supported between GPUs within the same node.
- Enabled by default
  --mca btl_smcuda_use_cuda_ipc 0
- To see if it is being used between two processes
  --mca btl_smcuda_cuda_ipc_verbose 100
- CUDA 6.0 has performance fixes
CUDA IPC SUPPORT IN OPEN MPI

Inter GPU Bandwidth

- Message Size (bytes)
- CUDA 6.0 RC, Open MPI 1.7.4
- Intel Xeon E5-2690 v2@3 Ghz
- Tesla K20m (PCIe Gen 2)

Intra GPU Bandwidth (with MPS)

- Message Size (bytes)
- CUDA 6.0 RC, Open MPI 1.7.4
- Intel Xeon E5-2690v2@3Ghz
- Tesla K20m (PCIe Gen 2)
GPU DIRECT RDMA SUPPORT

- Kepler class GPUs (K10, K20, K40)
- Mellanox ConnectX-3, ConnectX-3 Pro, Connect-IB
- CUDA 6.0 (EA, RC, Final), Open MPI 1.7.4 and Mellanox OFED 2.1 drivers.
- GPU Direct RDMA enabling software
  [http://www.mellanox.com/downloads/ofed/nvidia_peer_memory-1.0-0.tar.gz](http://www.mellanox.com/downloads/ofed/nvidia_peer_memory-1.0-0.tar.gz)
GPU DIRECT RDMA SUPPORT

- Implement with RDMA type protocol
- Register send and receive buffers and have NIC transfer data
- Memory registration is not cheap - need to have registration cache
GPU DIRECT RDMA SUPPORT

- Chipset implementations limit bandwidth at larger message sizes
- Still use pipelining with host memory staging for large messages (utilizing asynchronous copies)
- Final implementation is hybrid of both protocols

IVY BRIDGE
P2P READ: 3.5 Gbyte/sec
P2P WRITE: 6.4 Gbyte/sec
GPU DIRECT RDMA SUPPORT - PERFORMANCE

Small Message Latency

- No-GDR
- GDR
- Hybrid

Mellanox OFED 2.1, CUDA 6.0 RC, Open MPI 1.7.4
Intel Xeon E5-2690 v2 @ 3Ghz node with 20 cores
NVIDIA Tesla K40m GPU
Mellanox Connect-IB HCA FDR 56Gb/s PCIe3.0x16
GPU DIRECT RDMA SUPPORT - PERFORMANCE

Bandwidth

Max 3.5 Gb/sec read BW

Message Size (bytes)

Mellanox OFED 2.1, CUDA 6.0 RC, Open MPI 1.7.4
Intel Xeon E5-2690v2@3Ghz (20 cores)
Mellanox Connect-IB HCA FDR 56 GB/s PCIe3.0x16
GPU DIRECT RDMA SUPPORT - CONFIGURE

- Nothing different needs to be done at configure time
- `> configure --with-cuda`

The support is configured in if CUDA 6.0 cuda.h header file is detected.

To check:

- `> ompi_info --all | grep btl_openib_have_cuda_gdr`
  
  MCA btl: informational "btl_openib_have_cuda_gdr" (current value: "true", data source: default, level: 4 tuner/basic, type: bool)

- `> ompi_info -all | grep btl_openib_have_driver_gdr`
  
  MCA btl: informational "btl_openib_have_driver_gdr" (current value: "true", data source: default, level: 4 tuner/basic, type: bool)
GPU DIRECT RDMA SUPPORT - TUNING PARAMETERS

- Runtime parameters

  - Enable GPU Direct RDMA usage (off by default)
    - `--mca btl_openib.want_cuda_gdr 1`

  - Adjust when we switch to pipeline transfers through host memory. Current default is 30,000 bytes
    - `--mca btl_openib_cuda_rdma_limit 60000`
GPU DIRECT RDMA SUPPORT - NUMA ISSUES

- Configure system so GPU and NIC are close

![Diagram showing the configuration of GPU, CPU, and NIC with QPI connections. The diagram on the left shows a configuration where GPU and NIC are not close, while the diagram on the right shows a better configuration where GPU and NIC are close.]
Multi NIC - multi GPU: use hwloc to select GPU near NIC
LATENCY COMPARISON - CLOSE VS FAR GPU AND NIC

