MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

High-Performance Broadcast Designs for Streaming Applications on Multi-GPU InfiniBand Clusters

GPU Technology Conference (GTC 2017)

by

Dhabaleswar K. (DK) Panda

The Ohio State University

E-mail: panda@cse.ohio-state.edu

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

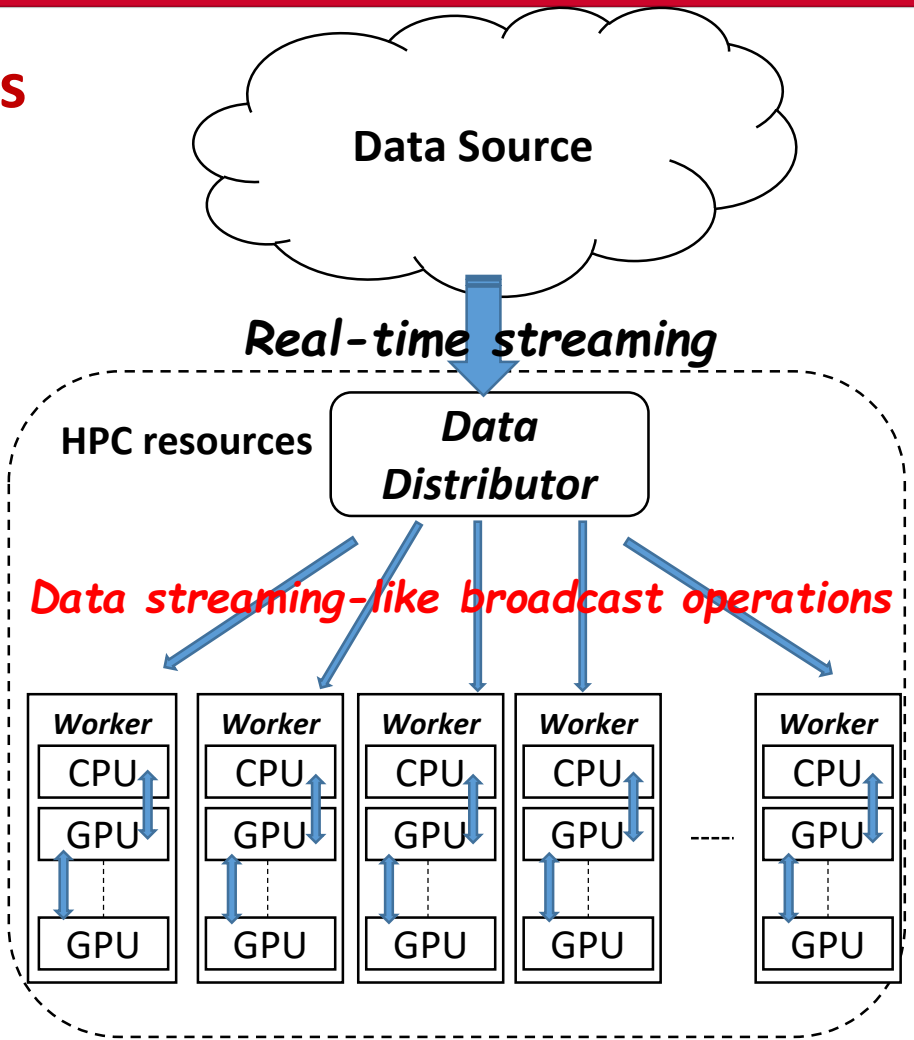
Streaming Applications

- 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
  - Form the crux of streaming applications lend themselves for GPGPUs
- Data distribution to GPGPU sites
  - Over PCIe within the node
  - Over InfiniBand interconnects across nodes
- Back-to-back Broadcast operation
  - Key dictator of throughput of streaming applications



Drivers of Modern HPC Cluster Architectures

Multi-core Processors

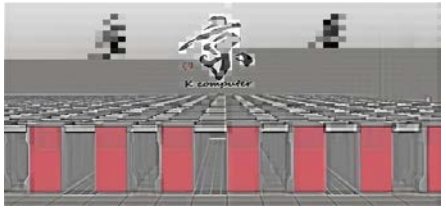
High Performance Interconnects - InfiniBand
<1usec latency, 100Gbps Bandwidth>

Accelerators / Coprocessors
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs and Intel Xeon Phi)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.

Sunway TaihuLight



K - Computer

Tianhe – 2

Titan

Large-scale InfiniBand Installations

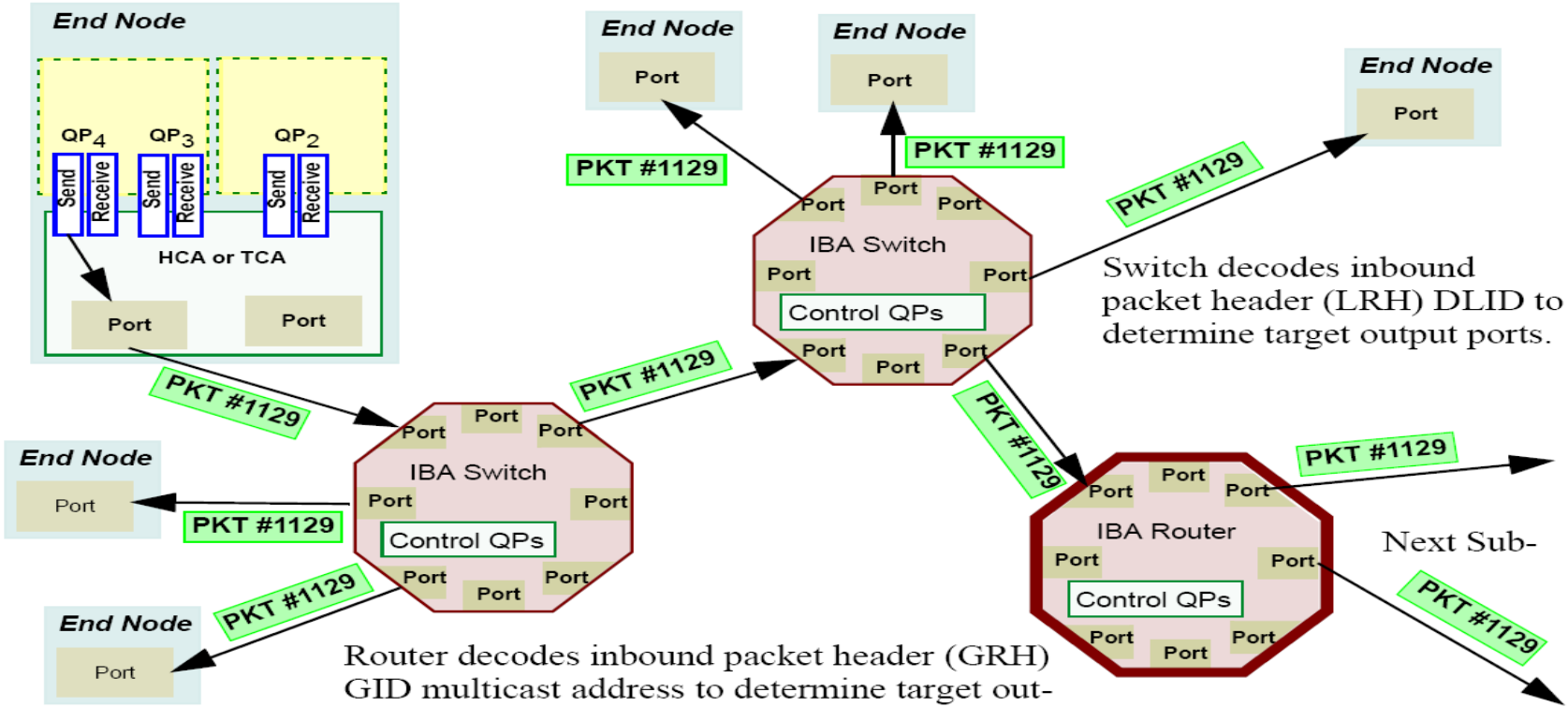
- 187 IB Clusters (37%) in the Nov'16 Top500 list
  - (http://www.top500.org)
- Installations in the Top 50 (15 systems):

241,108 cores (Pleiades) at NASA/Ames (13th)	147,456 cores (SuperMUC) in Germany (36th)
220,800 cores (Pangea) in France (16th)	86,016 cores (SuperMUC Phase 2) in Germany (37th)
462,462 cores (Stampede) at TACC (17th)	74,520 cores (Tsubame 2.5) at Japan/GSIC (40th)
144,900 cores (Cheyenne) at NCAR/USA (20th)	194,616 cores (Cascade) at PNNL (44th)
72,800 cores Cray CS-Storm in US (25th)	76,032 cores (Makman-2) at Saudi Aramco (49th)
72,800 cores Cray CS-Storm in US (26th)	72,000 cores (Prolix) at Meteo France, France (50th)
124,200 cores (Topaz) SGI ICE at ERDC DSRC in US (27th)	73,440 cores (Beaufix2) at Meteo France, France (51st)
60,512 cores (DGX SATURNV) at NVIDIA/USA (28th)	42,688 cores (Lomonosov-2) at Russia/MSU (52nd)
72,000 cores (HPC2) in Italy (29th)	60,240 cores SGI ICE X at JAEA Japan (54th)
152,692 cores (Thunder) at AFRL/USA (32nd)	**and many more!**

