Designing Efficient Small Message Transfer Mechanism for Inter-node MPI Communication on InfiniBand GPU Clusters

Rong Shi* Sreeram Potluri* Khaled Hamidouche*
Jonathan Perkins* Mingzhe Li*

Davide Rossetti+ Dhabaleswar K. Panda*

*Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University

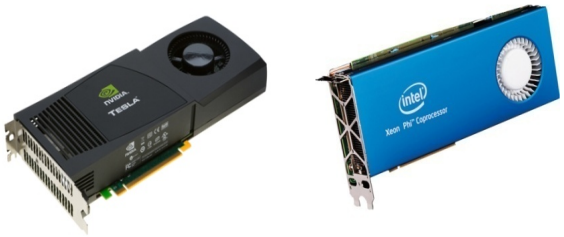
+NVIDIA Corporation

Outline

- Introduction
- Motivation
- Proposed Design for GPU-to-GPU Eager Protocol
- Performance Evaluation
- Conclusion and Future Work

Drivers of Heterogeneous HPC Cluster

Multi-core Processors



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

- Multi-core processors are ubiquitous
- High Performance Linpack (HPL) is used to measure the peak performance
- Accelerators/Coprocessors are becoming common in high-end systems
- Pushing the envelope for heterogeneous computing

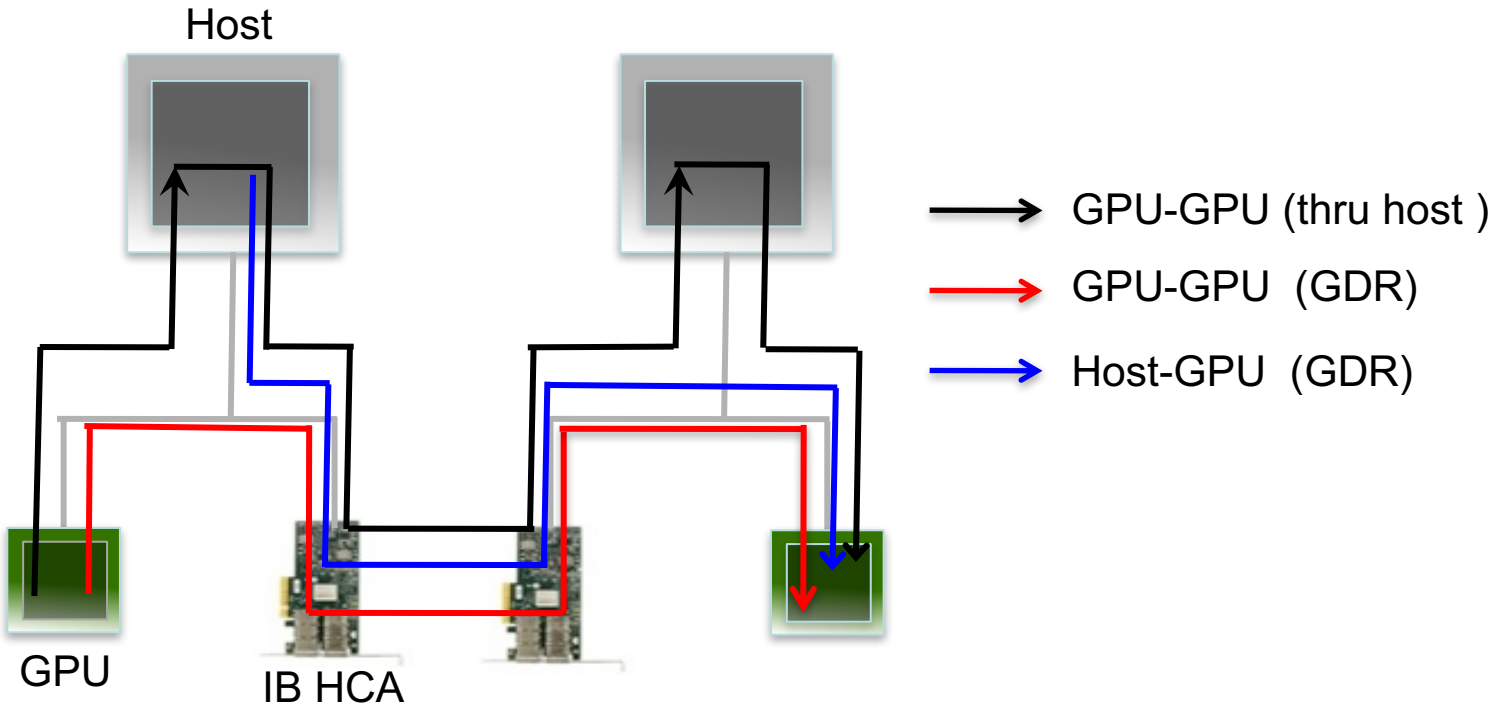
Tianhe – 1A (10)

Stampede (6)

Oakley (OSC)

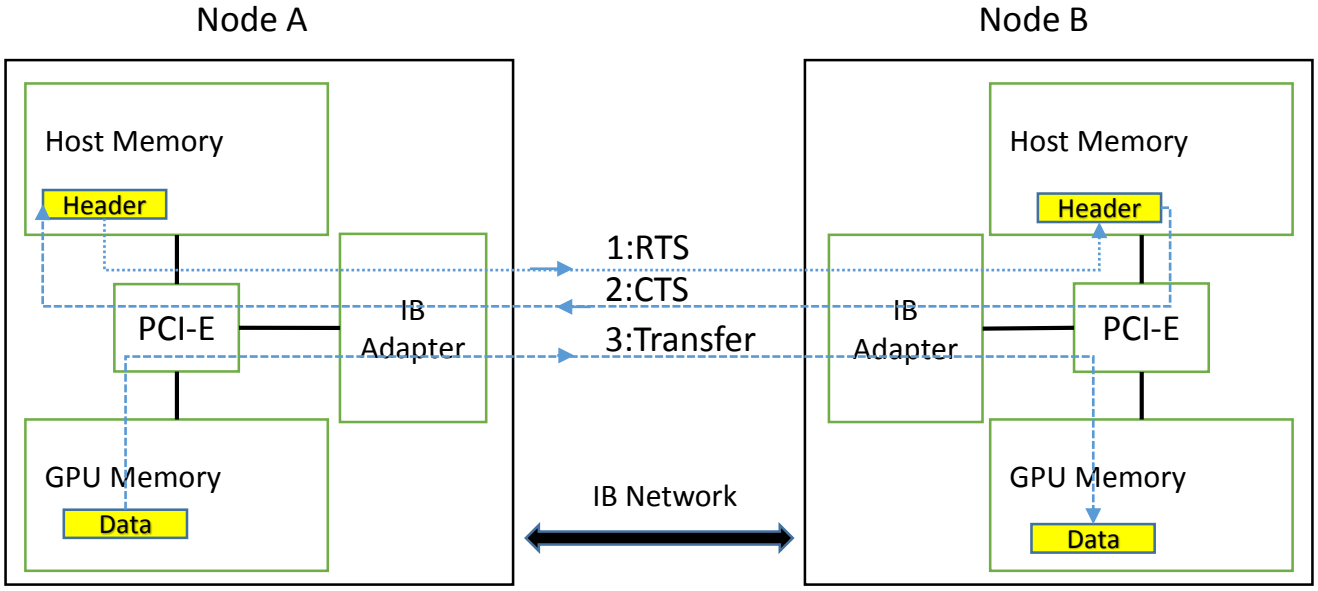
Blue Waters (NCSA)

