Optimizing MPI Communication on Multi-GPU Systems using CUDA Inter-Process Communication

Sreeram Potluri* **Hao Wang*** **Devendar Bureddy***

Ashish Kumar Singh* **Carlos Rosales+** **Dhabaleswar K. Panda***

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

+Texas Advanced Computing Center

Outline

- Motivation
- Problem Statement
- Using CUDA IPC
- CUDA IPC based Designs in MVAPICH2
  - Two Sided Communication
  - One-sided Communication
- Experimental Evaluation
- Conclusion and Future Work

GPUs for HPC

- GPUs are becoming a common component of modern clusters – higher compute density and performance/watt
- 3 of the top 5 systems in the latest Top 500 list use GPUs

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.0
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 / 2010 Dawning	120640	1271.00	2984.30	2580.0
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.6

- Increasing number of HPC workloads are being ported to GPUs - many of these use MPI
- MPI libraries are being extended to support communication from GPU device memory

MVAPICH/MVAPICH2 for GPU Clusters

At Sender:

cudaMemcpy (sbuf, sdev)
MPI_Send (sbuf, . . .)

At Receiver:

MPI_Recv (rbuf, . . .)
cudaMemcpy (rdev, rbuf)

Earlier

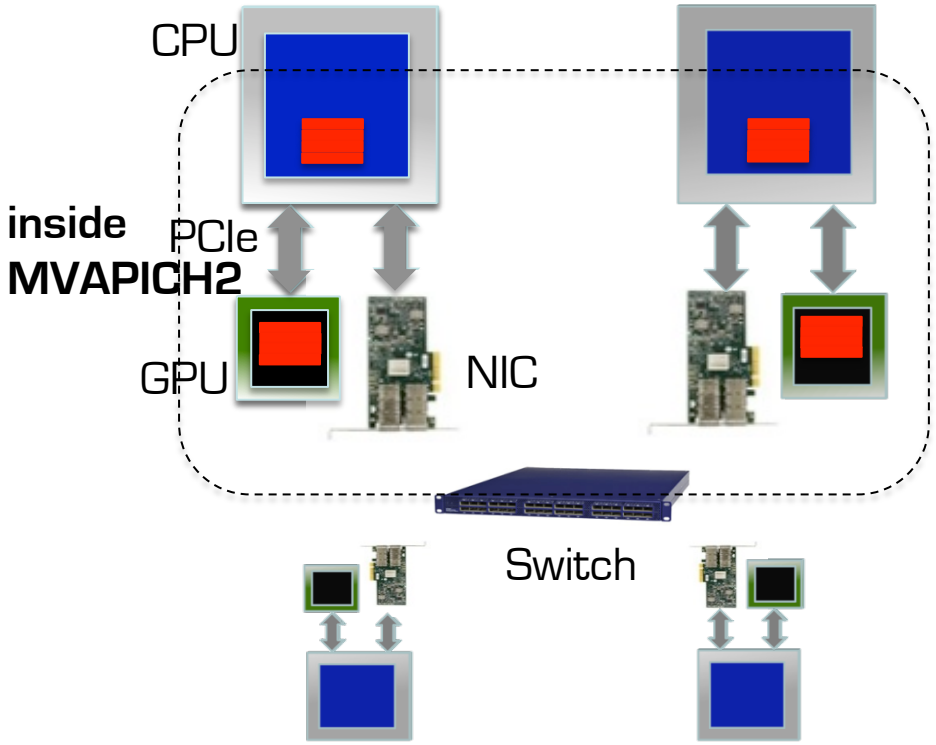
At Sender:

MPI_Send (sdev, . . .)