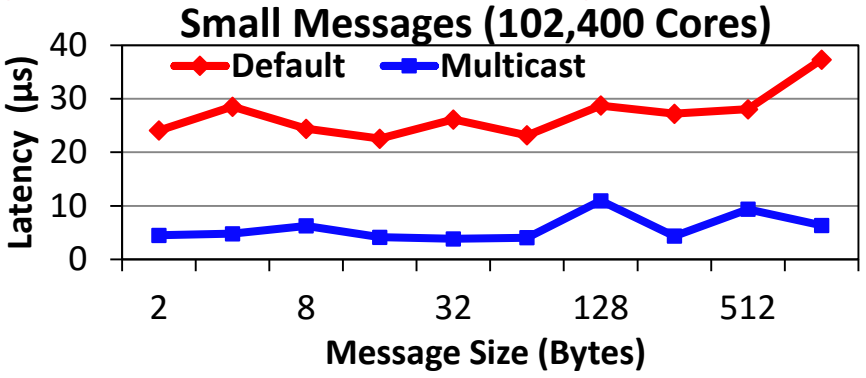
InfiniBand Networking Technology

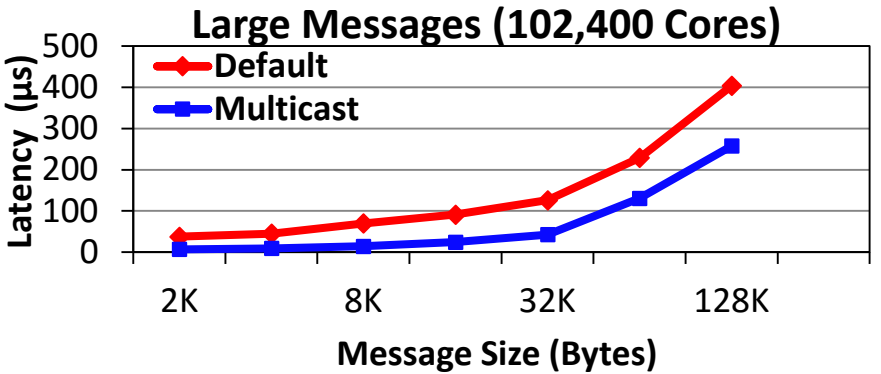
- Introduced in Oct 2000
- High Performance Point-to-point Data Transfer
  - Interprocessor communication and I/O
  - Low latency (<1.0 microsec), High bandwidth (up to 25 GigaBytes/sec -> 200Gbps), and low CPU utilization (5-10%)
- Multiple Features
  - Offloaded Send/Recv, RDMA Read/Write, Atomic Operations
  - Hardware Multicast support through Unreliable Datagram (UD)
    - A message sent from a single source can reach all destinations 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
    - Reliability needs to be addressed
- Leading to big changes in designing HPC clusters, file systems, cloud computing systems, grid computing systems, ….

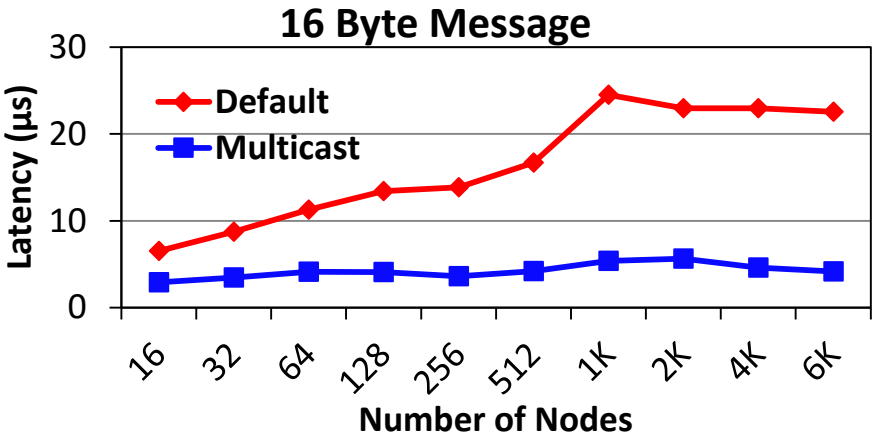
InfiniBand Hardware Multicast Example

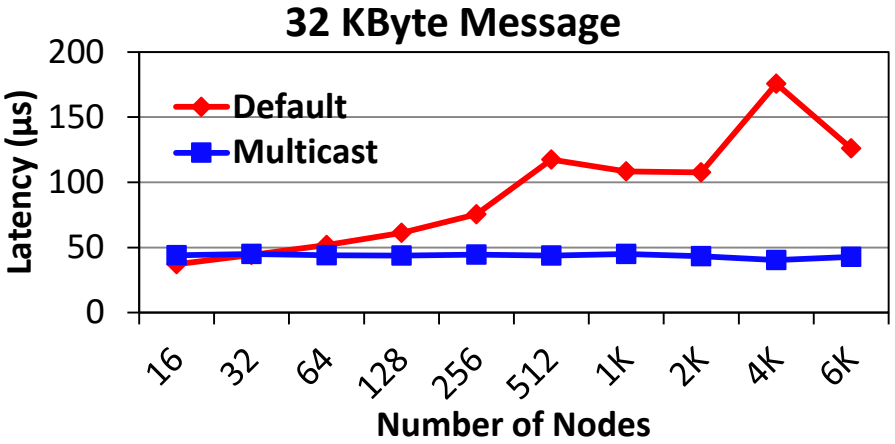


Multicast-aware CPU-Based MPI_Bcast on Stampede using MVAPICH2 (6K nodes with 102K cores)





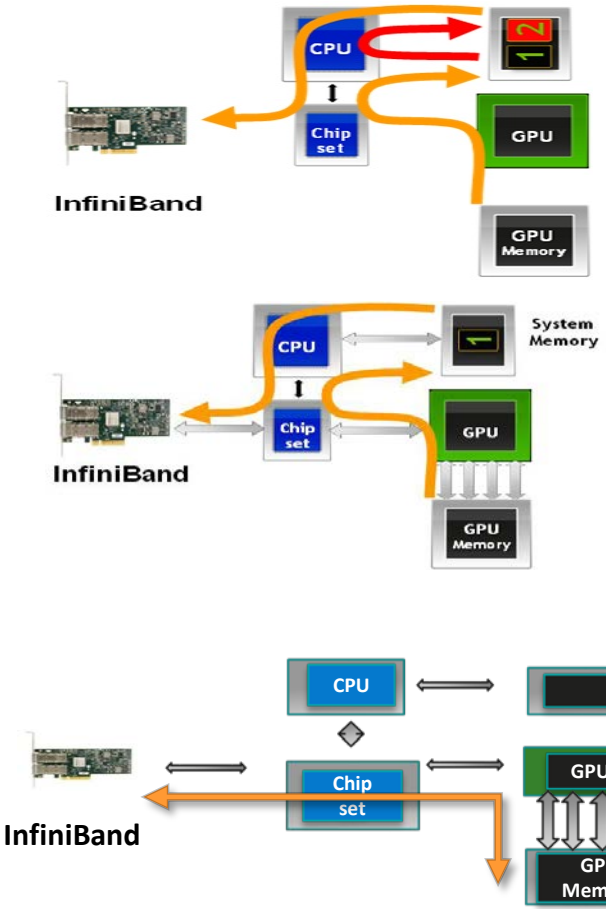




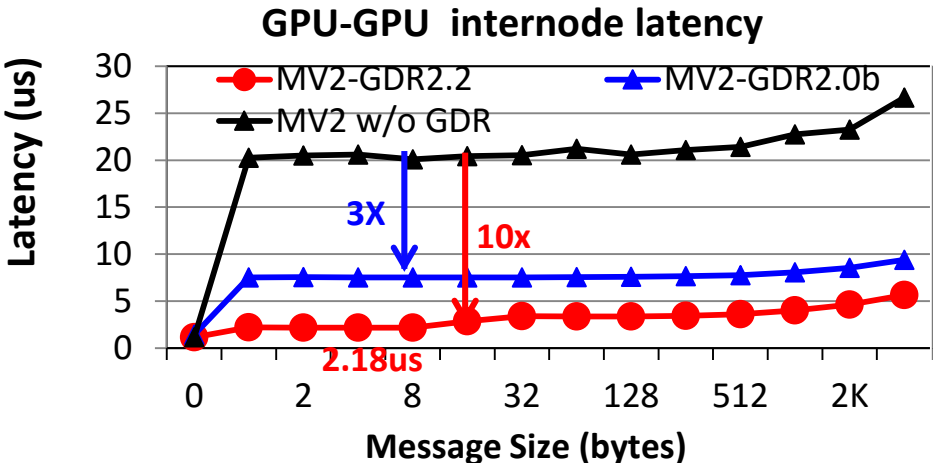
ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCIe Gen3 with Mellanox IB FDR switch