Open MPI 1.7.4 Latency over GPU
Direct RDMA

Latency (usec)

Message Size (bytes)
CUDA-AWARE OPEN MPI AND UNIFIED MEMORY

- CUDA 6 Unified Memory
  
  `cudaMallocManaged(buf, BUFSIZE, cudaMemAttachGlobal)`

- Unified Memory may not work correctly with CUDA-aware Open MPI

- Will fix in future release of Open MPI
HOOMD BLUE PERFORMANCE

- **Highly Optimized Object-oriented Many-particle Dynamics - Blue Edition**
  - Performs general purpose particle dynamics simulations
  - Takes advantage of NVIDIA GPU
  - Simulations are configured and run using simple python scripts
  - The development effort is led by Glotzer group at University of Michigan

HOOMD - CLUSTER 1

- Dell™ PowerEdge™ R720xd/R720 cluster
  - Dual-Socket Octa-core Intel E5-2680 V2 @ 2.80 GHz CPUs (Static max Perf in BIOS), Memory: 64GB DDR3 1600 MHz Dual Rank Memory Module, OS: RHEL 6.2, MLNX_OFED 2.1-1.0.0 InfiniBand SW stack

- Mellanox Connect-IB FDR InfiniBand, Mellanox SwitchX SX6036 InfiniBand VPI switch, NVIDIA® Tesla K40 GPUs (1 GPU per node), NVIDIA® CUDA® 5.5 Development Tools and Display Driver 331.20, Open MPI 1.7.4 rc1, GPUDirect RDMA (nvidia_peer_memory-1.0-0.tar.gz)

- Application: HOOMD-blue (git master 28Jan14), Benchmark datasets: Lennard-Jones Liquid Benchmarks (16K, 64K Particles)

GPU DIRECT RDMA SUPPORT - APPLICATION PERFORMANCE

Higher is better

**HOOMD-blue Performance**
(LJ Liquid Benchmark, 16K Particles)

<table>
<thead>
<tr>
<th>Number of Nodes</th>
<th>Without GPUDirect RDMA</th>
<th>GPUDirect RDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2500</td>
<td>3000</td>
</tr>
<tr>
<td>2</td>
<td>2200</td>
<td>2600</td>
</tr>
<tr>
<td>3</td>
<td>2100</td>
<td>2500</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>2400</td>
</tr>
</tbody>
</table>

**HOOMD-blue Performance**
(LJ Liquid Benchmark, 64K Particles)

<table>
<thead>
<tr>
<th>Number of Nodes</th>
<th>Without GPUDirect RDMA</th>
<th>GPUDirect RDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5000</td>
<td>5500</td>
</tr>
<tr>
<td>2</td>
<td>4800</td>
<td>5200</td>
</tr>
<tr>
<td>3</td>
<td>4700</td>
<td>5100</td>
</tr>
<tr>
<td>4</td>
<td>4600</td>
<td>5000</td>
</tr>
</tbody>
</table>

HOOMD - CLUSTER 2

- Dell™ PowerEdge™ T620 128-node (1536-core) Wilkes cluster at Univ of Cambridge
  - Dual-Socket Hexa-Core Intel E5-2630 v2 @ 2.60 GHz CPUs, Memory: 64GB memory, DDR3 1600 MHz, OS: Scientific Linux release 6.4 (Carbon), MLNX_OFED 2.1-1.0.0 InfiniBand SW stack

- Mellanox Connect-IB FDR InfiniBand adapters, Mellanox SwitchX SX6036 InfiniBand VPI switch, NVIDIA® Tesla K20 GPUs (2 GPUs per node), NVIDIA® CUDA® 5.5 Development Tools and Display Driver 331.20, Open MPI 1.7.4rc1, GPUDirect RDMA (nvidia_peer_memory-1.0-0.tar.gz)

- Application: HOOMD-blue (git master 28Jan14)

- Benchmark datasets: Lennard-Jones Liquid Benchmarks (512K Particles)

GPU DIRECT RDMA SUPPORT - APPLICATION PERFORMANCE

OPEN MPI CUDA-AWARE

- Work continues to improve performance
  Better interactions with streams
  Better small message performance - eager protocol
  CUDA-aware support for new RMA
  Support for reduction operations
  Support for non-blocking collectives

- Try it!
- Lots of information in FAQ at the Open MPI web site
- Questions?