MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

High Performance Broadcast with GPUDirect RDMA and InfiniBand Hardware Multicast for Streaming Applications

GTC 2015

MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Presented By

Dhabaleswar K. (DK) Panda

The Ohio State University

Email: panda@cse.ohio-state.edu

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

- Introduction
- Motivation and Problem Statement
- Design Considerations
- Proposed Approach
- Results
- Conclusion and Future Work

- 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

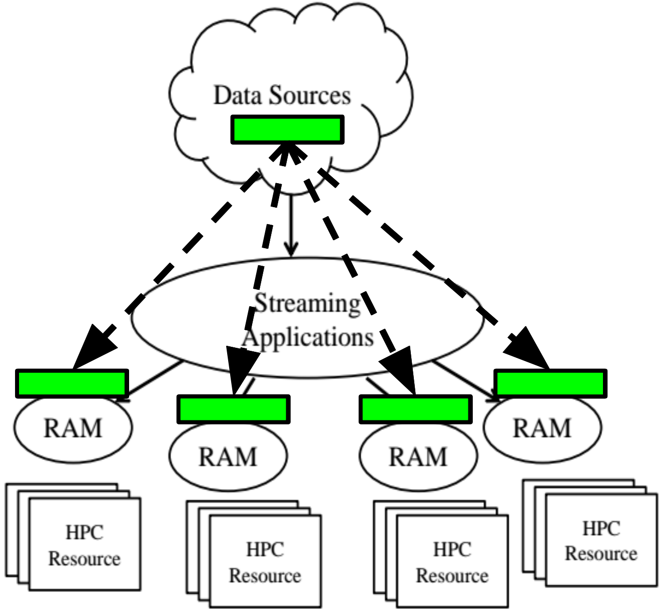
- Proliferation of Multi-Petaflop systems
- Heterogeneity in compute resources with GPGPUs
- High performance interconnects with RDMA capabilities to host and GPU memories
- Streaming applications leverage on such resources

Outline

- Introduction
- Motivation and Problem Statement
- Design Considerations
- Proposed Approach
- Results
- Conclusion and Future Work

Nature of Streaming Applications

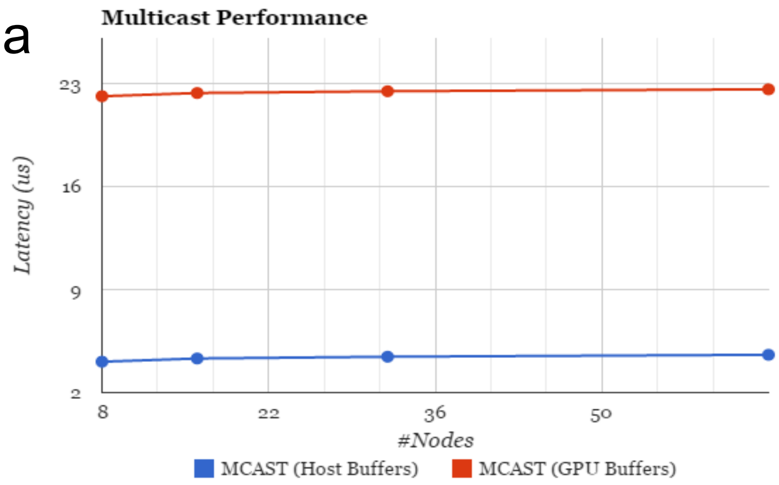
- Pipelined data parallel compute phases that form the crux of streaming applications lend themselves for GPGPUs
- Data distribution to GPGPU sites occur over PCIe within the node and over InfiniBand interconnects across nodes
- Broadcast operation is a key dictator of throughput of streaming applications
  - Reduced latency for each operation
  - Support multiple back-to-back operations
  - More critical with accelerators

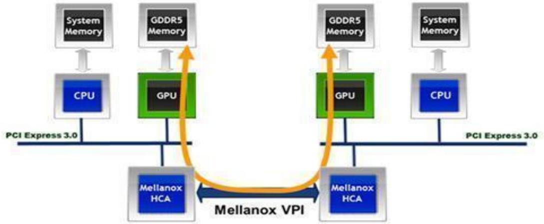


Courtesy: Agarwalla, Bikash, et al. "Streamline: A scheduling heuristic for streaming applications on the grid." *Electronic Imaging 2006*

Shortcomings of Existing GPU Broadcast

- 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
- GPU-Direct RDMA capabilities unused





Problem Statement

- Can we design a GPU broadcast mechanism that can completely avoid host-staging for streaming applications?
- Can we harness the capabilities of GPU-Direct RDMA (GDR)?
- Can we overcome limitations of UD transport and realize the true potential of multicast for GPU buffers?
- Succinctly, how do we multicast GPU data using GDR efficiently?