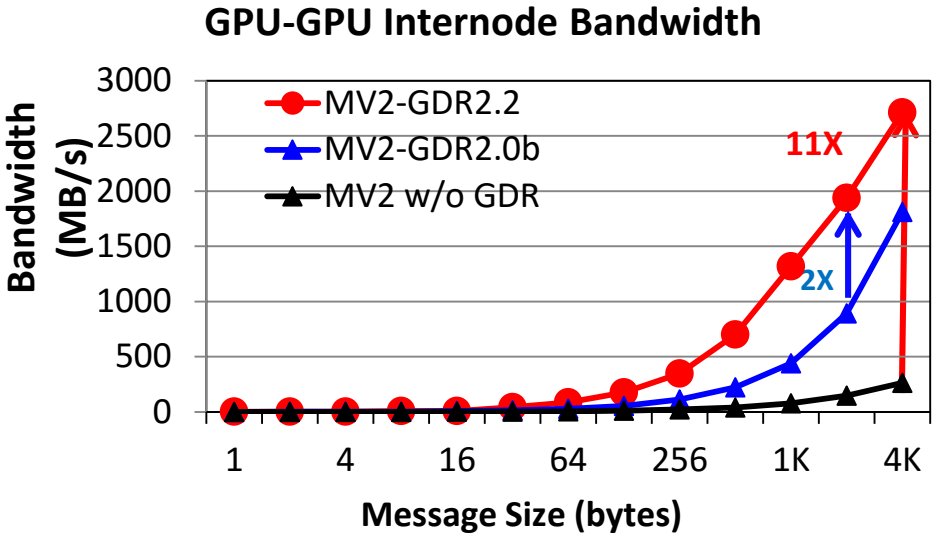
GPUDirect RDMA (GDR) and CUDA-Aware MPI

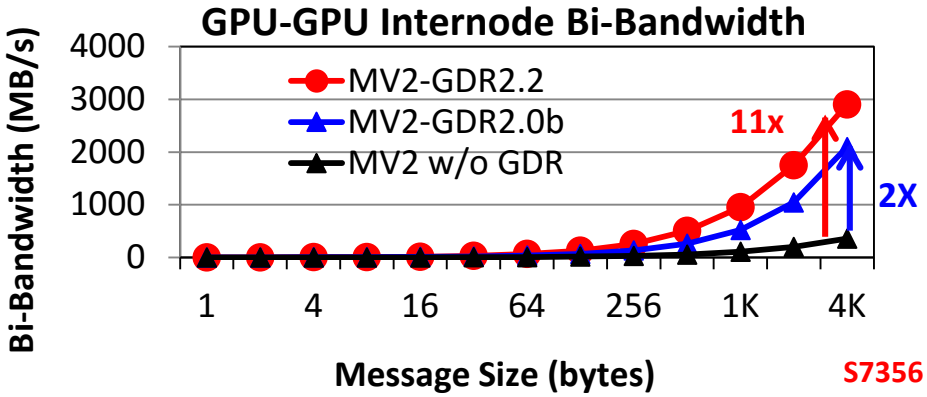
- Before CUDA 4: Additional copies
  - Low performance and low productivity
- After CUDA 4: Host-based pipeline
  - Unified Virtual Address
  - Pipeline CUDA copies with IB transfers
  - High performance and high productivity
- After CUDA 5.5: GPUDirect RDMA support
  - GPU to GPU direct transfer
  -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



Performance of MVAPICH2-GDR with GPUDirect RDMA (GDR)







MVAPICH2-GDR-2.2
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 8.0
Mellanox OFED 3.0 with GPU-Direct-RDMA

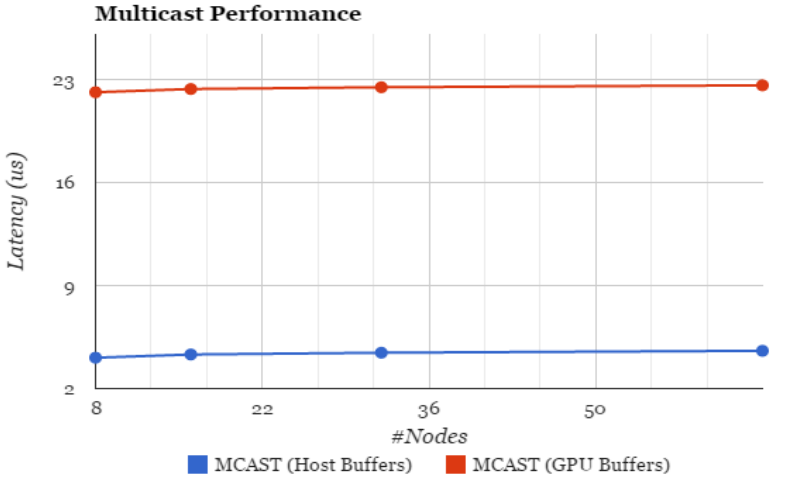
More details in 2:00pm session today

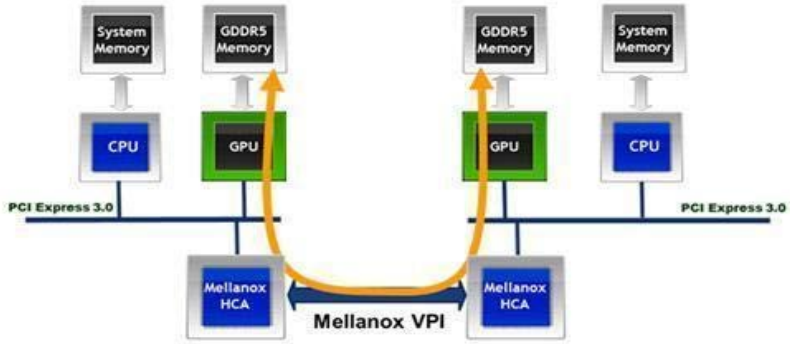
S7356 - MVAPICH2-GDR: PUSHING THE FRONTIER OF HPC AND DEEP LEARNING

Multicasting Data from one GPU to other GPUs: Shortcomings

- Host-Staged Multicast (HSM): Traditional short message broadcast operation between GPUs
  - Data copied from GPU to host memory
  - Using InfiniBand UD-based hardware multicast

- Sub-optimal use of near-scale invariant UD-multicast performance
- PCIe resources are wasted and benefits of multicast are nullified
- GPUDirect RDMA capabilities unused





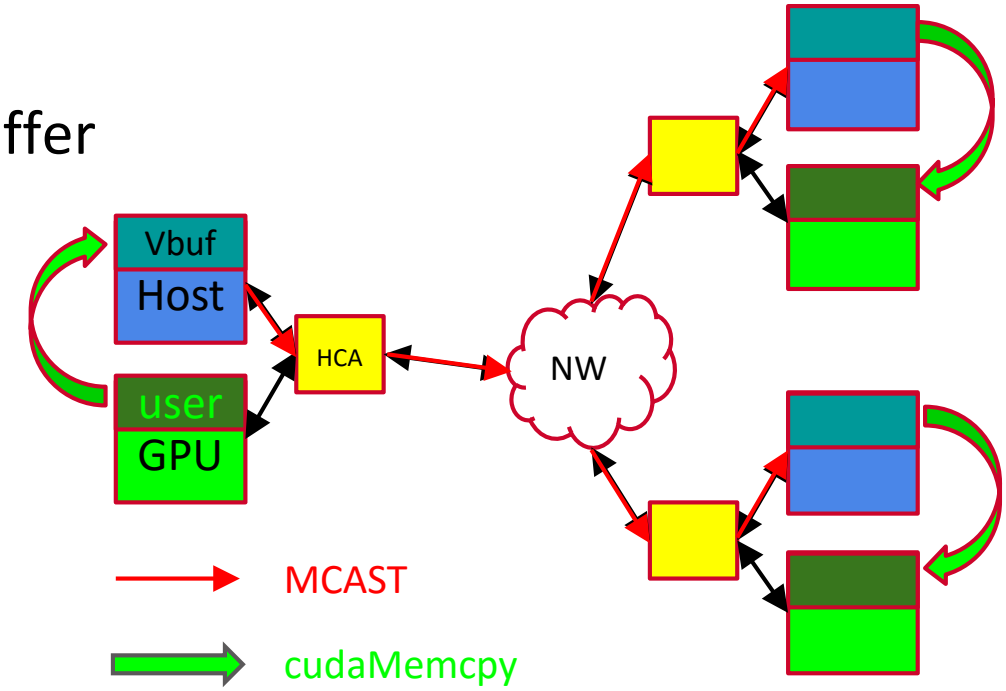
Problem Statement

- Can we design a GPU broadcast mechanism that can deliver low latency and high throughput for streaming applications?
- Can we combine GDR and MCAST features to
  - Achieve the best performance
  - Free-up the Host-Device PCIe bandwidth for application needs
- Can such design be extended to support heterogeneous configurations?
  - Host-to-Device (H2D): Most common in streaming applications
    - E.g., Camera connected to host and devices used for computation
  - Device-to-device (D2D)
  - Device-to-Host (D2H)
- Can we design an efficient MCAST based broadcast for multi-GPU systems?
- Can we design an efficient reliability support on top of the UD-based MCAST broadcast?
- 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
- Copy data back to user GPU buffer