GPU Direct RDMA (GDR)



- Eliminate CPU latency and bandwidth bottleneck
- Use RDMA transfers between GPUs
- Bring improved MPI_Send/Recv between GPUs in remote nodes

Existing Rendezvous Protocol



- Handshake with target process (RTS/CTS)
- Source process write data to target process using RDMA transfer
- Overhead: initial synchronization phase for small messages

S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, and D. K. Panda, Efficient Inter-node MPI Communication using GPUDirect RDMA for InfiniBand Clusters with NVIDIA GPUs, ICPP'13, Pittsburgh, PA, USA

Outline

- Introduction
- Motivation
- Proposed Design for GPU-to-GPU Eager Protocol
- Performance Evaluation
- Conclusion and Future Work

Motivation

- Current limitation

  Existing eager protocol utilize host-based copies using temporal buffers

  Current GPU-to-GPU communication relies on rendezvous protocol for all message sizes (header exchange & explicit synchronization overheads)

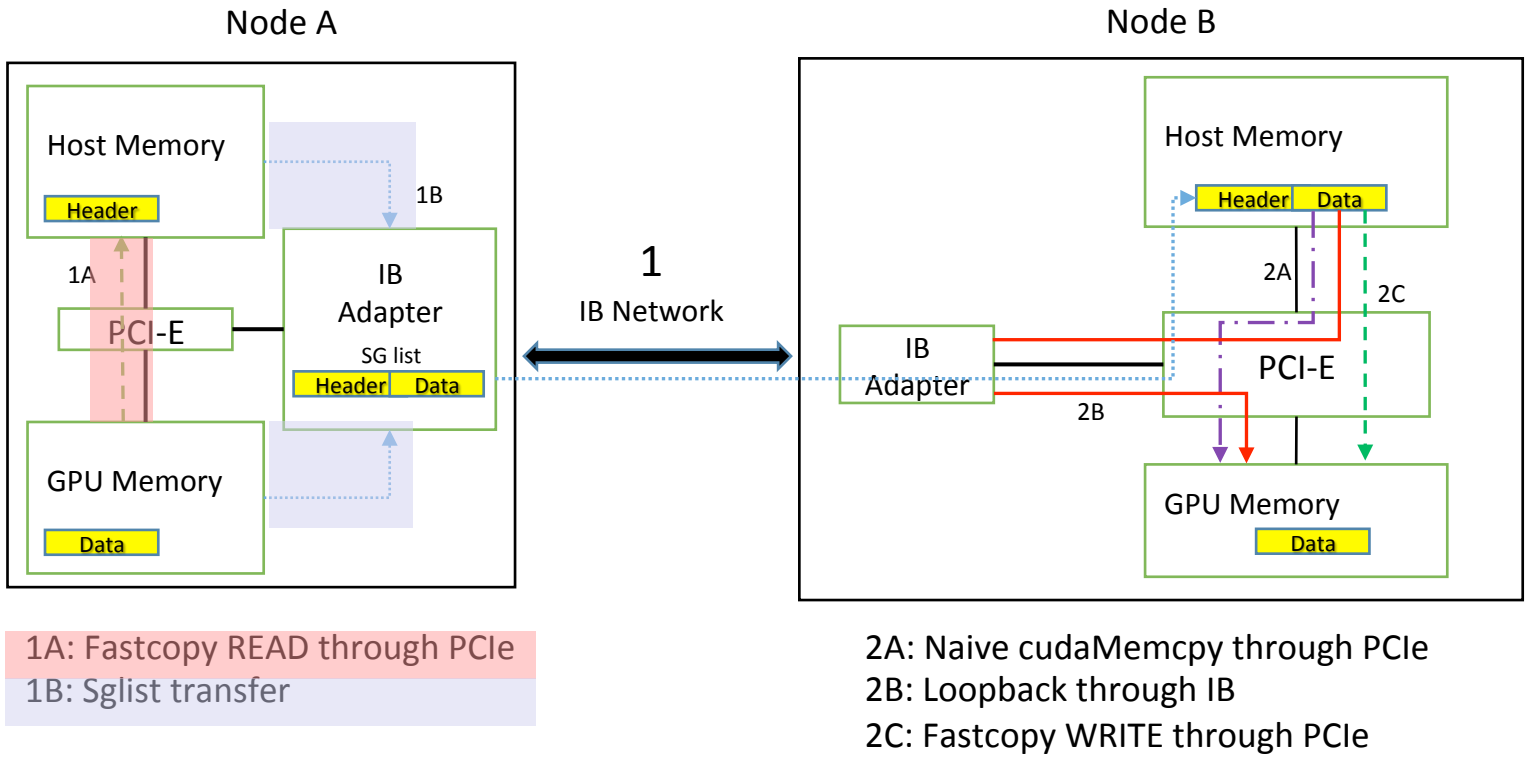
- Goal

  Design and implement an efficient eager protocol for small messages GPU-to-GPU transfer by taking advantage of GDR and InfiniBand Verbs

Outline

- Introduction
- Motivation
- Proposed Design for GPU-to-GPU Eager Protocol
- Performance Evaluation
- Conclusion and Future Work

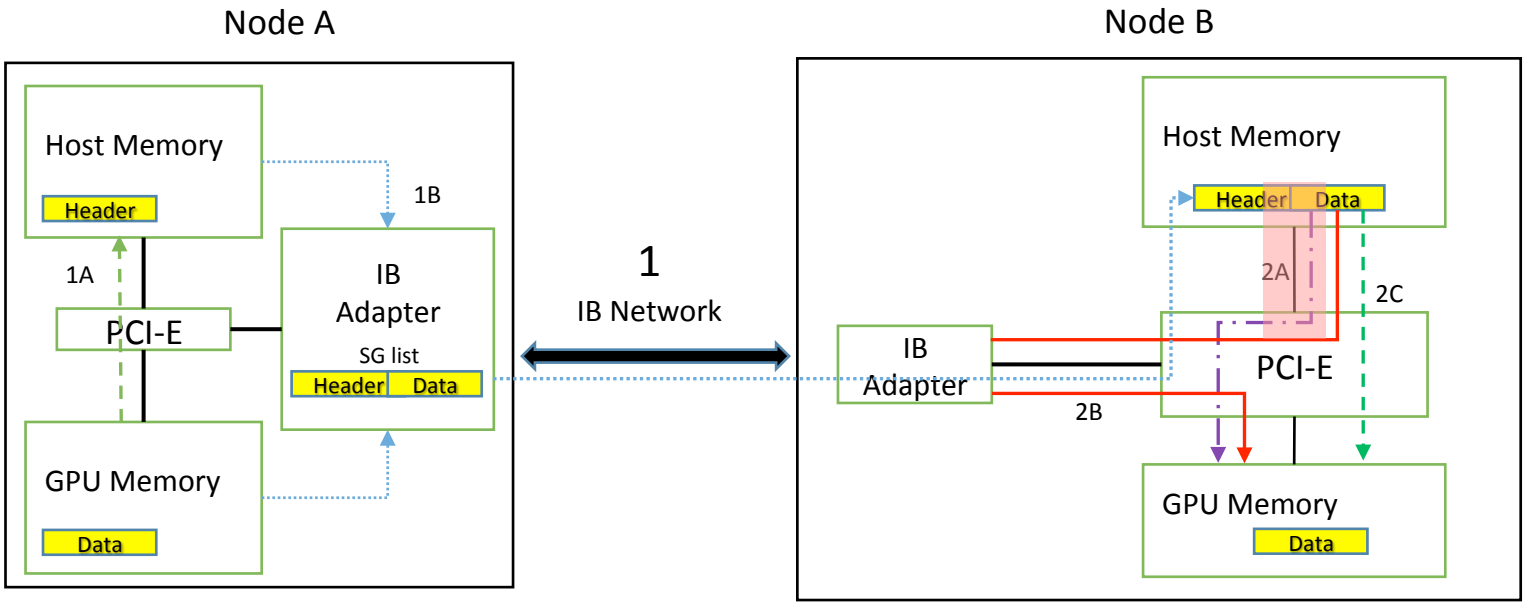
GDR Eager Sender Side Design



1A: Fastcopy Device-to-Host

2B: Scatter-Gather list scheme leverage the gather/scatter feature by one IB memory operation