At Receiver:

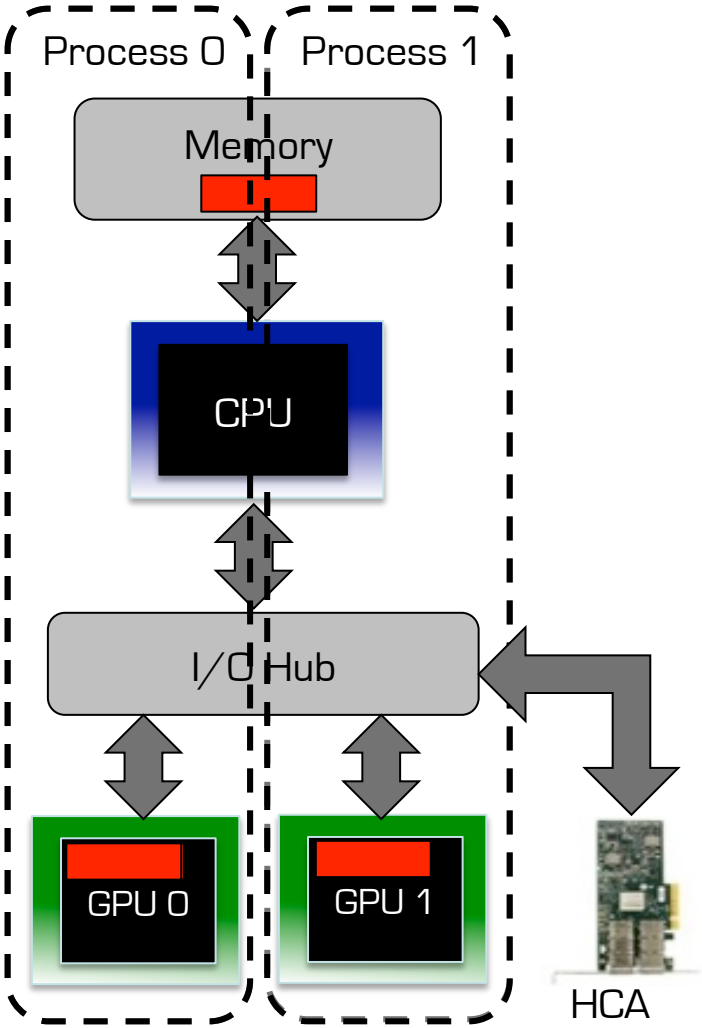
MPI_Recv (rdev, . . .)

Now



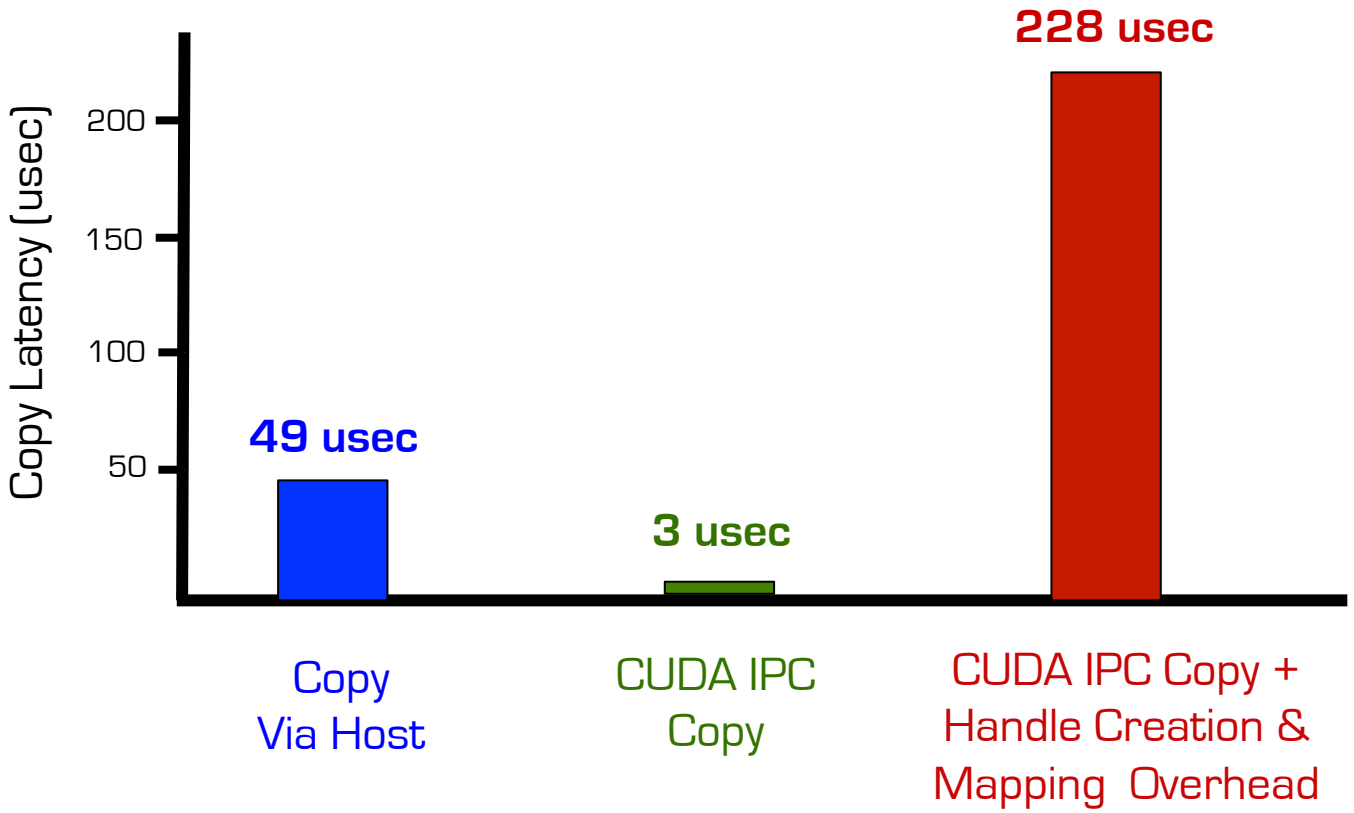
- Efficient overlap copies over the PCIe with RDMA transfers over the network
- Allows us to select efficient algorithms for MPI collectives and MPI datatype processing
- Available with MVAPICH2 v1.8 (http://mvapich.cse.ohio-state.edu)