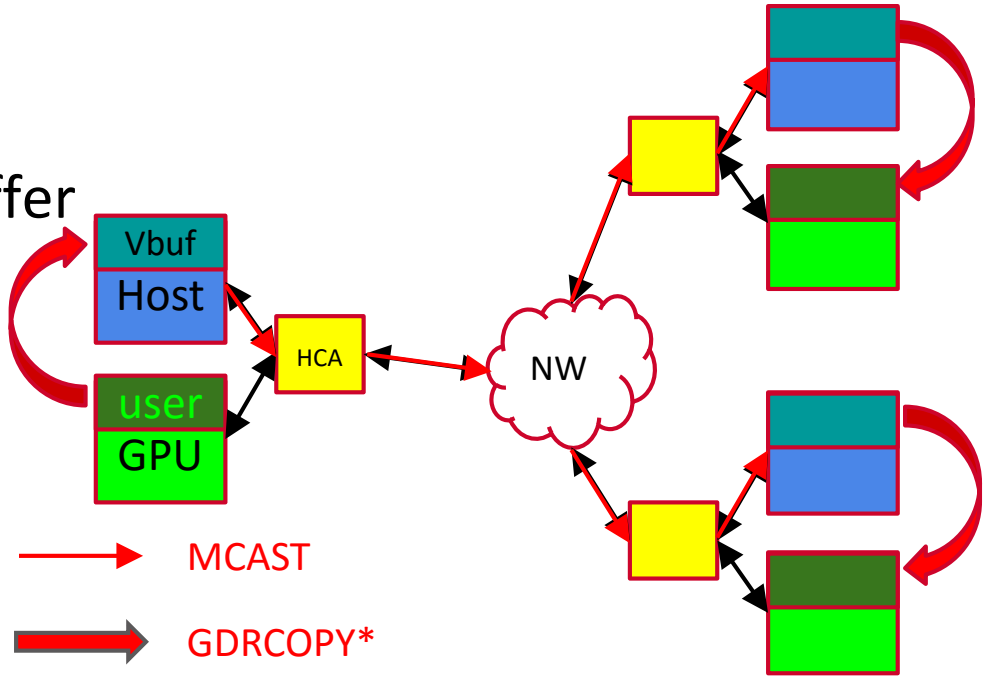
- Drawbacks:
  - CudaMemcpy dictates performance
  - Requires PCIe Host-Device resources



Enhanced Solution #1: GDRCOPY-based design

- Copy user GPU data to host buffers
  - Using GDRCOPY module*
- Perform Multicast
- Copy data back to user GPU buffer
  - Using GDRCOPY module

- Drawbacks:
  - D-H operation limits performance
  - Can we avoid GDRCOPY for D-H copies?

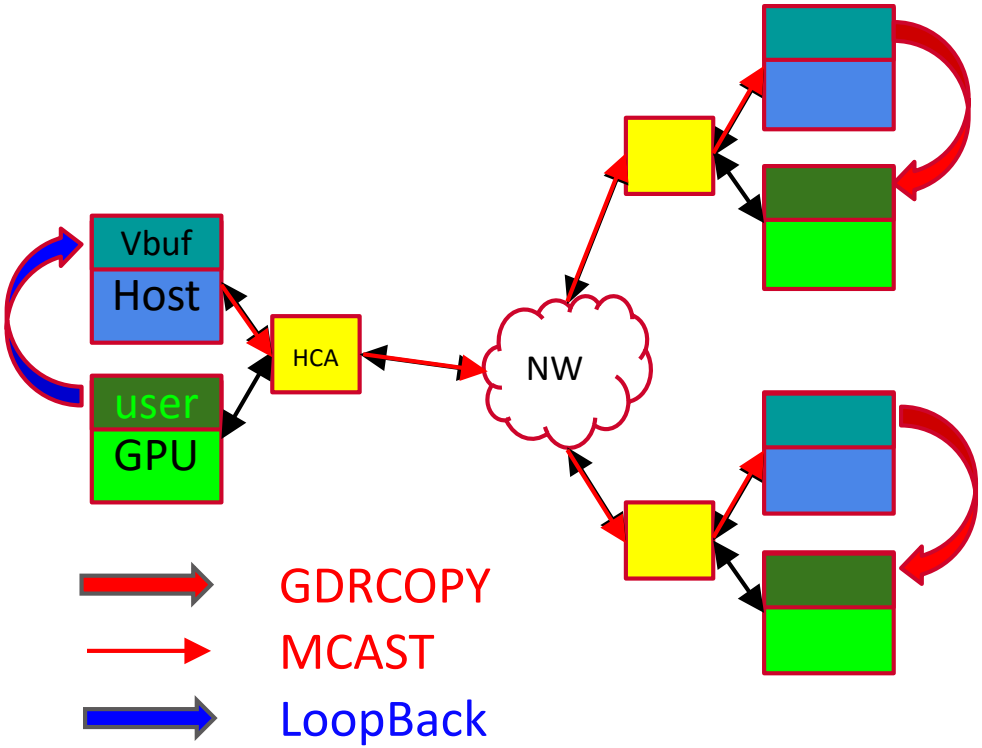


*https://github.com/NVIDIA/gdrcopy

Enhanced Solution #2: (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 good performance only for small message
- Still using the PCIe H-D resources



Can we do Better?

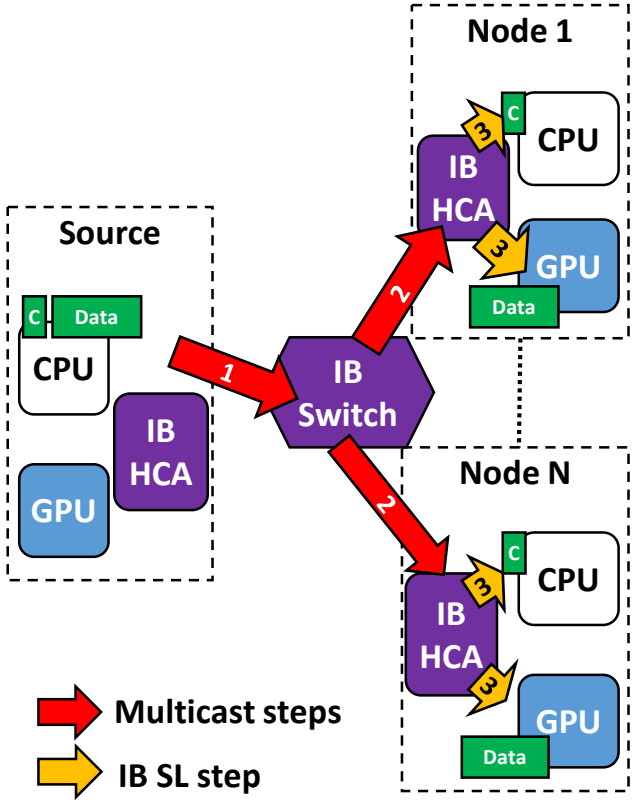
- How to design efficient and reliable broadcast operation from host to device for streaming applications on multi-GPU node systems?
- **Challenges**
  - How to handle heterogeneity of the configuration including H2D broadcast?
  - Can we have a topology-aware broadcast designs on multi-GPU nodes?
  - Can we enhance the reliability support for streaming applications?
  - Can we mimic such behavior at benchmark level?
    - mimic the need for PCIe H-D at application level
  - Demonstrate the benefits of such designs on such application patterns

Three Major Solutions

- Handling Efficient Broadcast on Multi-GPU node Systems
  - C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. “Designing High Performance Heterogeneous Broadcast for Streaming Applications on GPU Clusters, “ SBAC-PAD’16, Oct 2016.
- Providing Reliability Support
  - C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. “Efficient Reliability Support for Hardware Multicast-based Broadcast in GPU-enabled Streaming Applications,“ in COMHPC 2016 (SC Workshop), Nov 2016.
- Optimizing Broadcast for Multi-source Streaming
  - C.-H. Chu, X. Lu, A. Awan, H. Subramoni, J. Hashmi, B. Elton, and D. K. Panda, "Efficient and Scalable Multi-Source Streaming Broadcast on GPU Clusters for Deep Learning , “ Accepted for presentation, Int’l Conference on Parallel Processing, ICPP ’17, Aug 2017.

SL-based Design for Heterogeneous Configuration (H2D)

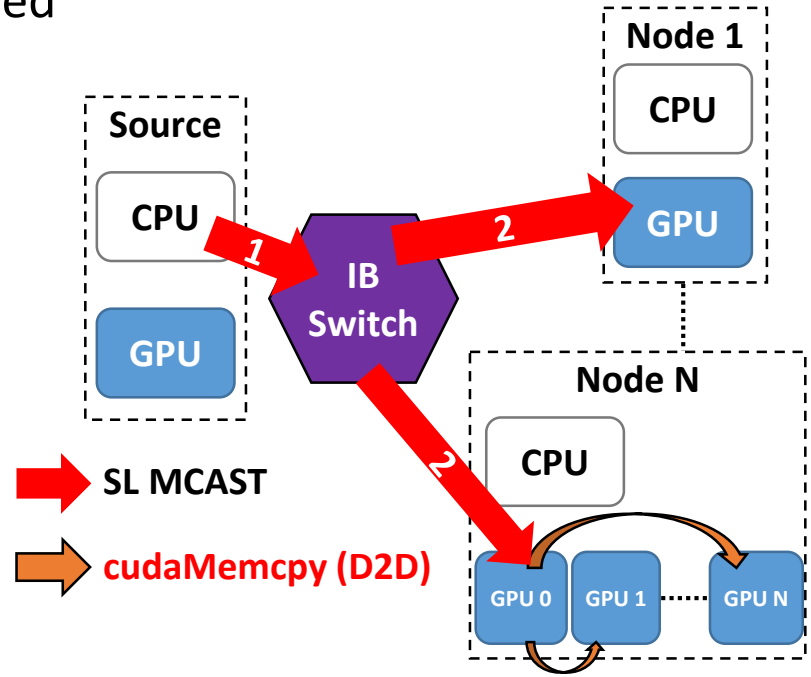
- **Combining MCAST+GDR hardware features for heterogeneous configurations:**
  - Source on the Host and destination on Device
  - SL design: Scatter at destination
    - Source: Data and Control on Host
    - Destinations: Data on Device and Control on Host
  - Combines IB MCAST and GDR features at receivers
  - CUDA IPC-based topology-aware intra-node broadcast
  - Minimize use of PCIe resources
  - Maximizing availability of PCIe Host-Device Resources



C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda.
"Designing High Performance Heterogeneous Broadcast for Streaming Applications on GPU Clusters, " SBAC-PAD'16, Oct 2016.