Outline

- Introduction
- Motivation and Problem Statement
- Design Considerations
- Proposed Approach
- Results
- Conclusion and Future Work

Outline

- Introduction
- Motivation and Problem Statement
- Design Considerations
  - Critical Factors
- Proposed Approach
- Results
- Conclusion and Future Work

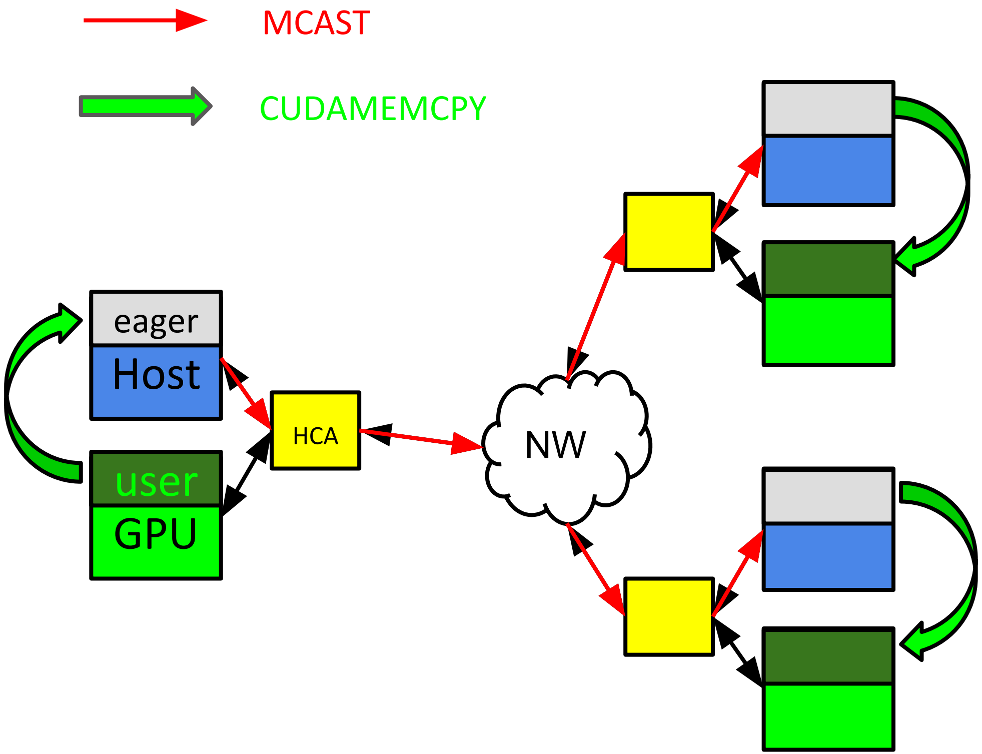
Factors to Consider for an Efficient GPU Multicast

- Goal is to be able to multicast GPU data in lesser time than the host-staged multicast (~20us)
- Cost of cudamemcpy is ~8us for short messages for host->gpu, gpu->host and gpu->gpu transfers
- Cudamemcpy costs and memory registration costs determine the viability of a multicast protocol for GPU buffers

- Introduction
- Motivation and Problem Statement
- Design Considerations
  - Eager Protocol
  - Rendezvous Protocol
- Proposed Approach
- Results
- Conclusion and Future Work