Motivation



- Multi-GPU node architectures are becoming common
- Until CUDA 3.2
  - Communication between processes staged through the host
  - Shared Memory (pipelined)
  - Network Loopback [asynchronous]
- CUDA 4.0
  - Inter-Process Communication (IPC)
  - Host bypass
  - Handled by a DMA Engine
  - Low latency and Asynchronous
  - Requires creation, exchange and mapping of memory handles - overhead

Comparison of Costs



- Comparison of bare copy costs between two processes on one node, each using a different GPU (outside MPI)
- 8 Bytes

Outline

- Motivation
- Problem Statement
- Basics of CUDA IPC
- CUDA IPC based Designs in MVAPICH2
  - Two Sided Communication
  - One-sided Communication
- Experimental Evaluation
- Conclusion and Future Work

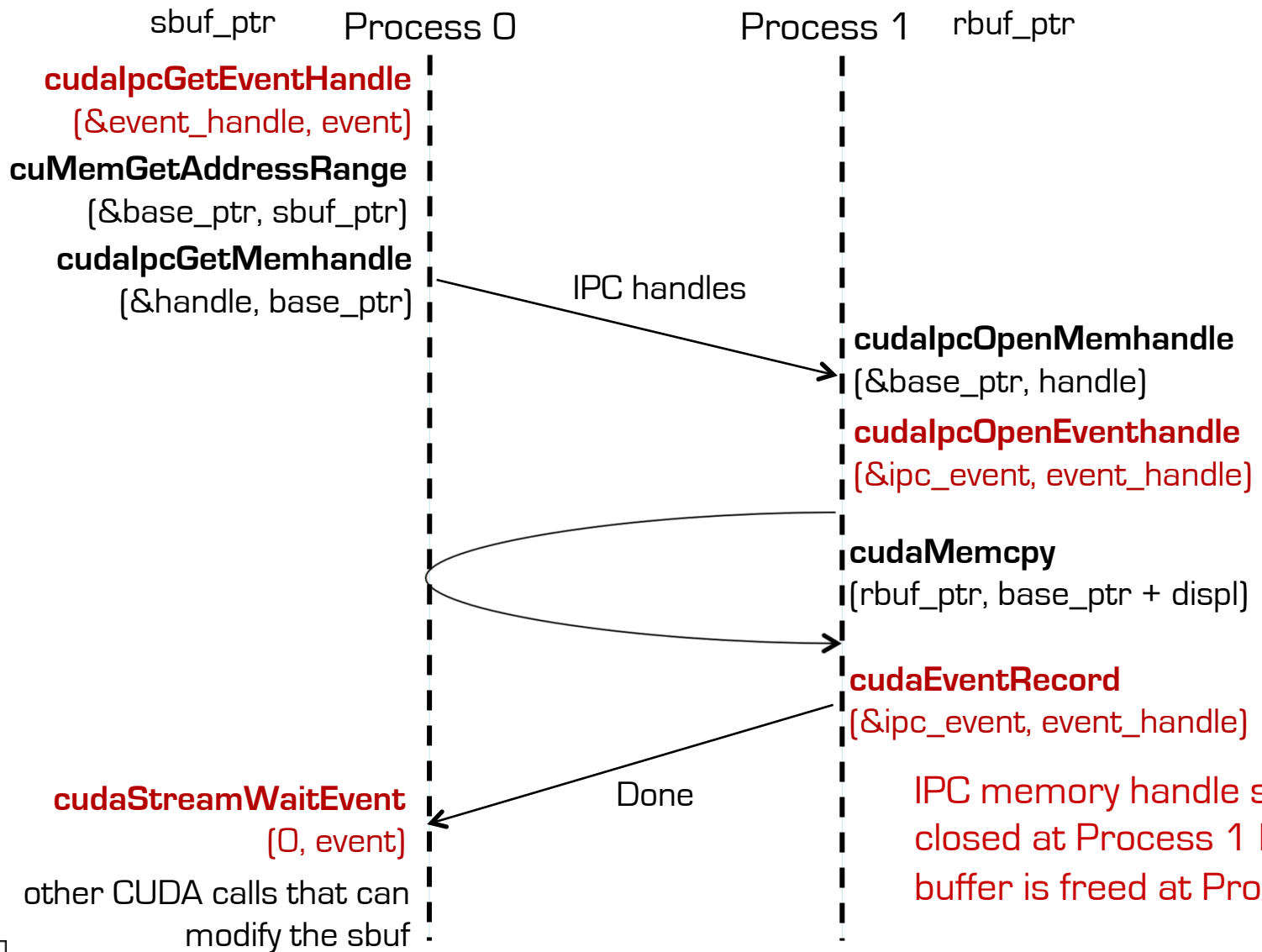
Problem Statement

- Can we take advantage of CUDA IPC to improve performance of MPI communication between GPUs on a node?
- How do we address the memory handle creation and mapping overheads?
- What kind of performance do the different MPI communication semantics deliver with CUDA IPC?
  - Two-sided Semantics
  - One-sided Semantics
- How do CUDA IPC based designs impact the performance of end-applications?

Outline

- Motivation
- Problem Statement
- Basics of CUDA IPC
- CUDA IPC based Designs in MVAPICH2
  - Two Sided Communication
  - One-sided Communication
- Experimental Evaluation
- Conclusion and Future Work

Basics of CUDA IPC



IPC memory handle should be closed at Process 1 before the buffer is freed at Process 0

Outline

- Motivation
- Problem Statement
- Basics of CUDA IPC
- CUDA IPC based Designs in MVAPICH2
  - Two Sided Communication
  - One-sided Communication
- Experimental Evaluation
- Conclusion and Future Work

Design of Two-sided Communication

- MPI communication costs
  - synchronization
  - data movement
- Small message communication
  - minimize synchronization overheads
  - pair-wise eager buffers for host-host communication
  - associated pair-wise IPC buffers on GPU
  - synchronization using CUDA Events
- Large message communication
  - minimize number for copies - rendezvous protocol
  - minimize memory mapping overheads using a mapping cache

Design of One-sided Communication

- Separates communication from synchronization
- Window
- Communication calls - put, get, accumulate
- Synchronization calls
  - active - fence, post-wait/start-complete
  - passive – lock-unlock
  - period between two synchronization calls is a communication epoch
- IPC memory handles created and mapped during window creation
- Put/Get implemented as cudaMemcpyAsync
- Synchronization using CUDA Events