GDR Eager Receiver Side Naïve Design

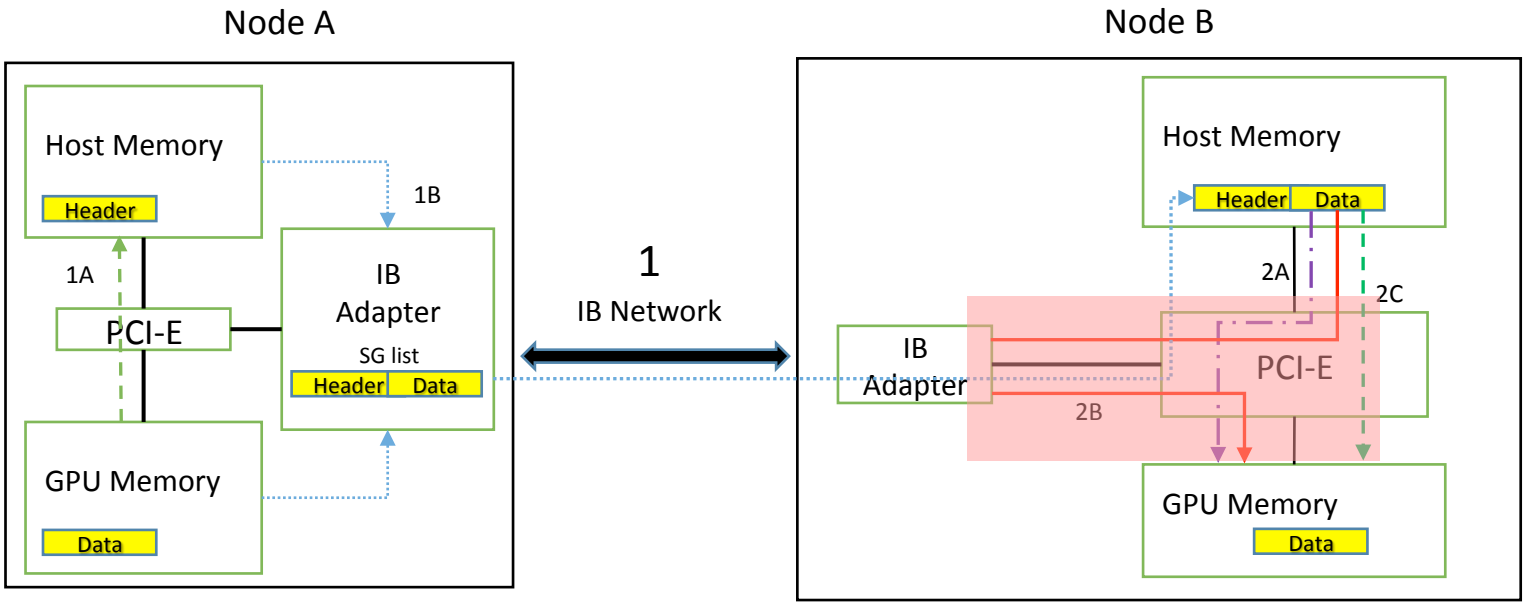


1A: Fastcopy READ through PCIe
1B: Sglist transfer

2A: Naive cudaMemcpy through PCIe
2B: Loopback through IB
2C: Fastcopy WRITE through PCIe

2A: utilize cudaMemcpy to move data between host and device buffer
Full PCIe bandwidth but incurs overhead caused by GPU DMA controller

GDR Eager Receiver Side Loopback Design

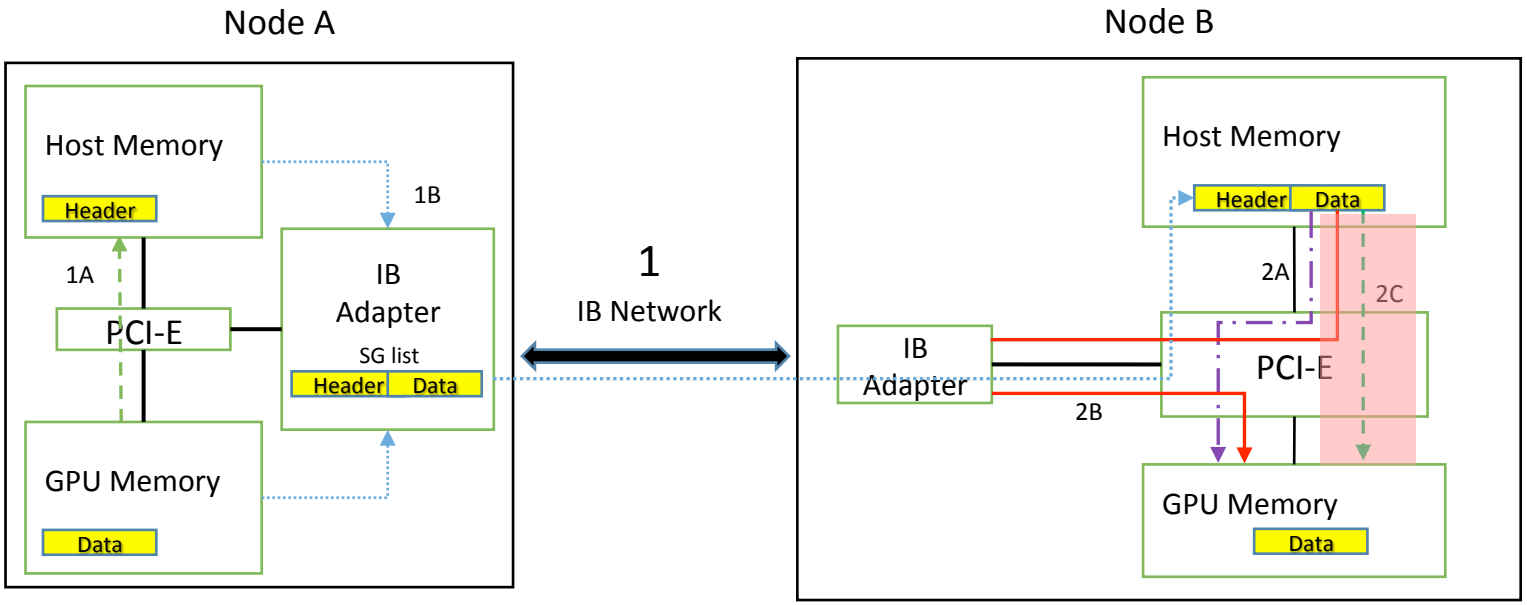


1A: Fastcopy READ through PCIe
1B: Sglist transfer

2A: Naive cudaMemcpy through PCIe
2B: Loopback through IB
2C: Fastcopy WRITE through PCIe

