MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Latest Advances in MVAPICH2 MPI Library for NVIDIA GPU Clusters with InfiniBand

Presented at GTC ’15

MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Presented by

Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

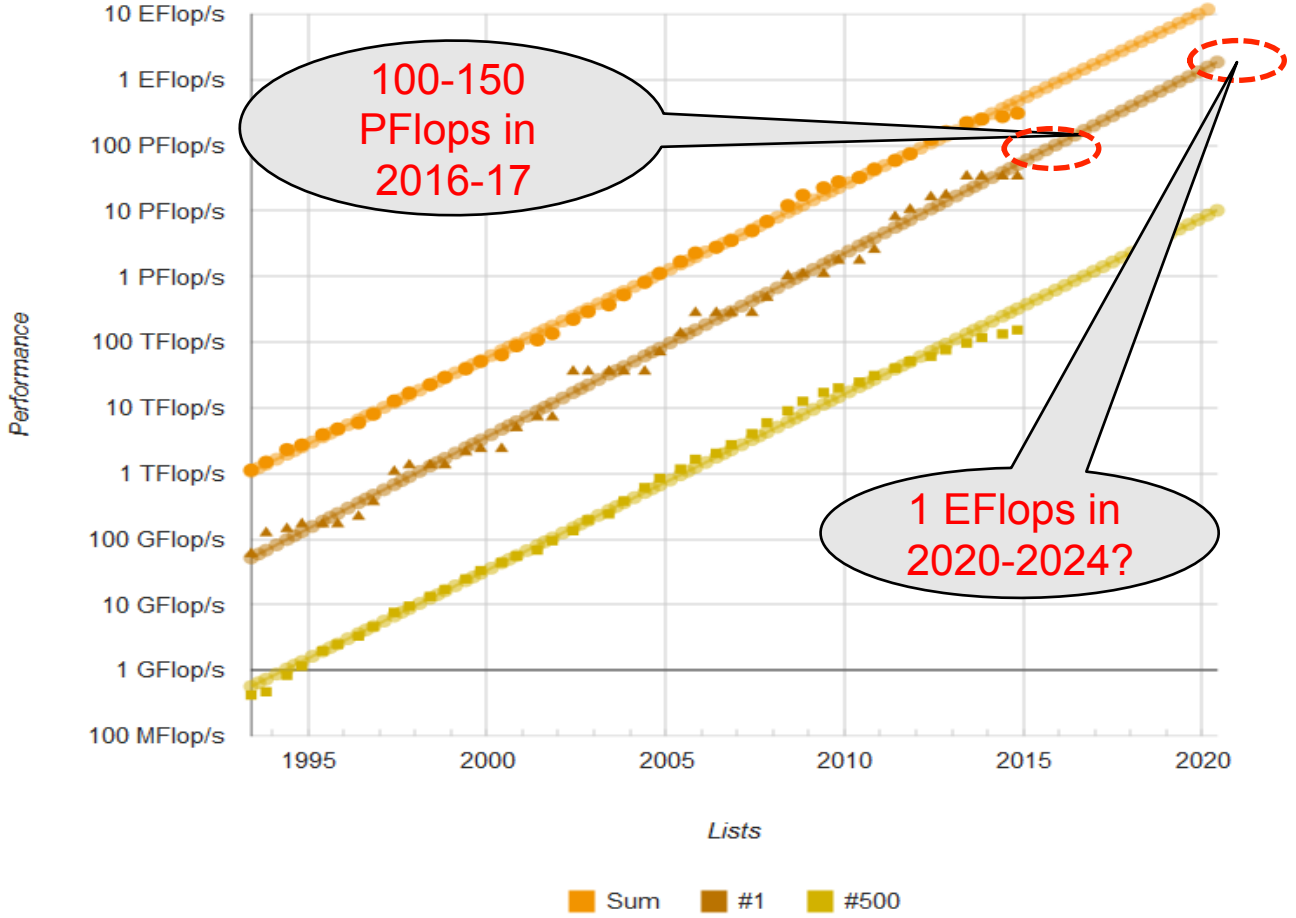
