Coupling GPUDirect RDMA and InfiniBand Hardware Multicast Technologies for Streaming Applications

GPU Technology Conference GTC 2016

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda
Streaming Applications

- Examples - surveillance, habitat monitoring, etc..
- Require efficient transport of data from/to distributed sources/sinks
- Sensitive to latency and throughput metrics

- Require HPC resources to efficiently carry out compute-intensive tasks
Nature of Streaming Applications

• Pipelined data parallel compute phases that form the crux of streaming applications lend themselves for GPGPUs
• Data distribution to GPGPU sites occur over PCIe within the node and over InfiniBand interconnects across nodes

• Broadcast operation is a key dictator of throughput of streaming applications
• Reduced latency for each operation
• Support multiple back-to-back operations

Drivers of Modern HPC Cluster Architectures

- Multi-core processors are ubiquitous
- InfiniBand very popular in HPC clusters
- Accelerators/Coprocessors becoming common in high-end systems
  - Pushing the envelope for Exascale computing
Large-scale InfiniBand Installations

- 235 IB Clusters (47%) in the Nov’ 2015 Top500 list

(http://www.top500.org)

- Installations in the Top 50 (21 systems):

| 462,462 cores (Stampede) at TACC (10th) | 76,032 cores (Tsubame 2.5) at Japan/GSIC (25th) |
| 185,344 cores (Pleiades) at NASA/Ames (13th) | 194,616 cores (Cascade) at PNNL (27th) |
| 72,800 cores Cray CS-Storm in US (15th) | 76,032 cores (Makman-2) at Saudi Aramco (32nd) |
| 72,800 cores Cray CS-Storm in US (16th) | 110,400 cores (Pangea) in France (33rd) |
| 265,440 cores SGI ICE at Tulip Trading Australia (17th) | 37,120 cores (Lomonosov-2) at Russia/MSU (35th) |
| 124,200 cores (Topaz) SGI ICE at ERDC DSRC in US (18th) | 57,600 cores (SwiftLucy) in US (37th) |
| 72,000 cores (HPC2) in Italy (19th) | 55,728 cores (Prometheus) at Poland/Cyfronet (38th) |
| 152,692 cores (Thunder) at AFRL/USA (21st) | 50,544 cores (Occigen) at France/GENCI-CINES (43rd) |
| 147,456 cores (SuperMUC) in Germany (22nd) | 76,896 cores (Salomon) SGI ICE in Czech Republic (47th) |
| 86,016 cores (SuperMUC Phase 2) in Germany (24th) | and many more! |
InfiniBand Networking Technology

• Introduced in Oct 2000
• High Performance Point-to-point Data Transfer
  – Interprocessor communication and I/O
  – Low latency (<1.0 microsec), High bandwidth (up to 12.5 GigaBytes/sec -> 100Gbps), and low CPU utilization (5-10%)
• Multiple Features
  – Offloaded Send/Recv
  – RDMA Read/Write
  – Atomic Operations
  – Hardware Multicast support through Unreliable Datagram (UD)
    • A message sent from a single source (host memory) can reach all destinations (host memory) in a single pass over the network through switch-based replication
    • Restricted to one MTU
    • Large messages need to be sent in a chunked manner
    • Unreliable, Reliability needs to be addressed
• Leading to big changes in designing HPC clusters, file systems, cloud computing systems, grid computing systems, ....
InfiniBand Hardware Multicast Example

Switch decodes inbound packet header (LRH) DLID to determine target output ports.

Router decodes inbound packet header (GRH) GID multicast address to determine target out-
Multicast-aware CPU-Based MPI_Bcast on Stampede using MVAPICH2
(6K nodes with 102K cores)

ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCIe Gen3 with Mellanox IB FDR switch
GPUDirect RDMA (GDR) and CUDA-Aware MPI

- Before CUDA 4: Additional copies
  - Low performance and low productivity
- After CUDA 4: Host-based pipeline
  - Unified Virtual Address
  - Pipeline CUDA copies with IB transfers
  - High performance and high productivity
- After CUDA 5.5: GPUDirect-RDMA support
  - GPU to GPU direct transfer
  - Bypass the host memory
  - Hybrid design to avoid PCI bottlenecks
Performance of MVAPICH2-GPU with GPU-Direct RDMA (GDR)