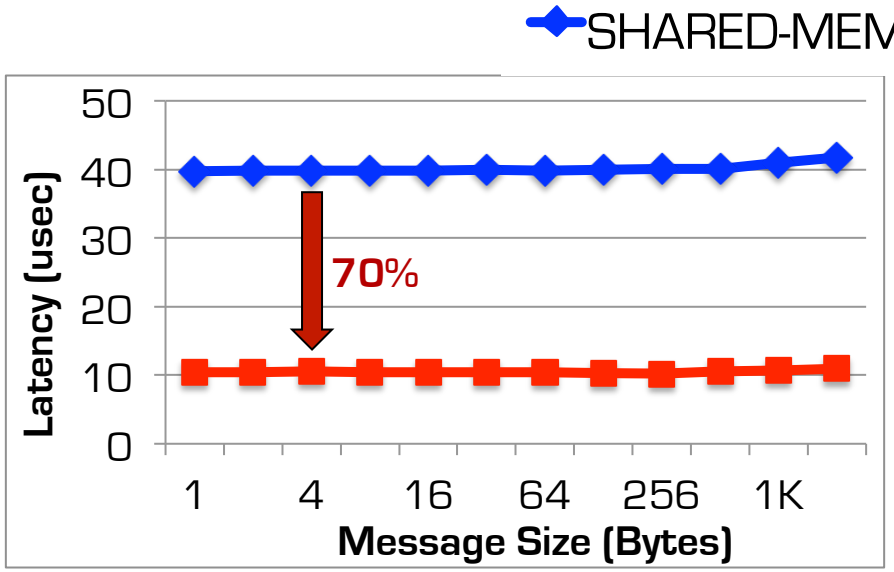
Outline

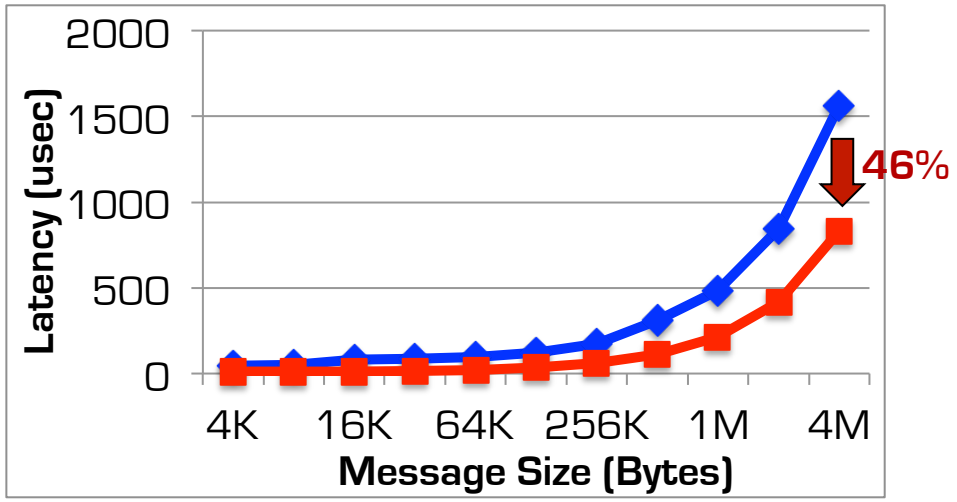
- Motivation
- Problem Statement
- Basics of CUDA IPC
- CUDA IPC based Designs in MVAPICH2
  - Two Sided Communication
  - One-sided Communication
- Experimental Evaluation
- Conclusion and Future Work