Intra-node Topology-aware (Hybrid SL+IPC) Design for Multi-GPU Node Configuration

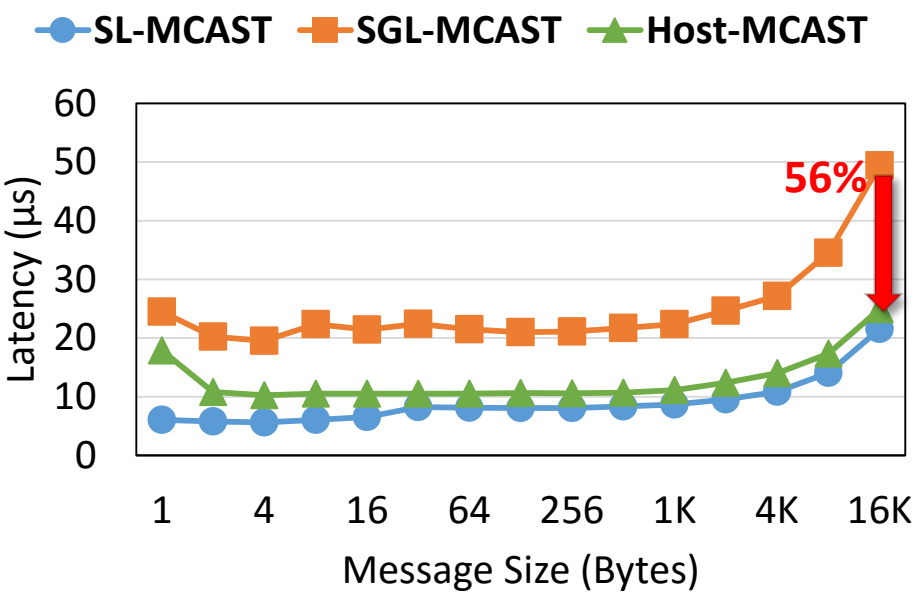
- Socket based leader (1 HCA per socket)
  - Control synchronization through Host Shared memory
    - Polling on shared flag
    - Reading the buffers addresses
  - IPC read of the GPU data
    - Direct (RMA semantics) IPC read
    - IPC read with other access patterns in the future
      - K-nomial Tree, Ring structure

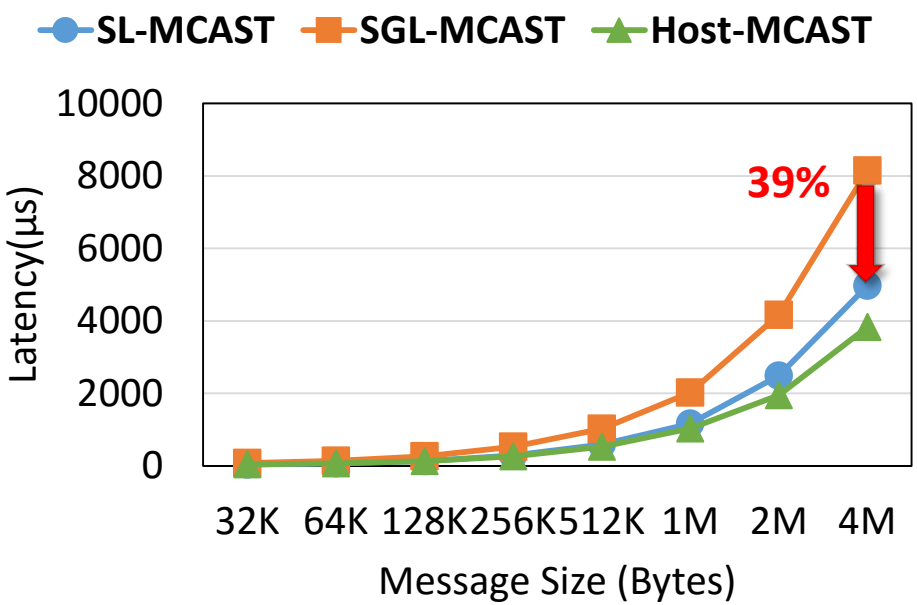


SL-based Design for H-D Heterogeneous Support

- Redesigned broadcast benchmark with Root buffer on Host & non-Root on Device
- Inter-node experiments @ Wilkes cluster, 32 GPUs, 1 GPU/node

Lower is better

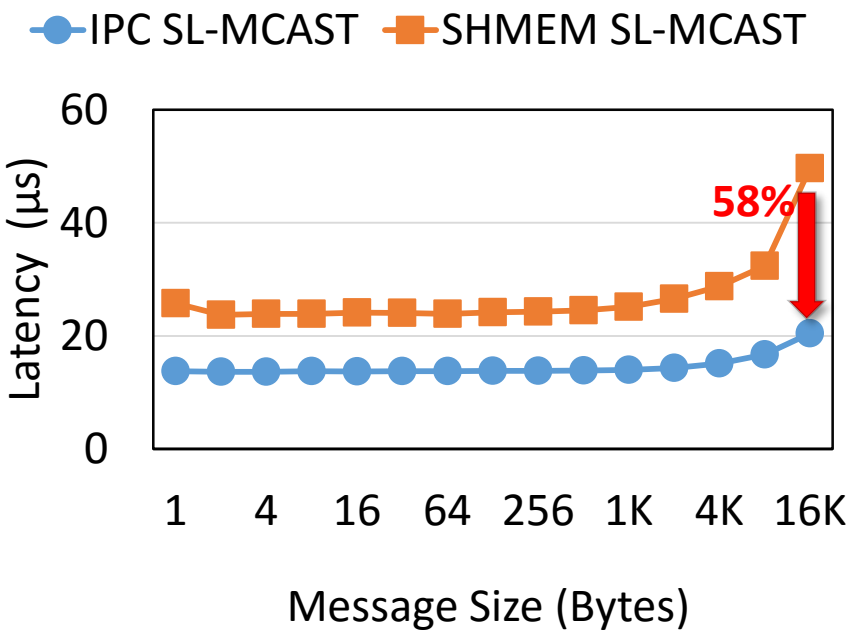


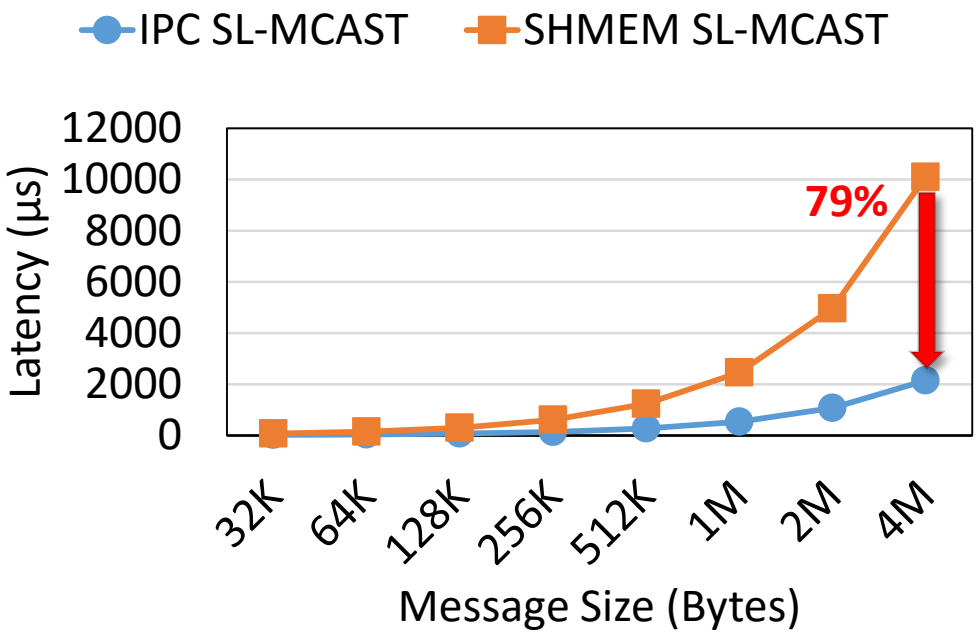


Evaluation of the Topology-aware (SL+IPC) Design

- Evaluate H-D heterogeneous support
  - Mixing Inter-node and Intra-node experiments @ CSCS cluster, 88 GPUs, 8 NVIDIA K80 GPUs per node