**GPU-GPU Internode Latency**

- MV2-GDR2.2b
- MV2-GDR2.0b
- MV2 w/o GDR

**GPU-GPU Internode Bandwidth**

- MV2-GDR2.2b
- MV2-GDR2.0b
- MV2 w/o GDR

**GPU-GPU Internode Bi-Bandwidth**

- MV2-GDR2.2b
- MV2-GDR2.0b
- MV2 w/o GDR

More details in 2:30pm session today
Broadcasting Data from One GPU Memory to Other GPU Memory: Shortcomings

- Traditional short message broadcast operation between GPU buffers involves a Host-Staged Multicast (HSM)
- Data copied from GPU buffers to host memory
- Using InfiniBand Unreliable Datagram (UD)-based hardware multicast
- Sub-optimal use of near-scale invariant UD-multicast performance
- PCIe resources wasted and benefits of multicast nullified
- GPUDirect RDMA capabilities unused
Problem Statement

- Can we design a new GPU broadcast scheme that can deliver low latency for streaming applications?
- Can we combine GDR and IB MCAST features to
  - Achieve the best performance
  - Free the Host-Device PCIe bandwidth for application needs
- Can such design be extended to support heterogeneous configurations?
  - Host-to-Device
    - Camera connected to host and devices used for computation
  - Device-to-device
  - Device-to-Host
- How to support such a design on systems with multiple GPUs/node?
- How much performance benefits can be achieved with the new designs?
Existing Protocol for GPU Multicast

• Copy user GPU data to host buffers
• Perform Multicast and copy back

• CudaMemcpy dictates performance
• Requires PCIe Host-Device resources
Alternative Approaches

- Can we substitute the cudaMemcpy with a better design?
  - cudaMemcpy: Default Scheme
    - Big overhead for small message
  - Loopback-based design: Uses GDR feature
    - Process establishes self-connection
    - Copy H-D $\Rightarrow$ RDMA write (H, D)
    - Copy D-H $\Rightarrow$ RDMA write (D, H)
    - P2P bottleneck $\Rightarrow$ good for small and medium sizes
  - GDRCOPY-based design: New module for fast copies
    - Involves GPU PCIe BAR1 mapping
    - CPU performing the copy $\Rightarrow$ block until completion
    - Very good performance for H-D for small and medium sizes
    - Very good performance for D-H only for very small sizes
GDRCOPY-based design

- Copy user GPU data to host buffers
- Perform Multicast and copy back

- D-H operation limits performance
- Can we avoid GDRCOPY for D-H copies?
(GDRCOPY + Loopback)-based design

- Copy user GPU data to host buffers using loopback scheme
- Perform Multicast
- Copy back the data to GPU using GDRCOPY scheme

- Good performance for both H-D and D-H copies
- Expected performance only for small message
- Still using the PCIe H-D resources
Experimental Setup and Details of Benchmarks

- Experiments were run on Wilkes @ University of Cambridge
  - 12-core IvyBridge Intel(R) Xeon(R) E5-2630 @ 2.60 GHz with 64 GB RAM
  - FDR ConnectX2 HCAs + NVIDIA K20c GPUs
  - Mellanox OFED version MLNX OFED LINUX-2.1-1.0.6 which supports GPUDirect RDMA (GDR) required
  - Use only one GPU and one HCA per node (same socket) configuration

- Based on latest MVAPICH2-GDR 2.1 release
  (http://mvapich.cse.ohio-state.edu/downloads)

- Use OSU MicroBenchmark test suit
  - osu_bcast benchmark
  - A modified version mimicking back-to-back broadcasts
Performance of Naive (CudaMemcpy-based) scheme

- Big overhead for small messages due to the overhead of the copies
- Good scalability with small messages as it is true MCAST based

Big overhead, 20 µs for 1 Byte

Reasonably good for large
Performance of GDRCOPY-based scheme

- Achieves 3 µs for small message broadcast
- GDRCOPY D-H operation has big overhead for large message

Very good performance

Limited D-H performance
Performance of LoopBack-based scheme

- Achieves less than 6 µs for small message broadcast
- Uses IB LoopBack path for both D-H and H-D copies
- Sender and receiver might share the same network bandwidth

Acceptable/Good performance
Performance of Hybrid (GDRCOPY+Loopback+Naïve) scheme

Switch to loopback design

- Takes advantage of the best of each scheme: Loopback for D-H and GDRCOPY for H-D
- Good scalability up to 64 GPU system
Comparing Different Schemes

- Up to 3X performance improvement
- Good scalability
- However, all these schemes still use Host-based staging ⇒ Use PCIe Host-Device resources
Can we do better?