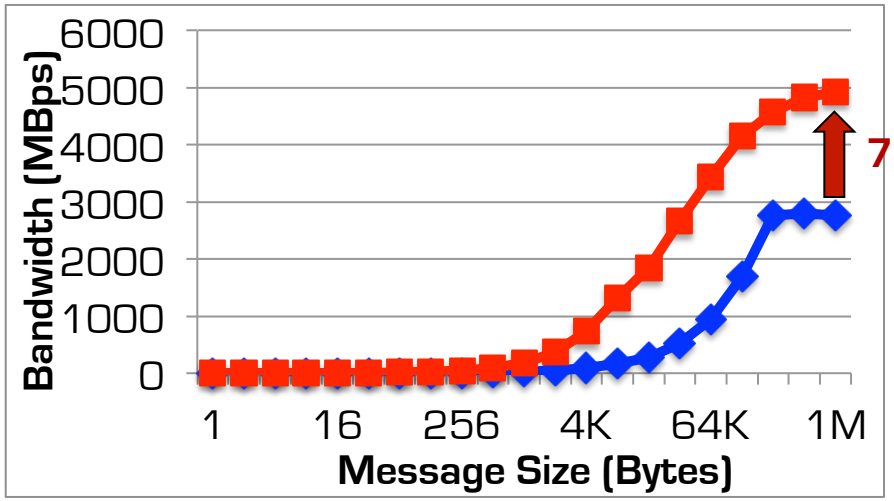
Experimental Setup

- Intel Westmere node
  - 2 NVIDIA Tesla C2075 GPUs
  - Red Hat Linux 5.8 and CUDA Toolkit 4.1
- MVAPICH/MVAPICH2 - High Performance MPI Library for IB, 10GigE/iWARP and RoCE
  - Available since 2002
  - Used by more than 1,930 organizations (HPC centers, Industries and Universities) in 68 countries
  - More than 111,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 5th ranked 73,278-core cluster (Tsubame 2.0) at Tokyo Institute of Technology
    - 7th ranked 111,104-core cluster (Pleiades) at NASA
    - 25th ranked 62,976-core cluster (Ranger) at TACC
  - http://mvapich.cse.ohio-state.edu