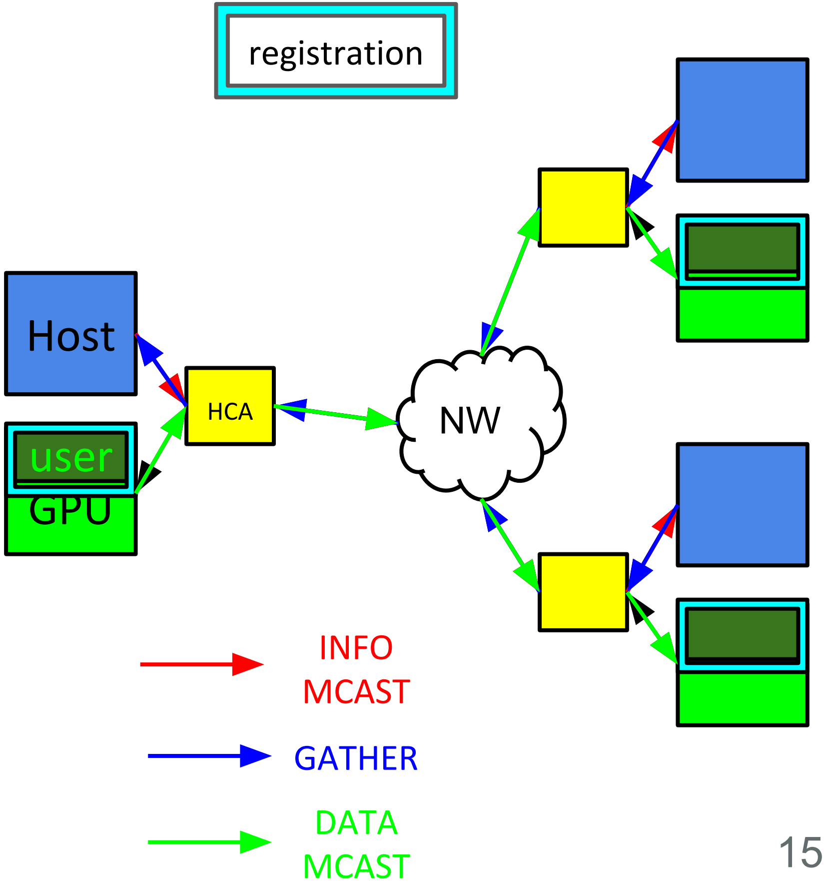
- Copy user GPU data to host eager buffers
- Perform Multicast and copy back
- Cudamemcpy dictates performance
- Similar variation with eager buffers on GPU

  -Header encoding expensive



Rendezvous Protocol for GPU multicast

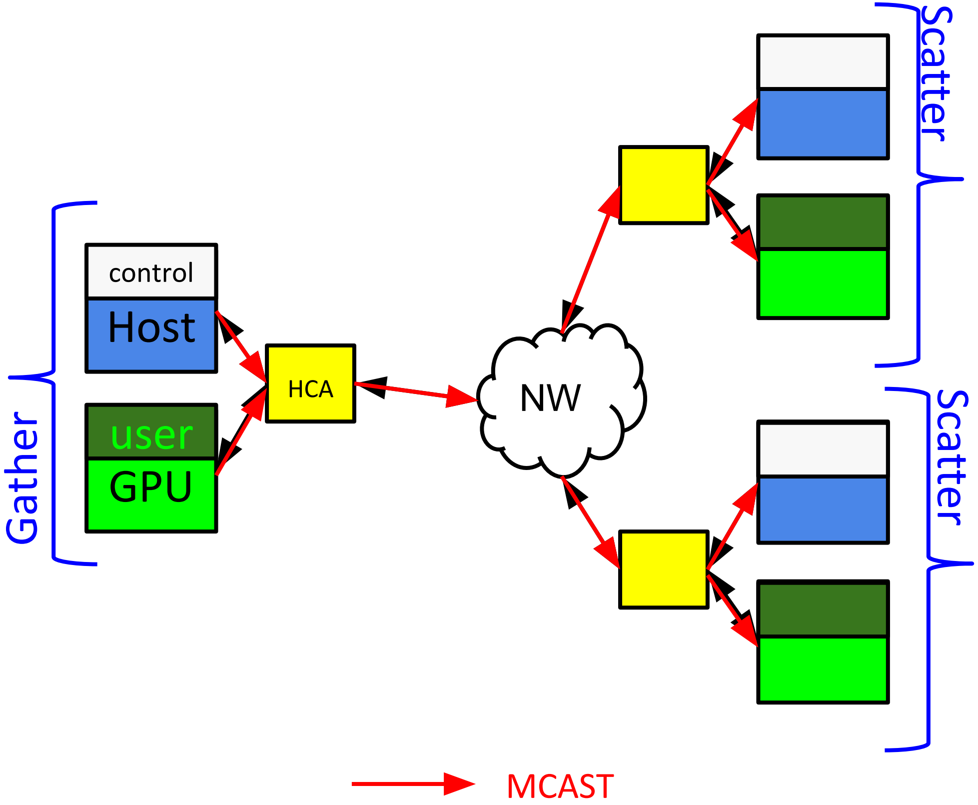
- Register user GPU data and start *RTS multicast* with control info
- Confirm ready receivers ≡ 0-byte gather
- Perform Data Multicast
- Registration cost and gather limitations
- Handshake for each operation – not required for streaming applications which are error tolerant