2B: utilize GDR feature after the header tag matching, the IB adapter reads the data from the temporary receive buffer on host to the destination memory on device
Full PCIe bandwidth but affected by PCIe peer-to-peer HW limitations

GDR Eager Receiver Side Fastcopy Design



1A: Fastcopy READ through PCIe
1B: Sglist transfer

2A: Naive cudaMemcpy through PCIe
2B: Loopback through IB
2C: Fastcopy WRITE through PCIe

2C: a new low-latency host- device copy data-path experimentally by NVIDIA
High Host-to-device (WRITE) bandwidth (~4GB/s)
low Device-to-Host (READ) bandwidth (~20MB/s)
Dynamic threshold to control the two transfer directions

Fastcopy Procedure

1. Pin/Unpin
Kernel-mode GDR APIs to setup HW mappings of GPU memory buffers

2. Map/Unmap
Pages are memory-mapped into a contiguous user-space CPU address

3. Copy to/from PCIe BAR
Highly tuned copy functions in terms of SSE instructions

Extended registration cache: keep track of the GPU buffers

Fastcopy only burns CPU cycles because of its CPU driven nature
still limited by PCIe bandwidth and affected by NUMA effect

Outline

- Introduction
- Motivation
- Proposed Design for GPU-to-GPU Eager Protocol
- Performance Evaluation
- Conclusion and Future Work

Experimental Setup

- Experiment Environment

Cluster Specification	A	B	Wilkes
CPU Processor	Intel E5-2670	Intel E5-2690 v2	Intel E5-2630
CPU Clock	2.60 GHz	3.00 GHz	2.60 GHz
Socket Type	Dual-Socket	Dual-Socket	Dual-Socket
Cores per Socket	8-Core	10-Core	6-Core
CPU Memory	32 GB	128 GB	64 GB
NVIDA GPUs	Tesla K40c	Tesla K40m	Tesla K20
GPUs per Node	2	1	2
GPU Memory	12 GB	12 GB	5 GB
Compilers	gcc 4.4.7	gcc 4.4.7	gcc 4.4.7
MPI Library	MVAPICH2-GDR	MVAPICH2-GDR	MVAPICH2-GDR
Mellanox Interconnect	IB FDR	Connect-IB FDR	IB FDR

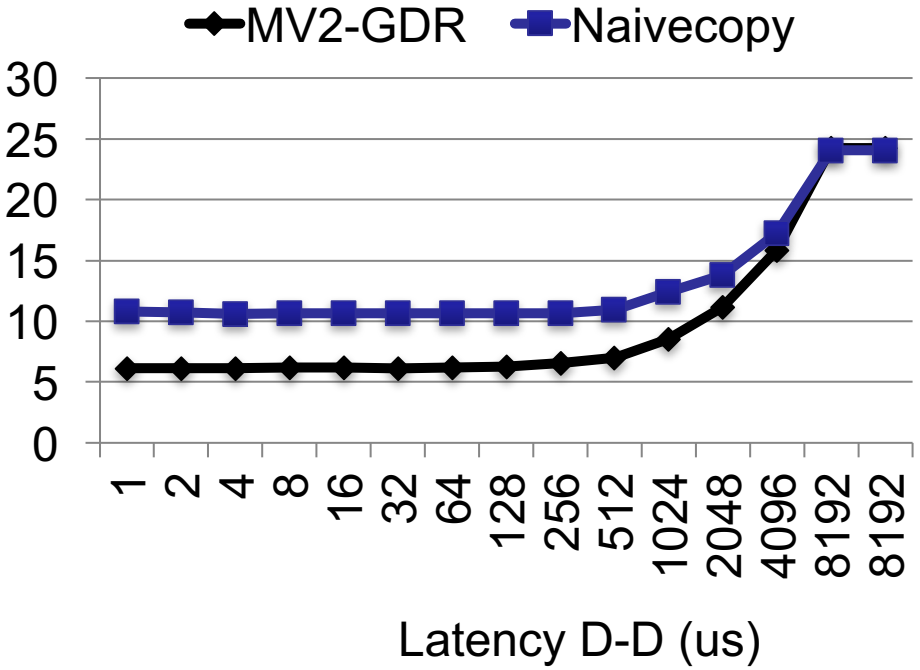
- MPI Library: MVAPICH2

High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)

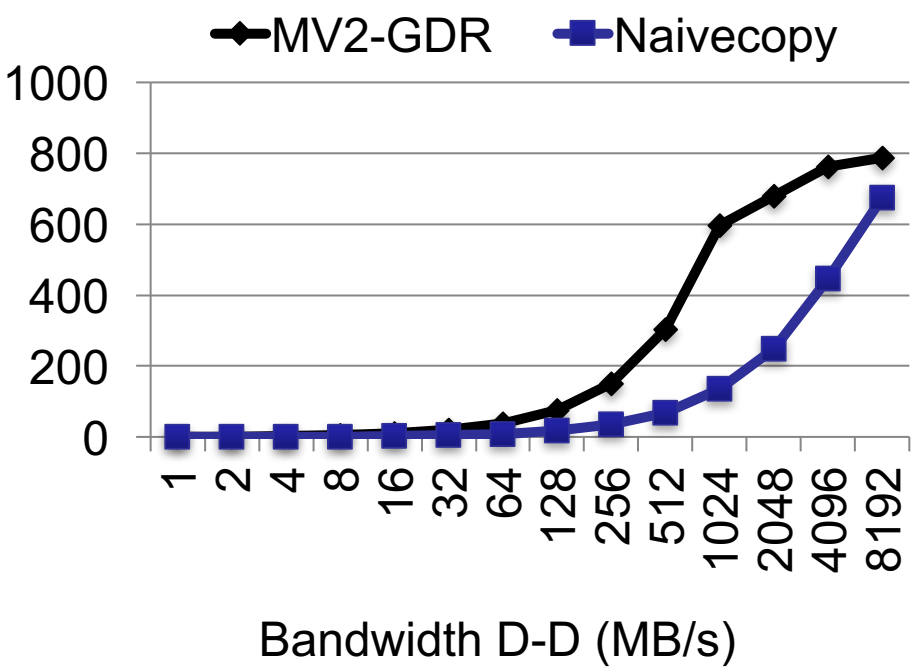
Used by more than 2,225 organizations (HPC Centers, Industry and Universities) in 74 countries

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

Comparison of Rendezvous GDR and Eager Naïve Designs



Latency D-D (us)