- Can we have enhanced designs that:
  - Delivers good performance (low latency for throughput broadcast operations)
  - Frees PCIe host-device resources
  - Provides good support for all message sizes (small and large)
Challenges in Combining GDR and MCAST Features

- How to handle control messages and data which belong to two different memories (control on Host, data on GPU)?

- How to efficiently handle multi-GPU configurations

- How to handle reliability as MCAST is UD-based transport? ⇒ Can we provide MPI_Bcast semantic support?
Combining GDR and MCAST Features: Scatter-Gather List (SGL) Approach

• MCAST two separate addresses (control on the host + data on GPU) in one IB message

• Direct IB read/write from/to GPU using GDR feature for low latency ⇒ Zerocopy based schemes

• MCAST feature to provide scalability ⇒ Switch based message duplication

• No extra copy between Host and GPU ⇒ frees-up PCIe resource for application needs

Overview of the envisioned SGL-based approach

• One time registration of window of persistent buffers in streaming apps

• *Gather* control and user data at the source and scatter them at the destinations using *Scatter-Gather-List* abstraction

• Scheme lends itself for pipelined phases abundant in Streaming Applications and avoids stressing PCIe
• SGL-based design is able to deliver:
  • Low latency and high scalability (less than 4us)
  • Free PCIe resource for applications
Benefits of SGL-based design with Streaming Benchmark

- HSM (Host Staged), GSM (GPU Staged) = SGL
- Based on a synthetic benchmark that mimics broadcast patterns in Streaming Applications
- Long window of persistent m-byte buffers with 1,000 back-to-back multicast operations issued
- Execution time reduces by 3x-4x
Limits of SGL-based Broadcast Designs

• Limit the support to only Device to Device broadcast
  • Requires the copy from the host to device at the source
  • Big overhead of the copy
  • Breaks the pipeline view of the streaming application

• Not scalable for multi-GPUs nodes
  • Flat design with one-to-one MCAST connection for each GPU
On-going Work

• Can MCAST+GDR be combined for heterogeneous configurations?
  - Source on the Host and destination on Device
  - Heterogeneity: Control+Data are contiguous on one side and non-contiguous on other side
  - Combine MCAST and GDR => No use of PCIe resources (free for application usage)

• How about multi-GPU nodes? Can intra-node topology-awareness help?
  - Hierarchical and complex PCIe interconnects
  - How to maximize the resource utilization of both PCIe and IB interconnects?

• Looking forward: Solution should benefits current generation systems and maximal benefits for next-generation systems
Conclusions

• IB MCAST feature provides high scalability and low latency
• GDR feature provides a direct access between IB and GPUs
• MVAPICH2-GDR provides several schemes to efficiently broadcast from/to GPU memories using host staged techniques
  • Naïve design + Host-based MCAST
  • GDRCOPY + Host-based MCAST
  • GDRCOPY + Loopback + Host-based MCAST
• Presented a set of designs to couple GDR and IB MCAST features
• Results are promising
• Designs need to be extended to support heterogeneity and multi-GPU support
• New designs will be available in future MVAPICH2-GDR library
Two Additional Talks

- **S6411 - MVAPICH2-GDR: Pushing the Frontier of Designing MPI Libraries Enabling GPUDirect Technologies**
  - **Day:** Wednesday, 04/06
  - **Time:** 14:30 - 14:55
  - **Location:** Room 211A

- **S6418 - Bringing NVIDIA GPUs to the PGAS/OpenSHMEM World: Challenges and Solutions**
  - **Day:** Wednesday, 04/06
  - **Time:** 16:30 - 16:55
  - **Location:** Room 211A
Thank You!

panda@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/