Outline

Orchestration of GDR-SGL-MCAST (GSM)

- One time registration of window of persistent buffers in streaming apps
- Combine 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



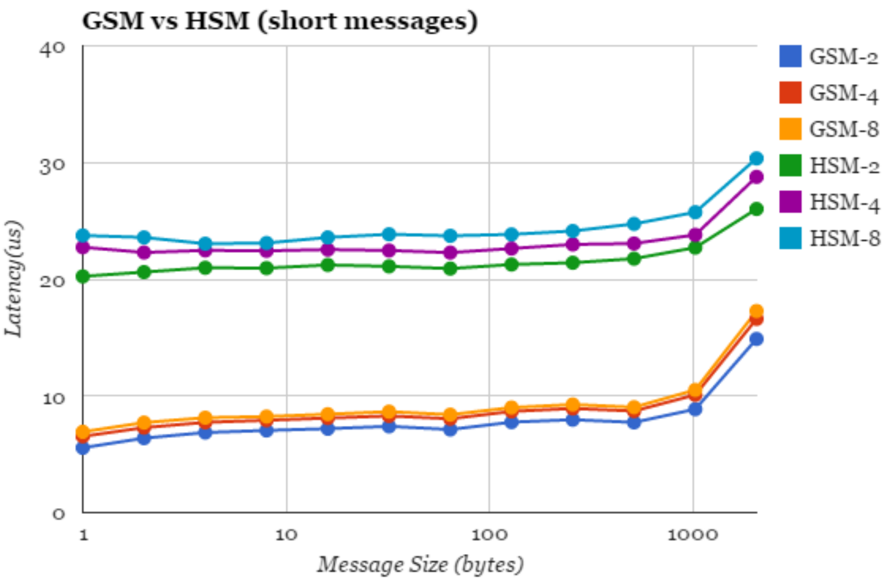
Outline

MVAPICH

- Introduction
- Motivation and Problem Statement
- Design Considerations
- Proposed Approach
- Results
- Conclusion and Future Work

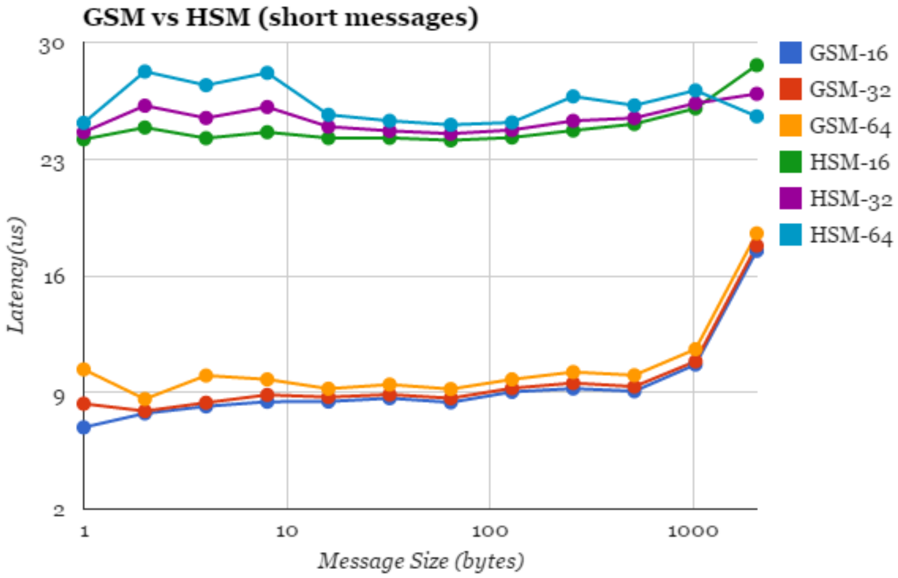
Experiment Setup

- Experiments were run on Wilkes @ University of Cambridge
- 12-core Ivy Bridge 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
- Baseline Host-based MCAST uses MVAPICH2-GDR (http://mvapich.cse.ohio-state.edu/downloads)
- GDR-SGL-MCAST is based on MVAPICH2-GDR