Khaled Hamidouche

The Ohio State University

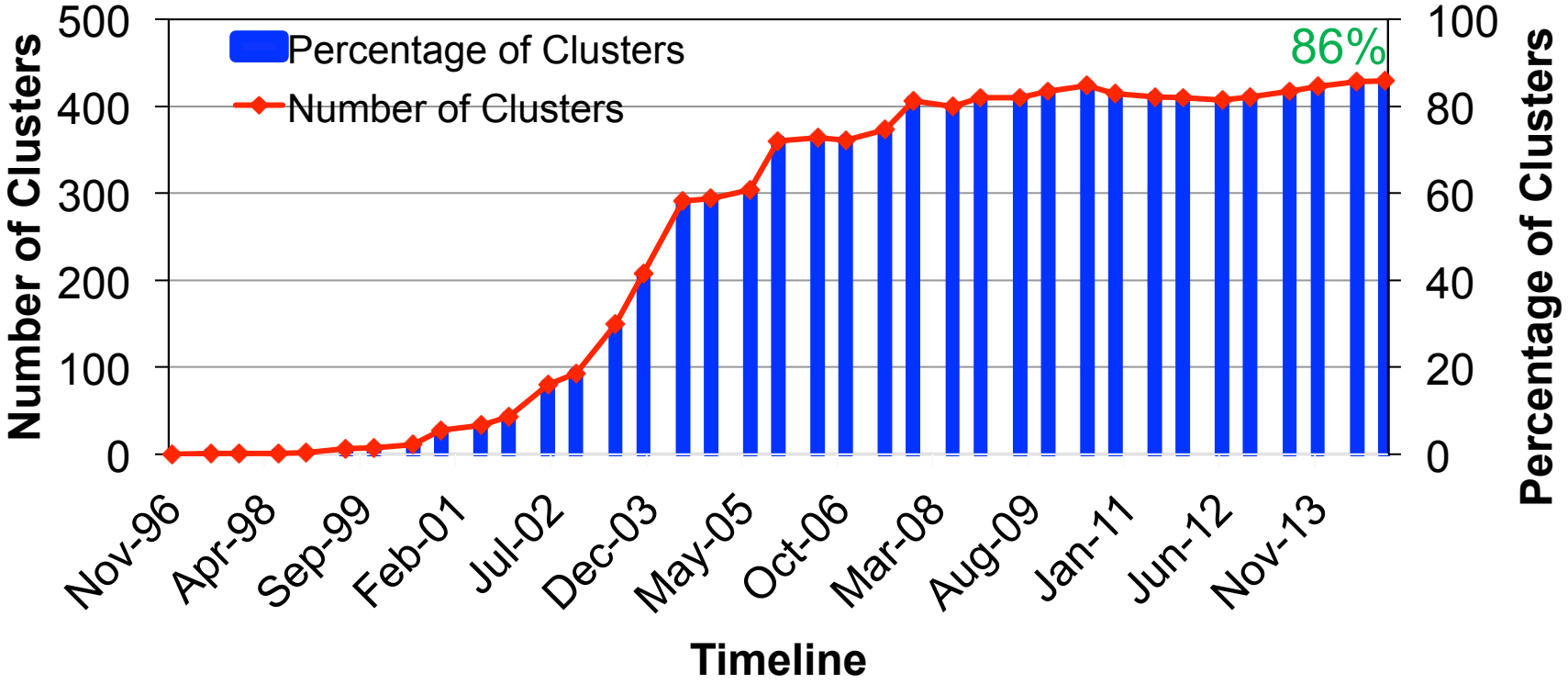
E-mail: hamidouc@cse.ohio-state.edu

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

High-End Computing (HEC): Petaflop to Petaflop



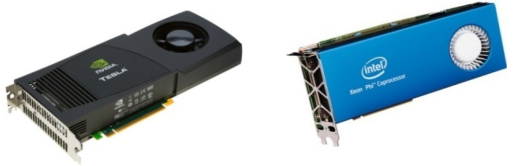
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)