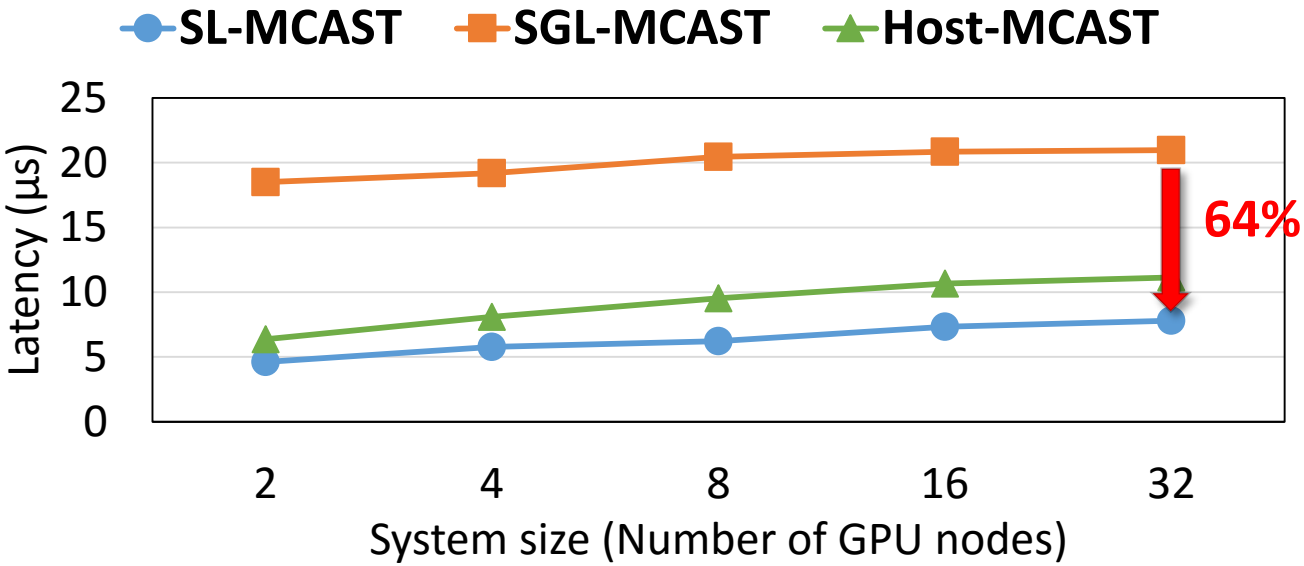
Lower is better





Scalability Evaluation of the Proposed Design

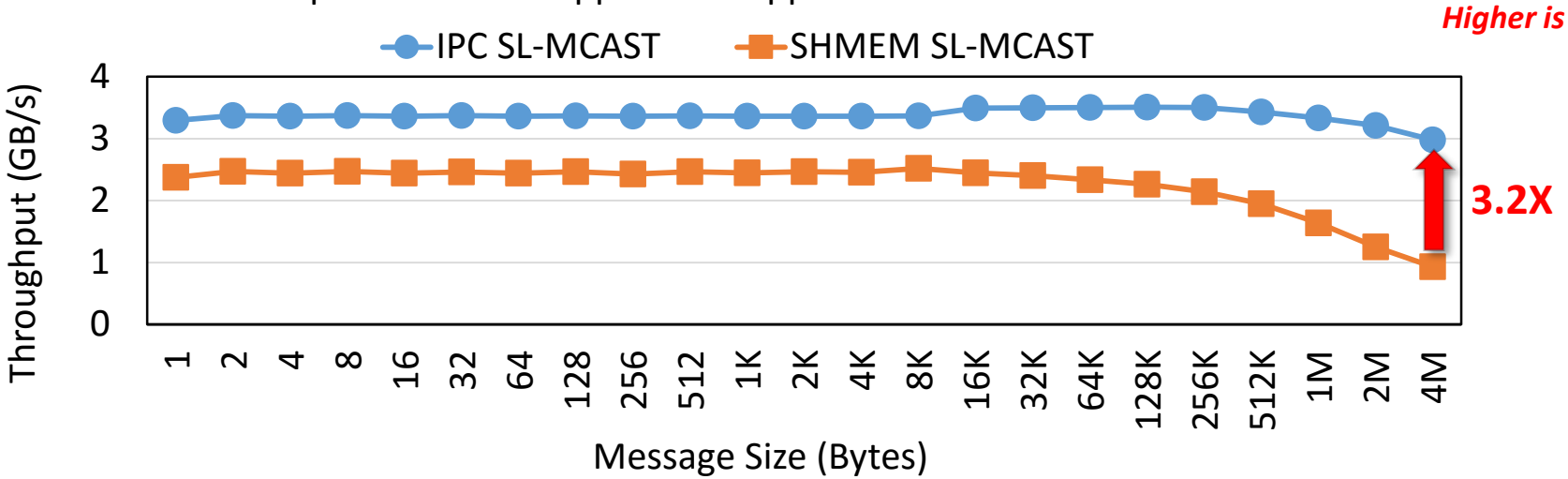
- Inter-node experiments @ Wilkes cluster, 32 GPUs, 1 GPU/node
  - 1K byte messages



Maintain good Scalability while yielding up to 64% reduction of latency

Benefits of the Availability of Host-Device PCI Resources

- Mimic the behavior of streaming applications @ CSCS cluster, 88 GPUs, 8 NVIDIA K80 GPUs per node
  - Broadcast operations overlapped with application level Host-Device transfers



Maintain near-peak throughput over all message sizes

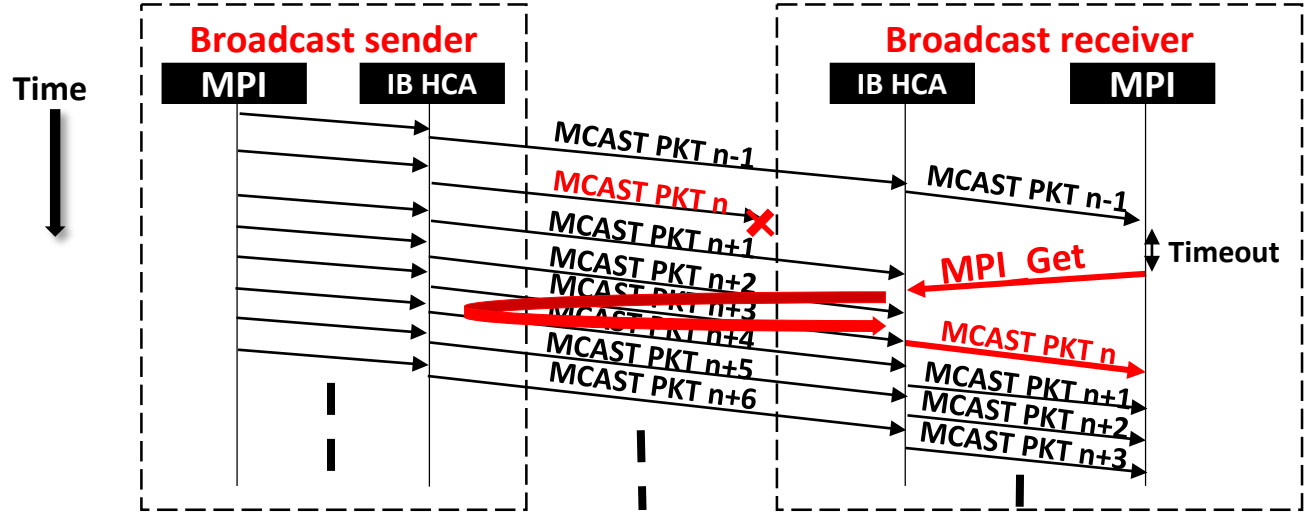
C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda.
"Designing High Performance Heterogeneous Broadcast for Streaming Applications on GPU Clusters, " SBAC-PAD'16, Oct 2016.

Three Major Solutions

- Handling Efficient Broadcast on Multi-GPU node Systems
  - C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. “Designing High Performance Heterogeneous Broadcast for Streaming Applications on GPU Clusters, “ SBAC-PAD’16, Oct 2016.
- Providing Reliability Support
  - C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. “Efficient Reliability Support for Hardware Multicast-based Broadcast in GPU-enabled Streaming Applications,“ in COMHPC 2016 (SC Workshop), Nov 2016.
- Optimizing Broadcast for Multi-source Streaming
  - C.-H. Chu, X. Lu, A. Awan, H. Subramoni, J. Hashmi, B. Elton, and D. K. Panda, "Efficient and Scalable Multi-Source Streaming Broadcast on GPU Clusters for Deep Learning , “ Accepted for presentation, Int’l Conference on Parallel Processing, ICPP ’17, Aug 2017.

Efficient Reliability Support for MCAST-based broadcast

- **Remote Memory Access (RMA)-based Design**
  - Sender maintains a backup buffer for the MCAST packets
    - **Sender is not interrupted**
  - Receiver Performs the RMA Get operation to the sender's backup buffer to retrieve lost MCAST packets