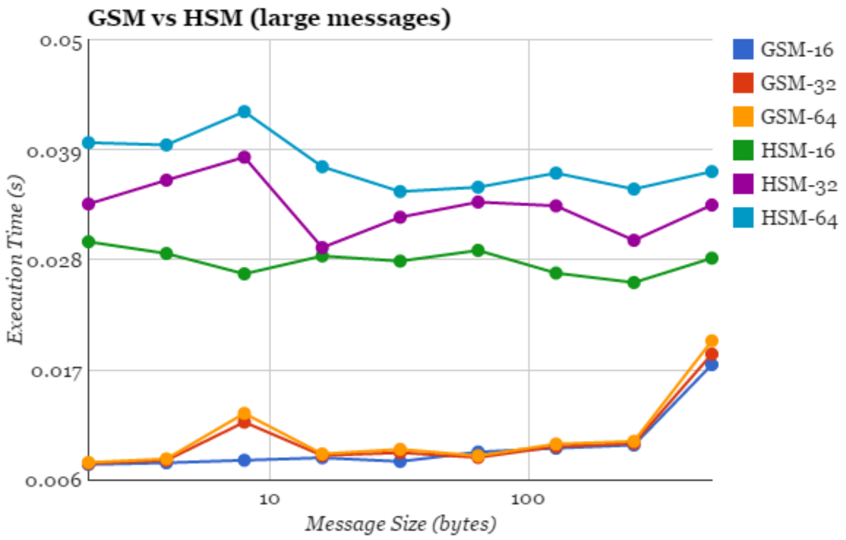
- GDR-SGL-MCAST (GSM)
- Host-Staged-MCAST (HSM)
- GSM Latency ≤ ~10us vs HSM Latency ≤ ~23us
- Small latency increase with scale

A. Venkatesh, H. Subramoni, K. Hamidouche and D. K. Panda, A High Performance Broadcast Design with Hardware Multicast and GPUDirect RDMA for Streaming Applications on InfiniBand Clusters, IEEE International Conference on High Performance Computing (HiPC '14), Dec 2014.



- Both GSM and HSM continue to show near scale invariant latency with 60% improvement (8 bytes)

- 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



Outline

- Introduction
- Motivation and Problem Statement
- Design Considerations
- Proposed Approach
- Results
- Conclusion and Future Work

Conclusion and Future work

- Designed an efficient GPU data broadcast for streaming applications which uses near-constant-latency hardware multicast feature and GPUDirect RDMA
- Proposed a new methodology which overcomes the performance challenges posed by UD transport
- Benefits shown with latency and streaming-application-communication mimicking throughput benchmark
- Exploration of NVIDIA's Fastcopy module for MPI_Bcast

Learn about recent advances and upcoming features in CUDA-aware MVAPICH2-GPU library

- S5461 - Latest Advances in MVAPICH2 MPI Library for NVIDIA GPU Clusters with InfiniBand
- Thursday, 03/19 (Today)
- Time: 17:00–17:50
- Room 212 B

Contact

panda@cse.ohio-state.edu

MVAPICH

Network Based Computing Laboratory

http://mvapich.cse.ohio-state.edu

http://nowlab.cse.ohio-state.edu

Funding Acknowledgments

Funding Support by

Office of Science
U.S. DEPARTMENT OF ENERGY
NSF
Mellanox TECHNOLOGIES
CISCO SYSTEMS
QLOGIC
CRAY THE SUPERCOMPUTER COMPANY
intel
LINUX NETWORX
NVIDIA
NetApp
Sun microsystems

Equipment Support by

Mellanox TECHNOLOGIES
intel
AMD
Sun microsystems
APPRO
advanced clustering technologies, inc.
Microway Technology you can count on
NVIDIA
QLOGIC

Backup

- UD makes no ordering and reliability guarantees
- UD requires memory registration and has an MTU of 2KB
- Notification through *polling* preferred for performance
- Multicast scheme is window-based and NACK-based
- GDR allows buffers on GPU memory to be registered
- Once registered, the IB network interface can directly access GPU memory

- IB specifies use of SG elements for non-contiguous transfer
- Control and data specified in an array of SG elements
- Avoids expensive cudaMemcpy calls
- Persistent buffers amortize registration costs and facilitate pipelining in SA