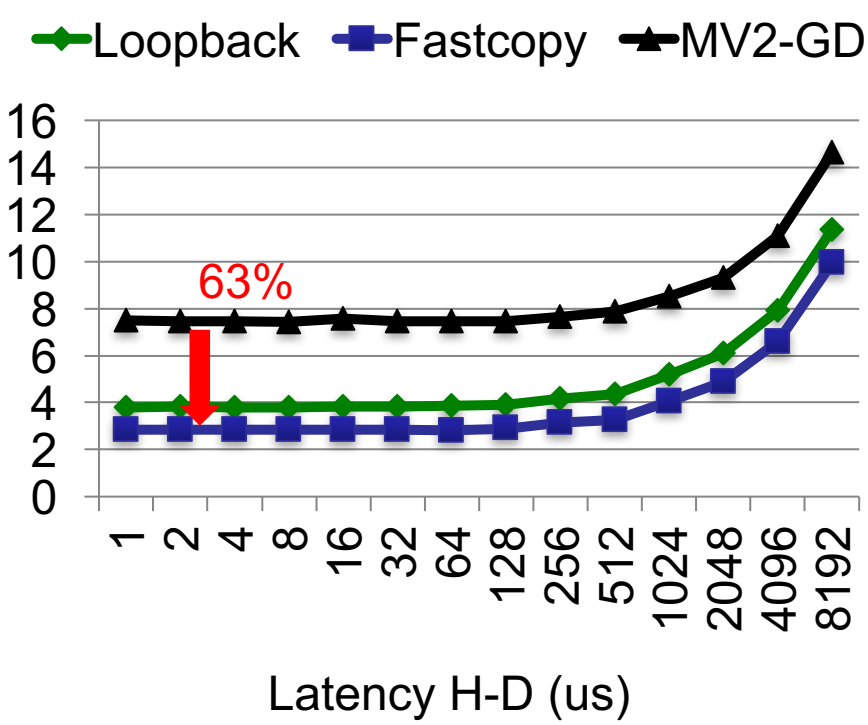


Bandwidth D-D (MB/s)

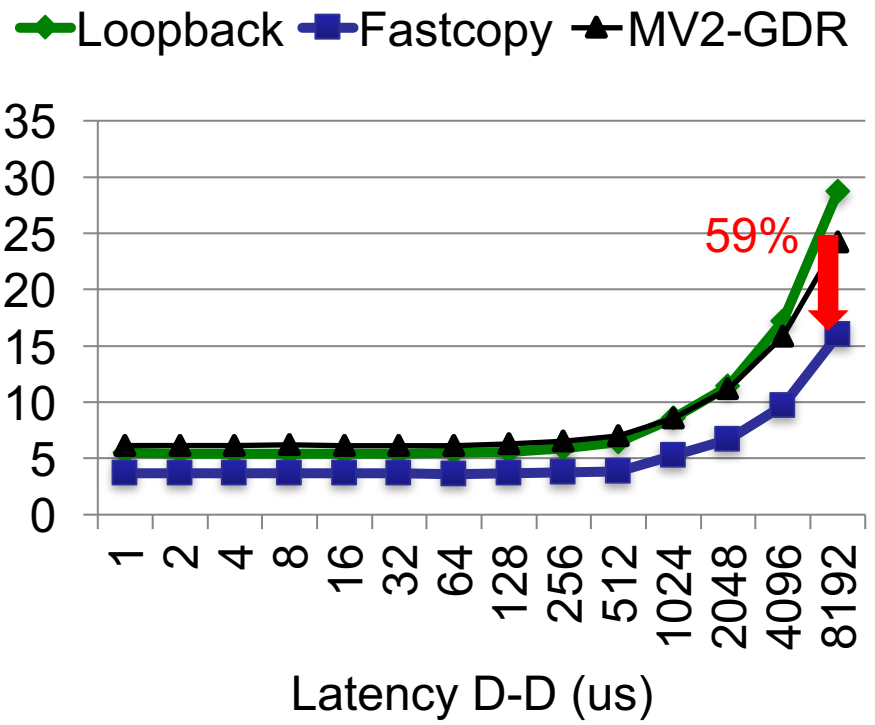
Expensive cudaMemcpy operation in Naïve copy (~8 us) and achieves only 30% of the peak bandwidth

Take existing rendezvous GDR protocol (MV2-GDR) as baseline

Latency with 2 GPUs on Cluster A



Latency H-D (us)

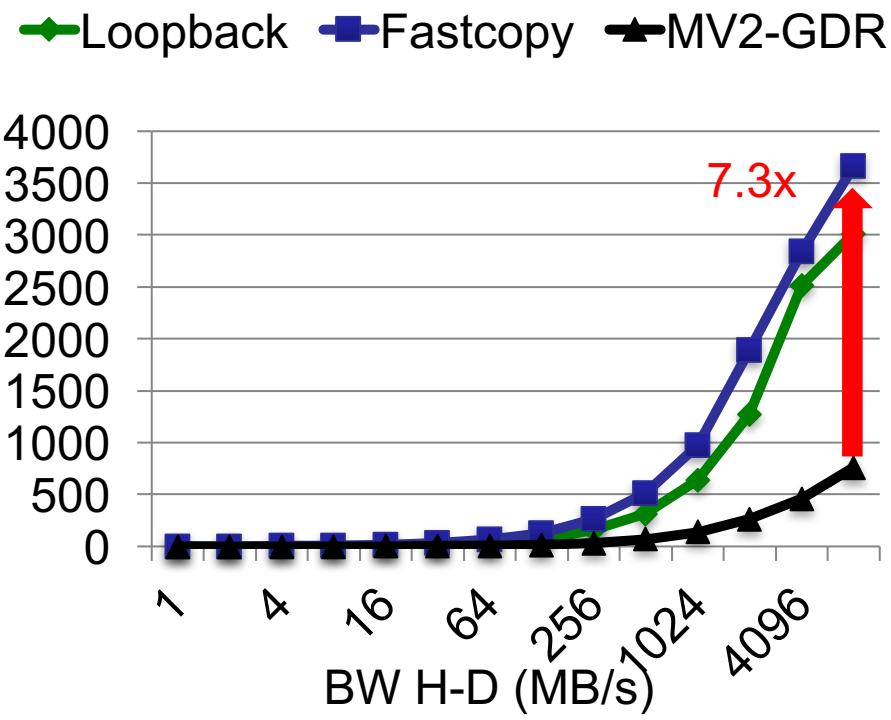


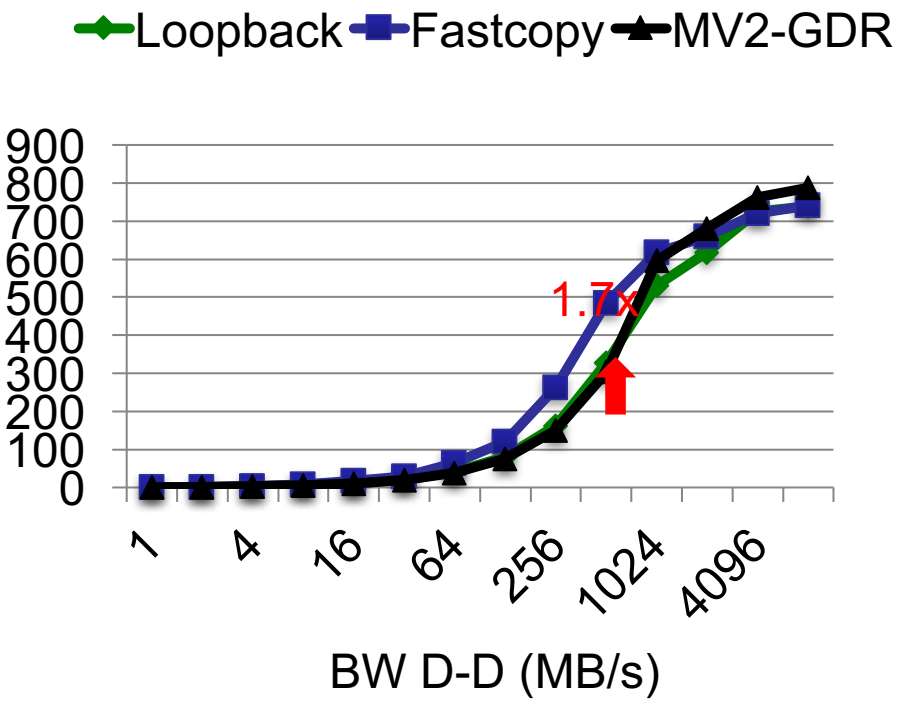
Latency D-D (us)