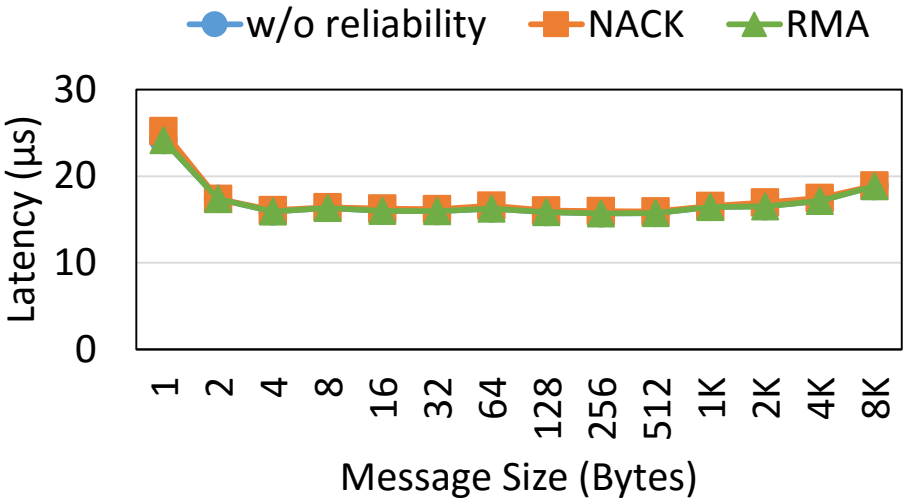


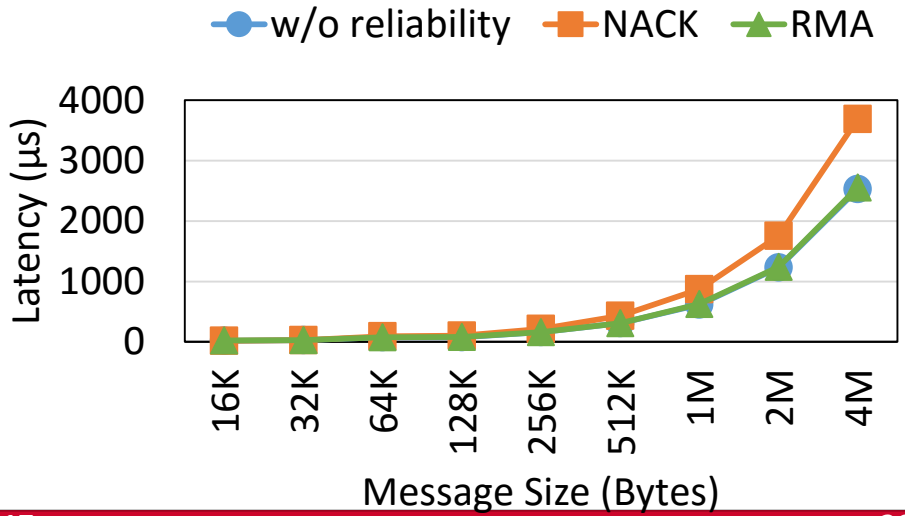
Evaluation: Efficient Reliability Design

- Evaluate the RMA-based reliability support for SL-based MCAST design

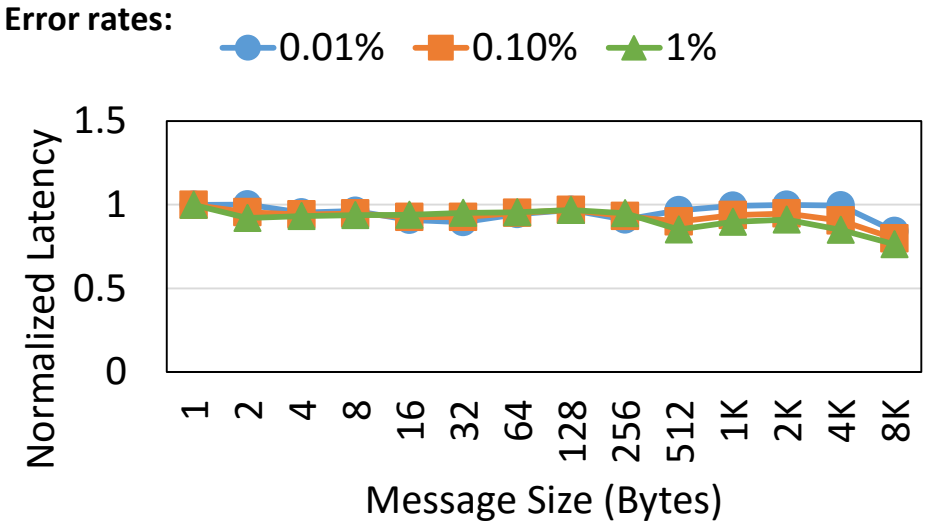
  @ CSCS cluster, 88 GPUs, 8 NVIDIA K80 GPUs per node

  - **Negligible overhead**
  - **RMA-based design performs better than NACK-based scheme for large messages**

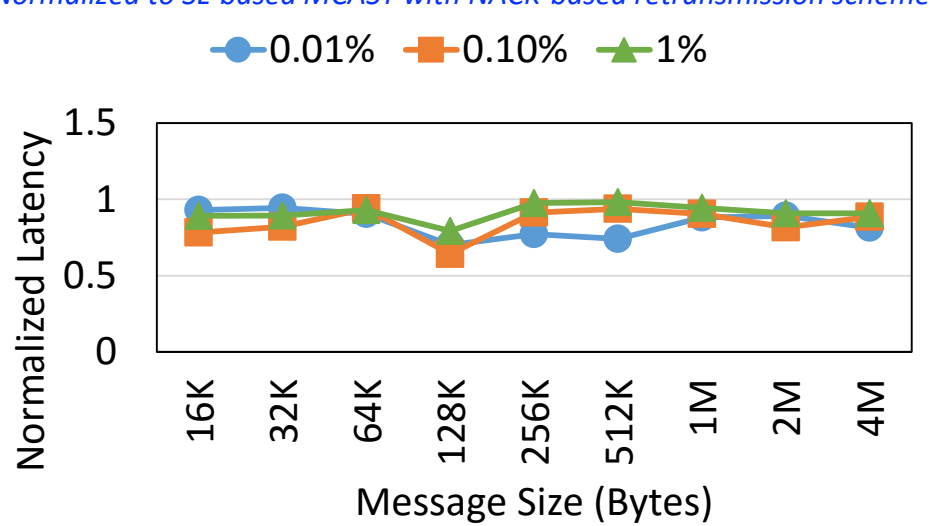




Benefits of the RMA-based Reliability Design



Normalized to SL-based MCAST with NACK-based retransmission scheme



Latency reduction compared to the existing NACK-based scheme

		Message Size		
		8KB	128KB	2MB
Error Rate	0.01%	16%	31%	11%
	0.1%	21%	36%	19%
	1%	24%	21%	10%

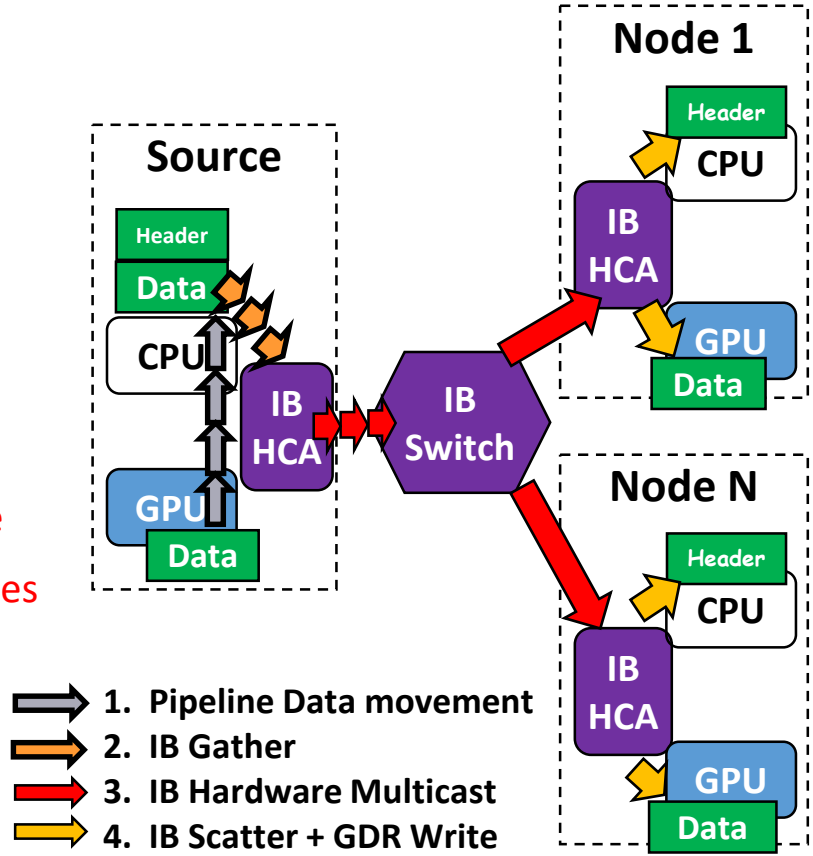
C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. "Efficient Reliability Support for Hardware Multicast-based Broadcast in GPU-enabled Streaming Applications," in COMHPC 2016 (SC Workshop), Nov 2016.

Three Major Solutions

- Handling Efficient Broadcast on Multi-GPU node Systems
  - C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. “Designing High Performance Heterogeneous Broadcast for Streaming Applications on GPU Clusters, “ SBAC-PAD’16, Oct 2016.
- Providing Reliability Support
  - C.-H. Chu, K. Hamidouche, H. Subramoni, A. Venkatesh, B. Elton, and D. K. Panda. “Efficient Reliability Support for Hardware Multicast-based Broadcast in GPU-enabled Streaming Applications,“ in COMHPC 2016 (SC Workshop), Nov 2016.
- Optimizing Broadcast for Multi-source Streaming
  - C.-H. Chu, X. Lu, A. Awan, H. Subramoni, J. Hashmi, B. Elton, and D. K. Panda, "Efficient and Scalable Multi-Source Streaming Broadcast on GPU Clusters for Deep Learning , “ Accepted for presentation, Int’l Conference on Parallel Processing, ICPP ’17, Aug 2017.