Two-sided Communication Performance



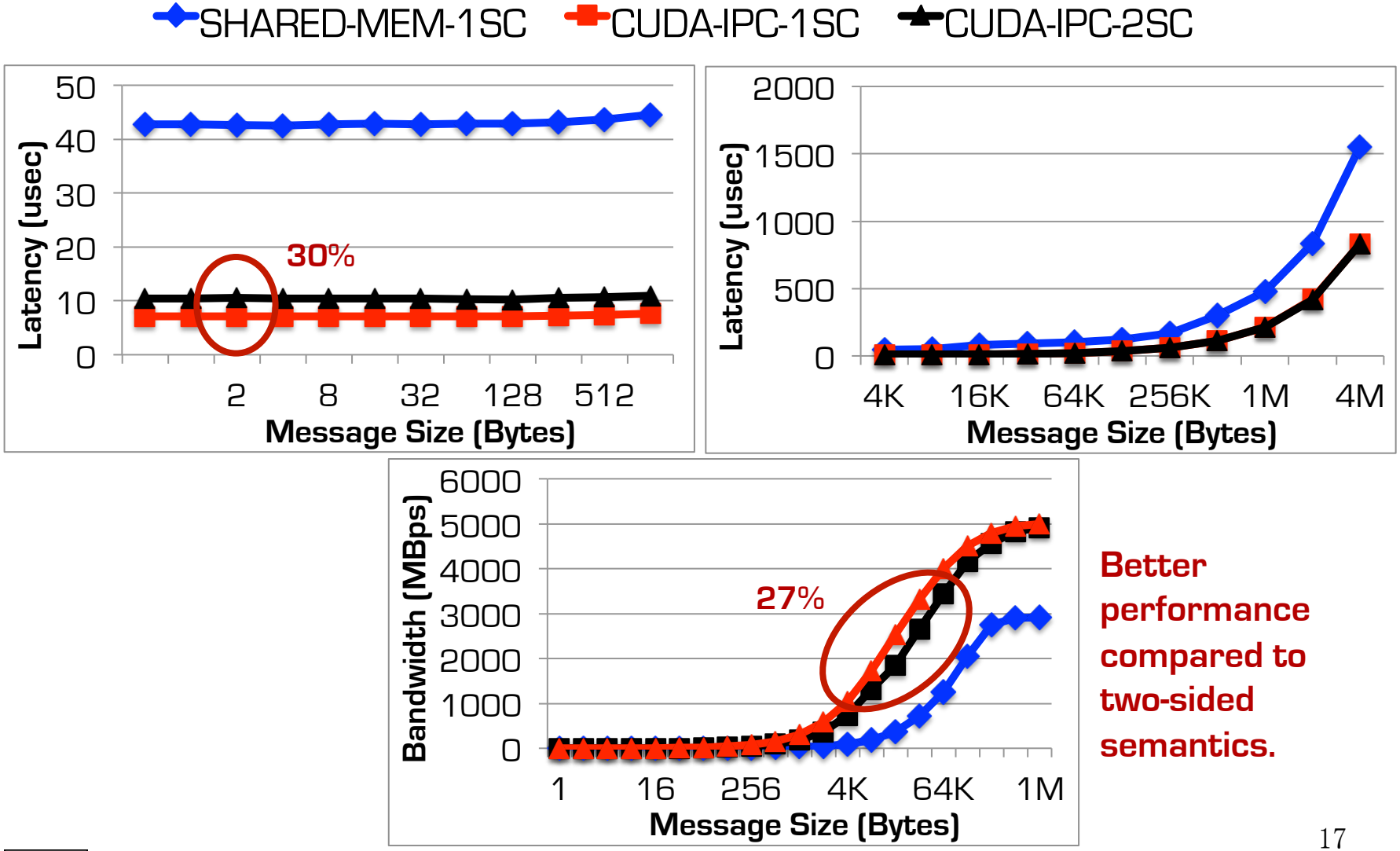




considerable improvement in MPI performance due to host bypass

One-sided Communication Performance

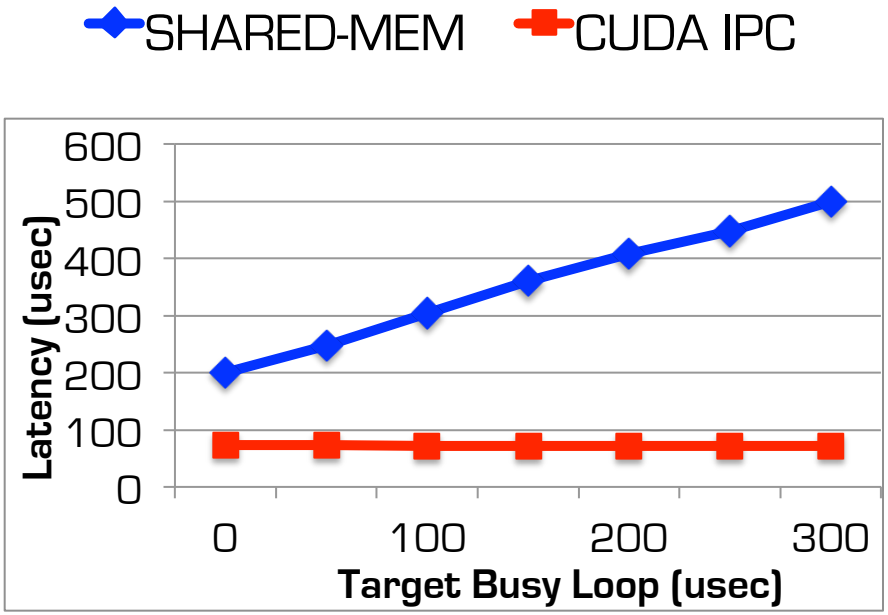
(get + active synchronization vs. send/recv)



Better performance compared to two-sided semantics.