H-D: Loopback and Fastcopy achieve up to 50% and 63% reduction
Fastcopy achieves an additional 27% lower latency compared to Loopback

D-D: Loopback and Fastcopy achieve up to 12% and 59% reduction
Fastcopy achieves an additional 54% lower latency compared to Loopback

Bandwidth with 2 GPUs on Cluster A

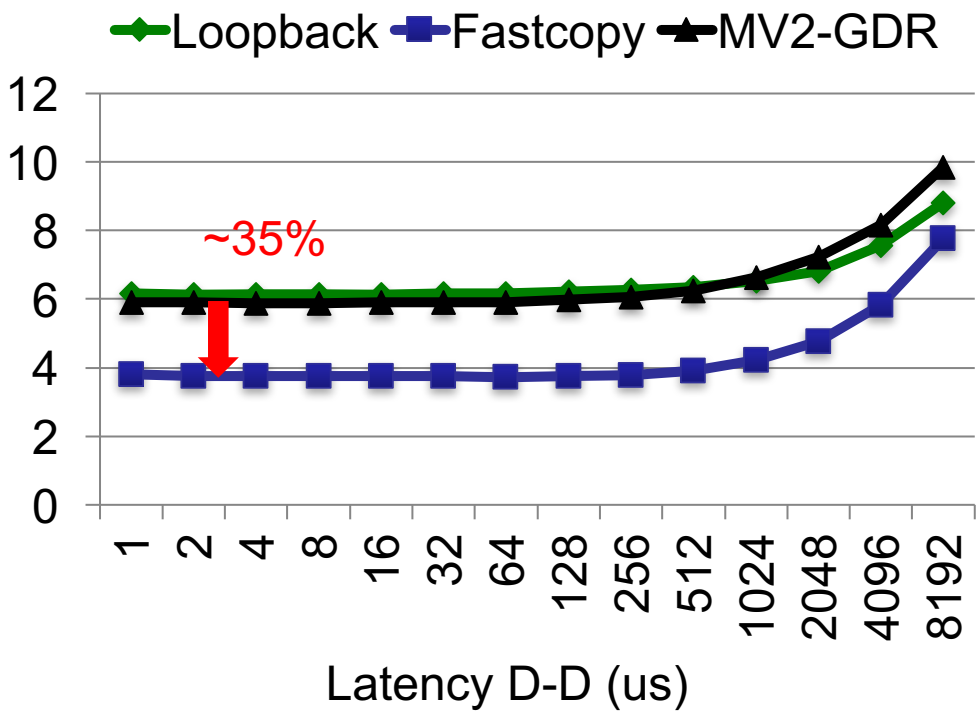


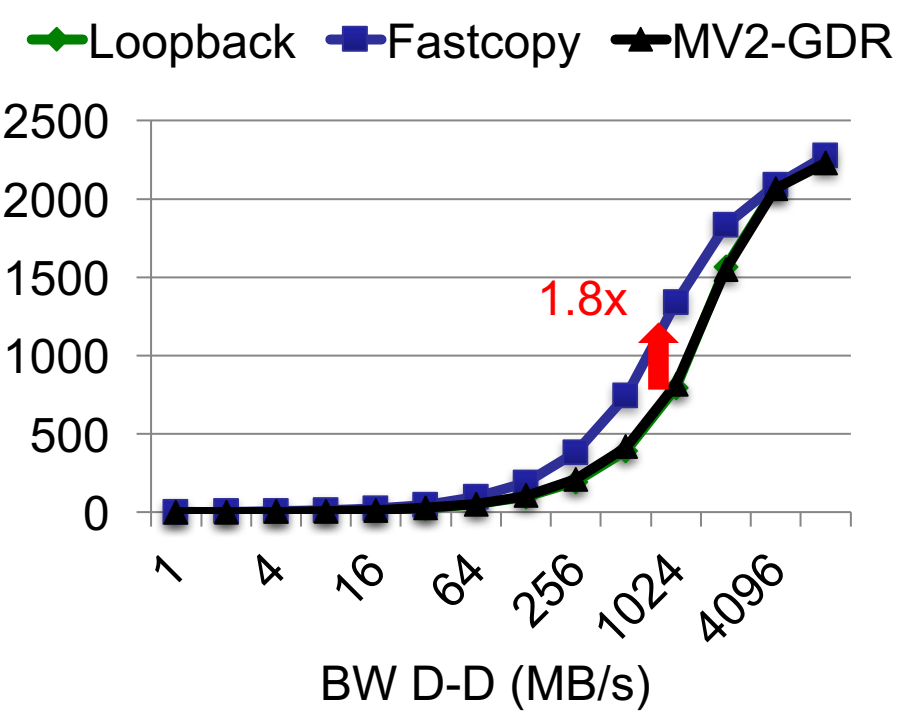


H-D: Loopback and Fastcopy achieve up to 5.5x and 7.3x bandwidth increase
Fastcopy gains an average of 1.8x improvement compared to Loopback

D-D: 1.1x and 1.7x bandwidth improvement, small "D-D" bandwidth increase comes from the limitation of Sandy Bridge

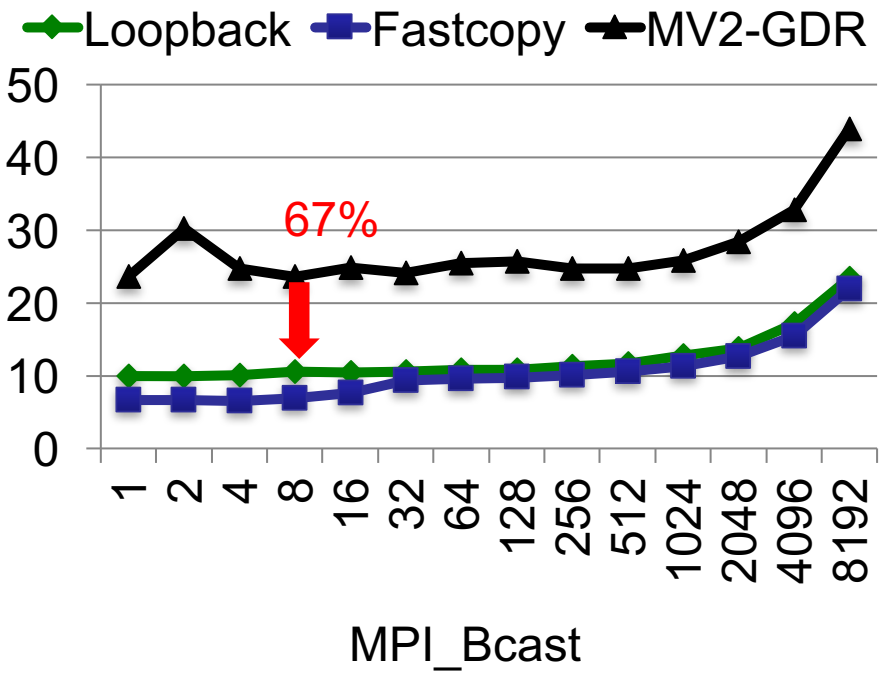
D-D Communication with 2 GPUs on Cluster B