Optimized Design for Multi-Source Streaming

- **Optimizing MCAST+GDR Broadcast:**
  - Source and destination buffers are on GPU Device
    - Typically very large messages (>1MB)
  - Pipelining data from Device to Host
    - Avoid GDR read limit
    - Leverage high-performance SL design
  - Combines IB MCAST and GDR features
  - Minimize use of PCIe resources on the receiver side
  - Maximizing availability of PCIe Host-Device Resources



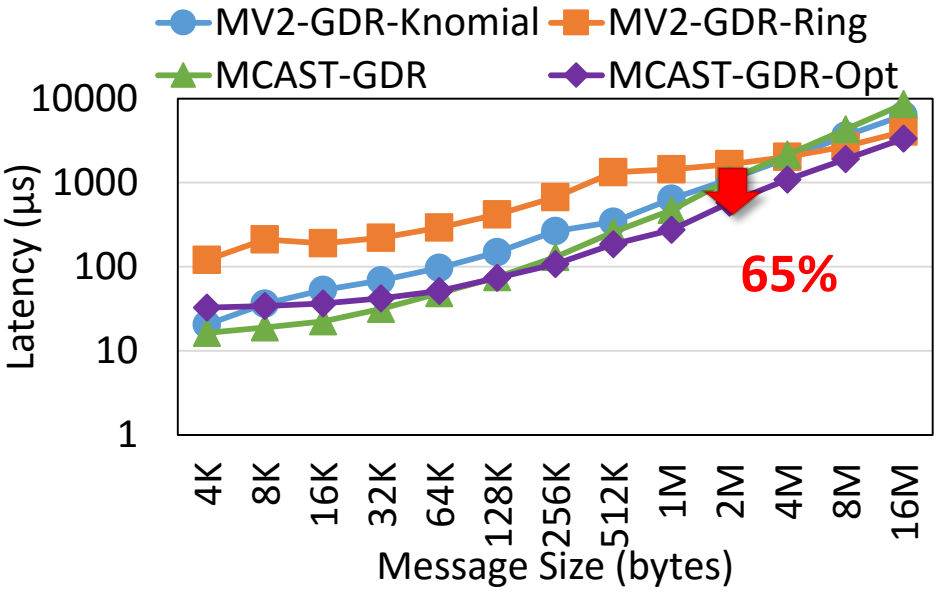
Analysis of the Optimized Design

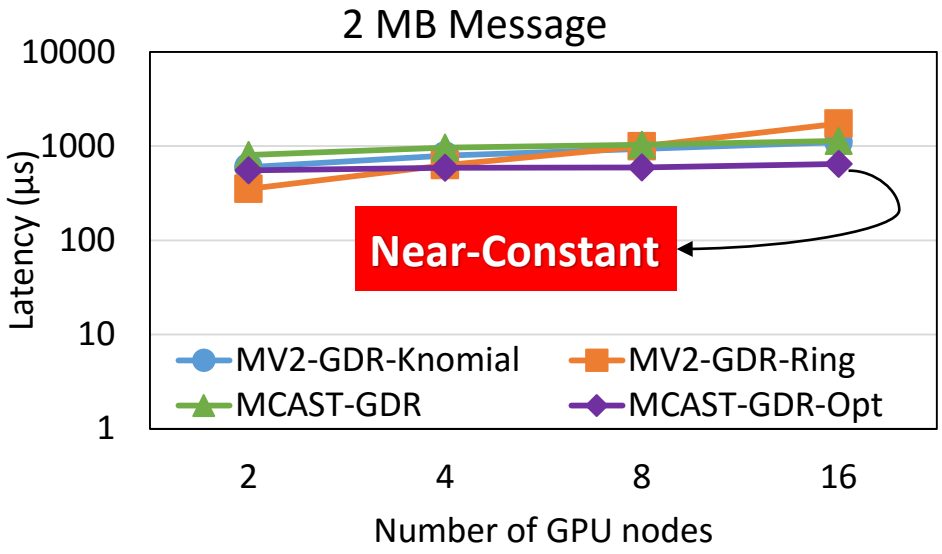
- **Pipelined MCAST+GDR Design**
  - Pipelines data from Device to Host on the source node
    - Streaming broadcast
  - Leverages high-performance SL-based design
- High Scalability
- High overlap between multiple broadcast calls

Benchmark Evaluation

- @ OSU RI2 cluster, 16 NVIDIA K80 GPUs, 1 GPU/node

Lower is better



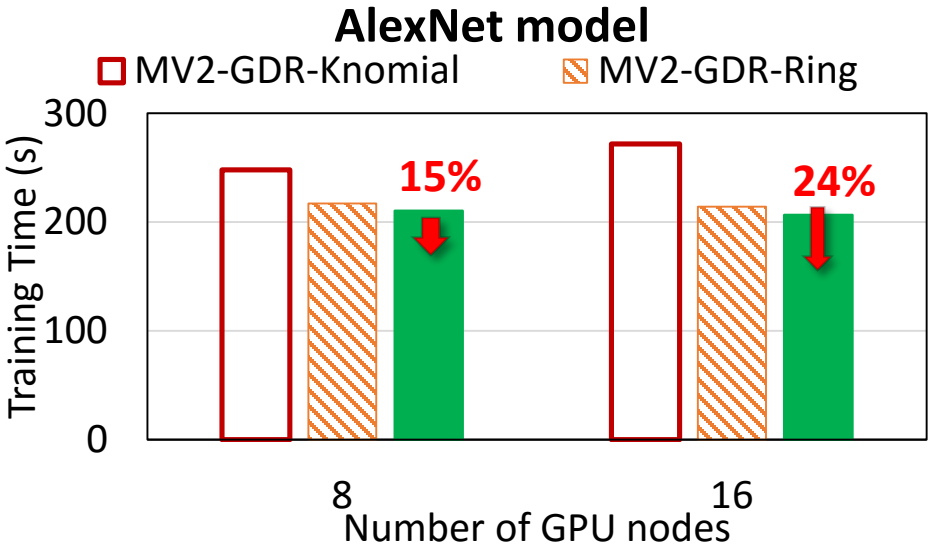


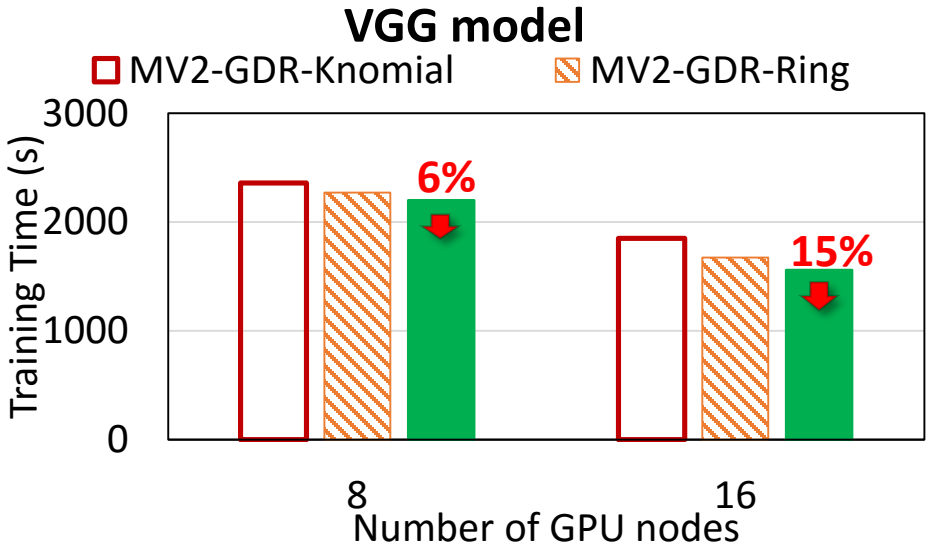
- Provide near-constant latency over the system sizes
- Reduces up to 65% of latency for large messages

Application Evaluation: Deep Learning Frameworks

- **@ OSU RI2 cluster, 16 NVIDIA K80 GPUs, 1 GPU/node**
  - Microsoft Cognitive Toolkit (CNTK) with CUDA-Aware MPI*

Lower is better





- **Reduces up to 24% and 15% of latency for AlexNet and VGG models**
  - **Average training time of one Epoch**
- **Higher improvement can be observed for larger system sizes**

*D. Banerjee, K. Hamidouche, D. Panda, Re-designing CNTK Deep Learning Framework on Modern GPU Enabled Clusters, IEEE CloudCom'16, Dec 2016.

Conclusions

- IB MCAST feature provides high scalability and low latency
- NVIDIA GDR feature provides a direct access between IB and GPUs
- MVAPICH2-GDR provides schemes to efficiently broadcast from/to GPU memories using host staged techniques
- Presented a set of designs to couple GDR and IB MCAST features for
  - Heterogeneous Systems
  - Multi-GPU systems
  - Single-source and Multi-source Streaming
- New designs will be available in future MVAPICH2-GDR library

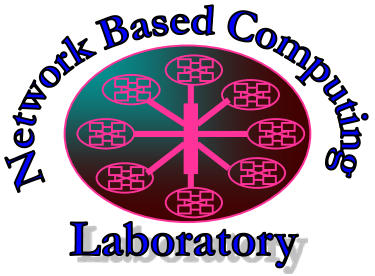
Two Additional Talks

- **S7356 - MVAPICH2-GDR: Pushing the Frontier of HPC and Deep Learning**
  - **Day:** Today, 05/11
  - **Time:** 14:00 - 14:50
  - **Location:** Room 211B

- **S7324 - Bringing NVIDIA GPUs to the PGAS/OpenSHMEM World: Challenges and Solutions**
  - **Day:** Today, 05/11
  - **Time:** 15:00 - 15:25
  - **Location:** Room 211B

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/