One-sided Communication Performance

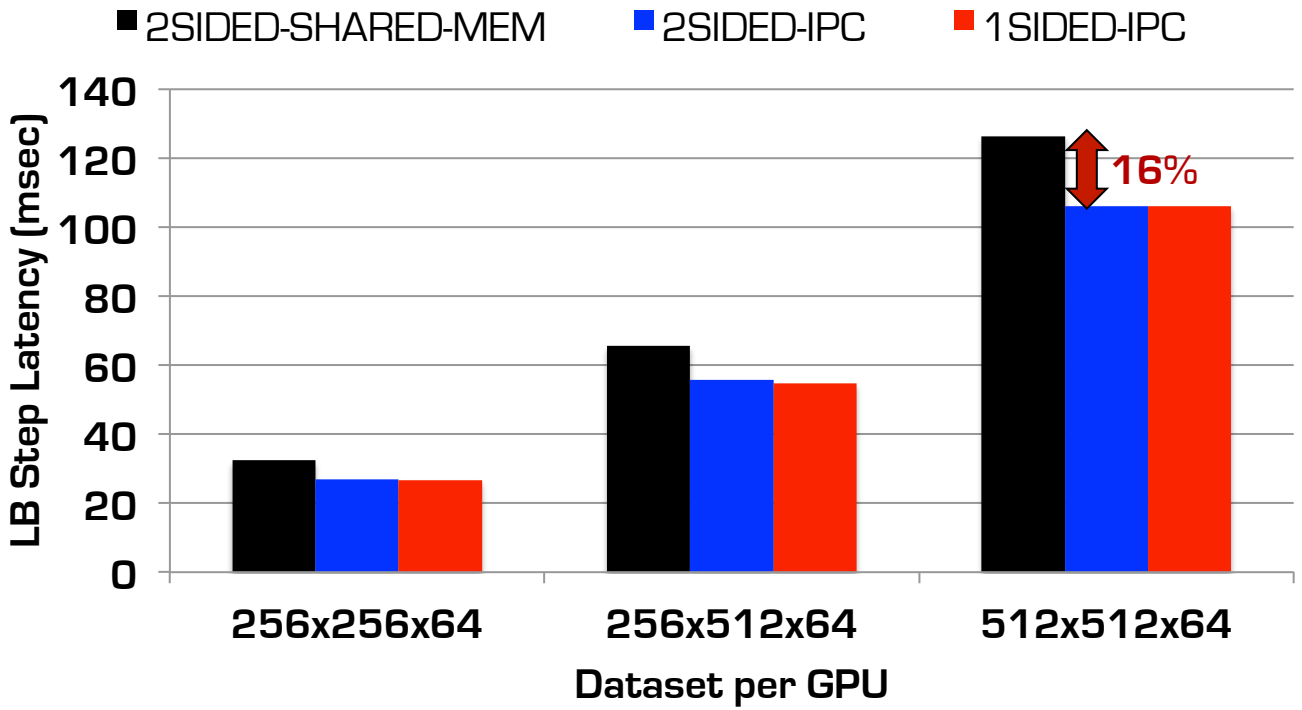
(get + passive synchronization)



true asynchronous progress

- Lock + 8 Gets + Unlock with the target in a busy loop (128KB messages)

Lattice Boltzmann Method



- Computation fluid dynamics code with support for multi-phase flows with large density ratios
- Modified to use MPI communication from GPU device memory - one-sided and two-sided semantics
- Up to 16% improvement in per step

Outline

- Motivation
- Problem Statement
- Basics of CUDA IPC
- CUDA IPC based Designs in MVAPICH2
  - Two Sided Communication
  - One-sided Communication
- Experimental Evaluation
- Conclusion and Future Work

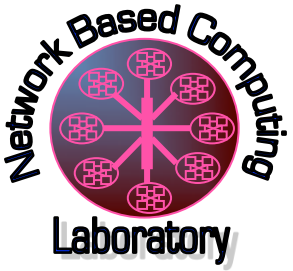
Conclusion and Future Work

- Take advantage of CUDA IPC to improve MPI communication between GPUs on a node
- 70% improvement in latency and 78% improvement in bandwidth for two-sided communication
- One-sided communication gives better performance and allows for truly asynchronous communication
- 16% improvement in execution time of Lattice Boltzmann Method code
- Studying the impact on other applications while exploiting computation-communication overlap
- Exploring efficient designs for inter-node one-sided communication on GPU clusters

Thank You!

{potluri, wangh, bureddy, singhas, panda} @cse.ohio-state.edu

carlos@tacc.utexas.edu



Network-Based Computing Laboratory

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

MVAPICH Web Page

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