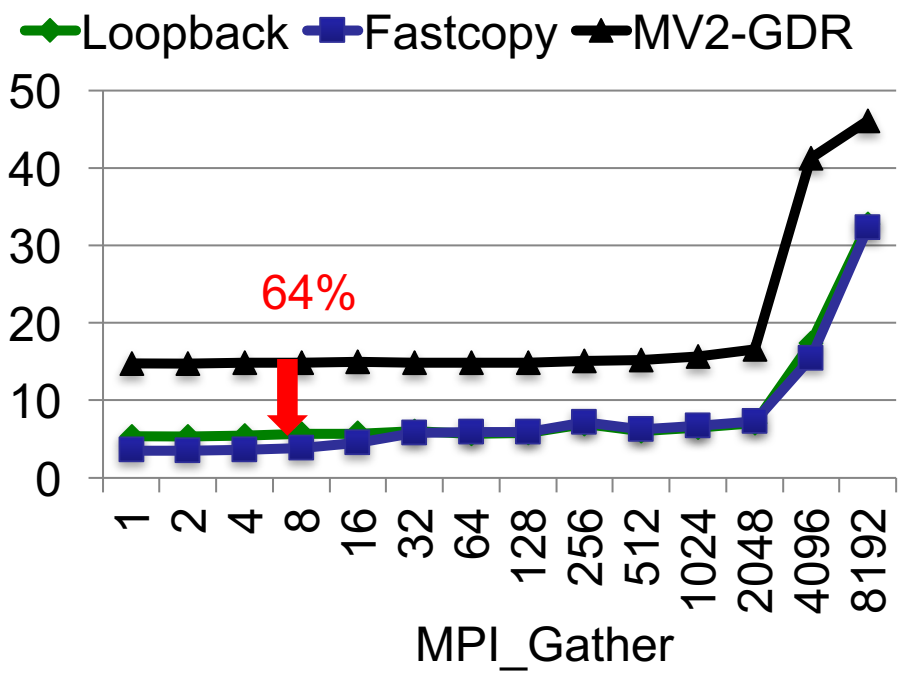




Loopback and Fastcopy achieve ~4% and ~35% lower latency on average
The PCIe switch (connecting the IB card and the GPU) improves the performance for D-D, and Sandy Bridge bandwidth cap at roughly 800 MB/s disappears and reaches 2.2GB/s

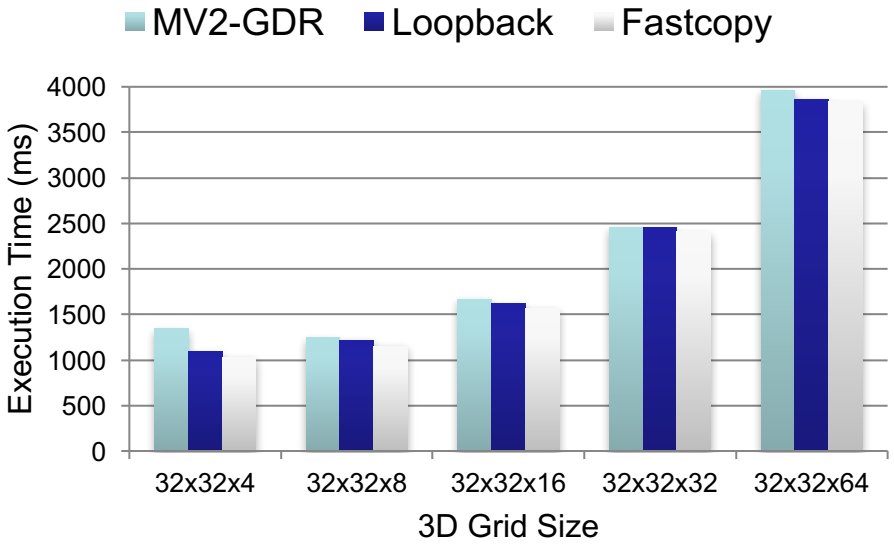
Evaluation of Collective with 16 GPUs on Wilkes



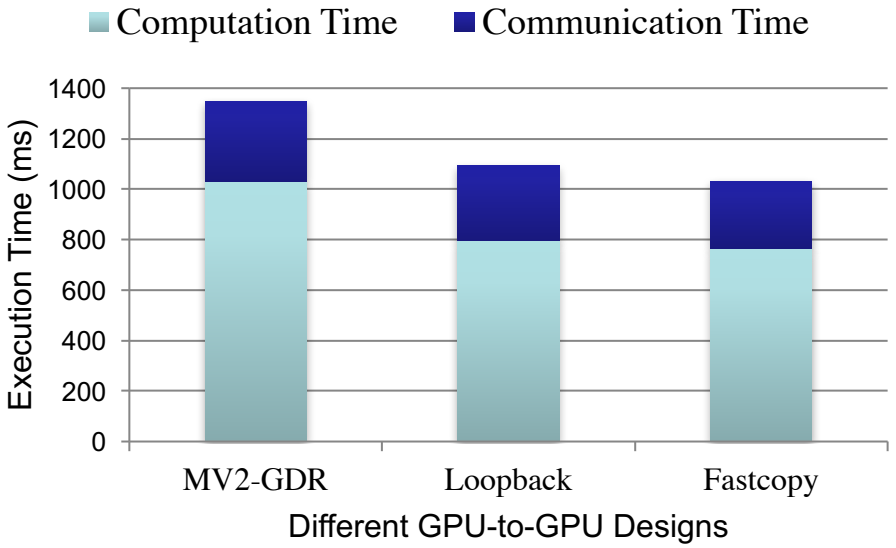


Loopback achieves up to 67% and 64% latency reduction in Bcast and Gather
Fastcopy delivers better performance (<16 bytes) because of the use of D-H Fastcopy on sender side

Evaluation of GPULBM with 2 GPUs on Cluster A



3D Grid with various dim_z



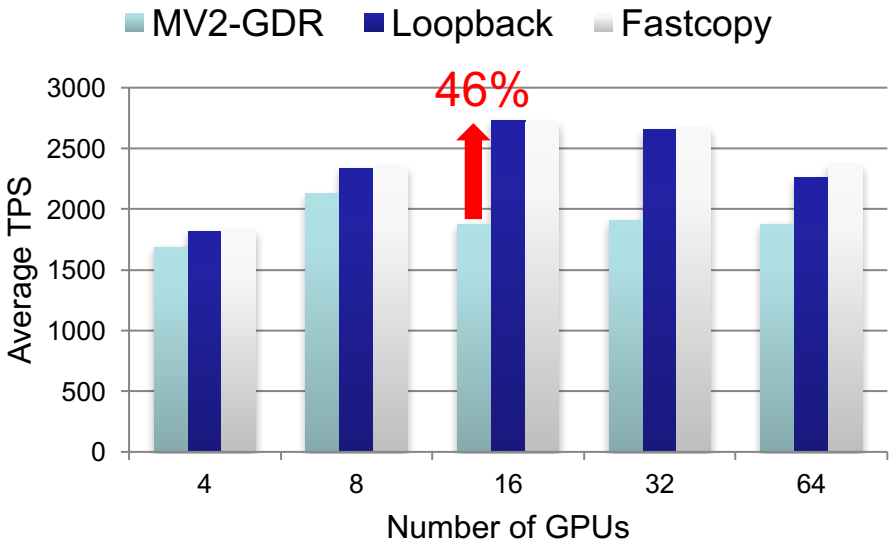
Decomposition Time of [32, 32, 4] Grid

GPULBM is a parallel distributed CUDA implementation of Lattice Boltzmann Method (LBM) for multiphase flows with large density ratios

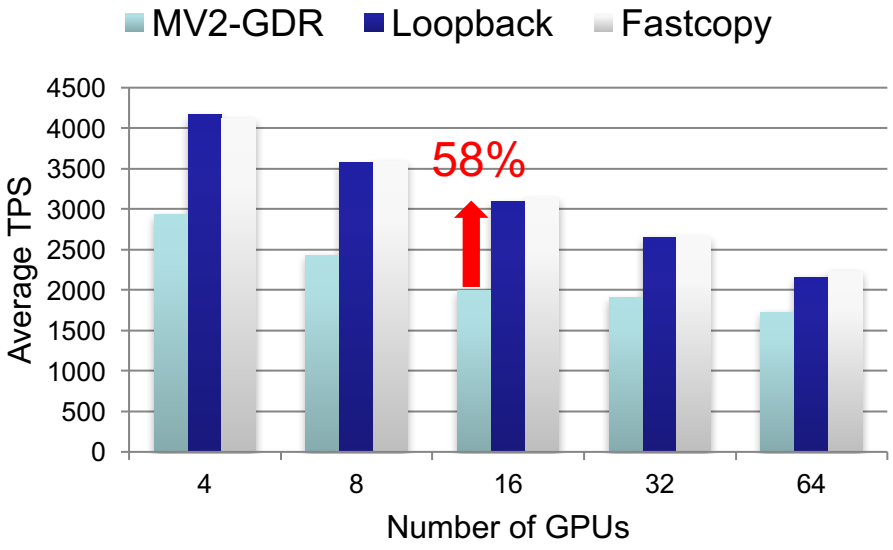
Loopback and Fastcopy designs achieve up to 18.8% and 23.4% reduction in total execution time for 32×32×4 3D grid size

In the decomposition time of 32x32x4 grid, 5.3% and 15.4% reductions in MPI communication time as well as better overlap

Evaluation of HOOMD-blue on Wilkes



Strong Scaling



Weak Scaling

HOOMD-Blue is a general-purpose particle simulation toolkit that is widely used in molecular dynamics areas

Strong Scaling: fixed 64K particles
Weak Scaling: fixed 2K particles/GPU ratio
For case of 16 GPU nodes, Loopback and Fastcopy achieves up to 46%, 45% and 55%, 58% increase on strong and weak scaling

Outline

- Introduction
- Motivation
- Proposed Design for GPU-to-GPU Eager Protocol
- Performance Evaluation
- Conclusion and Future Work

Conclusion

- Propose several techniques to improve the GPU-to-GPU communication using eager protocol
- the MVAPICH2 library can boost more efficient CUDA-Aware MPI communications achieving better latency and bandwidth
- Exhibit sustained strong & weak performance scalability
- Exhibit sustained performance efficiency scalability

Future Work

- Does the new eager protocol benefit other applications?
- How about the interaction of new design with the non-contiguous MPI derived datatype in GPU-to-GPU communication?

Acknowledgement

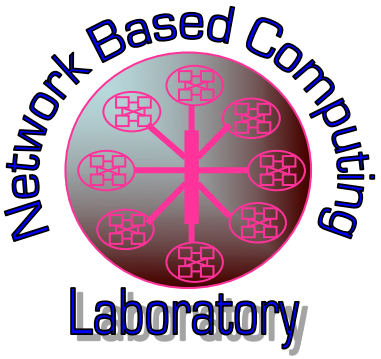
Mark Arnold (The Ohio State University)

Filippo Spiga (University of Cambridge)

Thank You!

{shir, potluri, hamidouc, perkinjo,limin, panda} @cse.ohio-state.edu

drossetti@nvidia.com



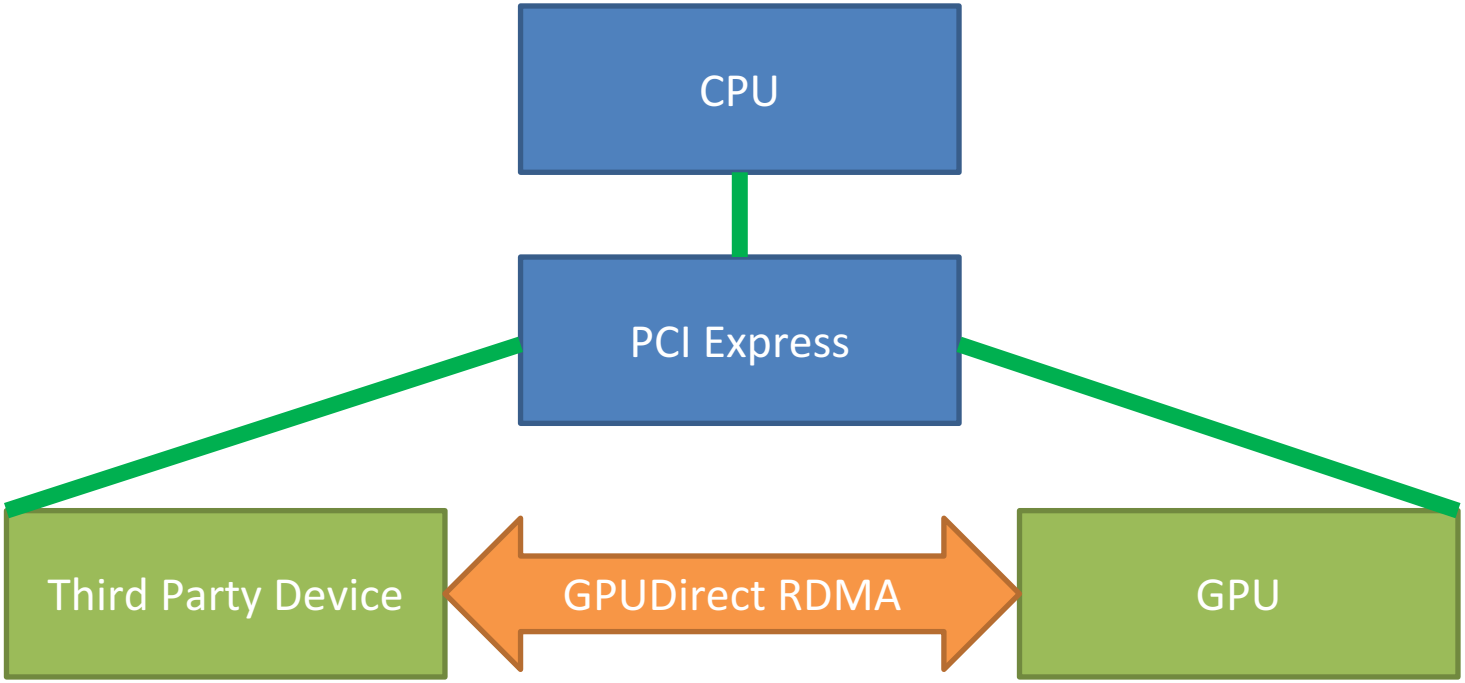
Network-Based Computing Laboratory

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

MVAPICH Web Page

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

GPU Direct RDMA



CUDA 4.0 enables the same host memory regions to be registered by both network adapter and GPU device

Communication over IB requires registration of the host buffers

avoid an additional copy in host memory when data copied from the GPU has to be transferred over the InfiniBand network

http://docs.nvidia.com/cuda/gpudirect-rdma/index.html#axzz3H1BvFmpl