Drivers of Modern HPC Cluster Architectures

Multi-core Processors

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



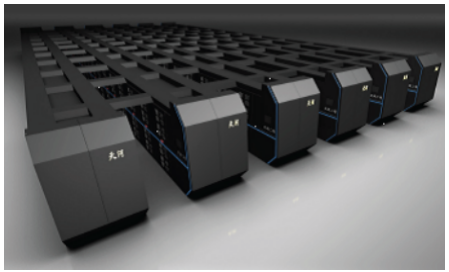
Accelerators / Coprocessors
high compute density, high performance/watt
>1 TFlop DP on a chip

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



Tianhe – 2

Titan

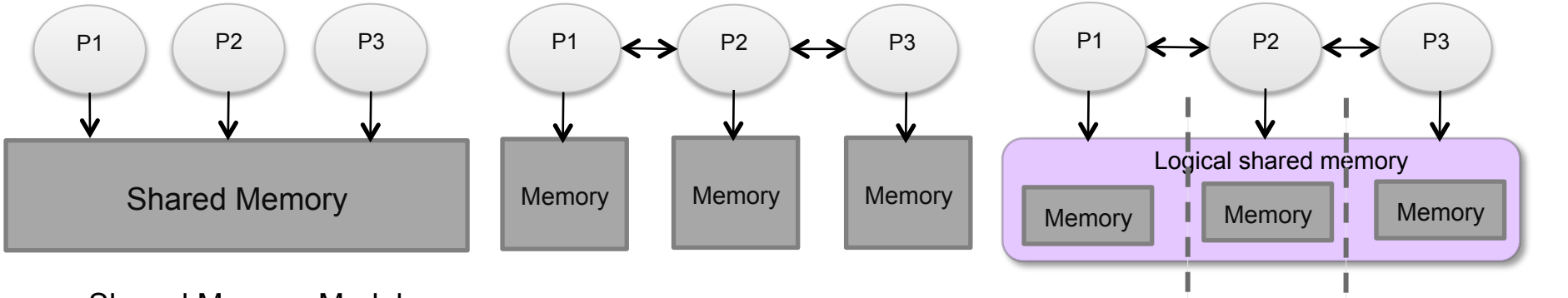
Stampede

Tianhe – 1A

225 IB Clusters (45%) in the November 2014 Top500 list (http://www.top500.org)

Installations in the Top 50 (21 systems):

519,640 cores (Stampede) at TACC (7th)	73,584 cores (Spirit) at USA/Air Force (31st)
72,800 cores Cray CS-Storm in US (10th)	77,184 cores (Curie thin nodes) at France/CEA (33rd)
160,768 cores (Pleiades) at NASA/Ames (11th)	65,320-cores, iDataPlex DX360M4 at Germany/Max-Planck (34th)
72,000 cores (HPC2) in Italy (12th)	120, 640 cores (Nebulae) at China/NSCS (35th)
147,456 cores (Super MUC) in Germany (14th)	72,288 cores (Yellowstone) at NCAR (36th)
76,032 cores (Tsubame 2.5) at Japan/GSIC (15th)	70,560 cores (Helios) at Japan/IFERC (38th)
194,616 cores (Cascade) at PNNL (18th)	38,880 cores (Cartesisu) at Netherlands/SURFsara (45th)
110,400 cores (Pangea) at France/Total (20th)	23,040 cores (QB-2) at LONI/US (46th)
37,120 cores T-Platform A-Class at Russia/MSU (22nd)	138,368 cores (Tera-100) at France/CEA (47th)
50,544 cores Occigen at France/GENCI-CINES (26th)	**and many more!**



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

Global Arrays, UPC, Chapel, X10, CAF, …

Programming models provide abstract machine models

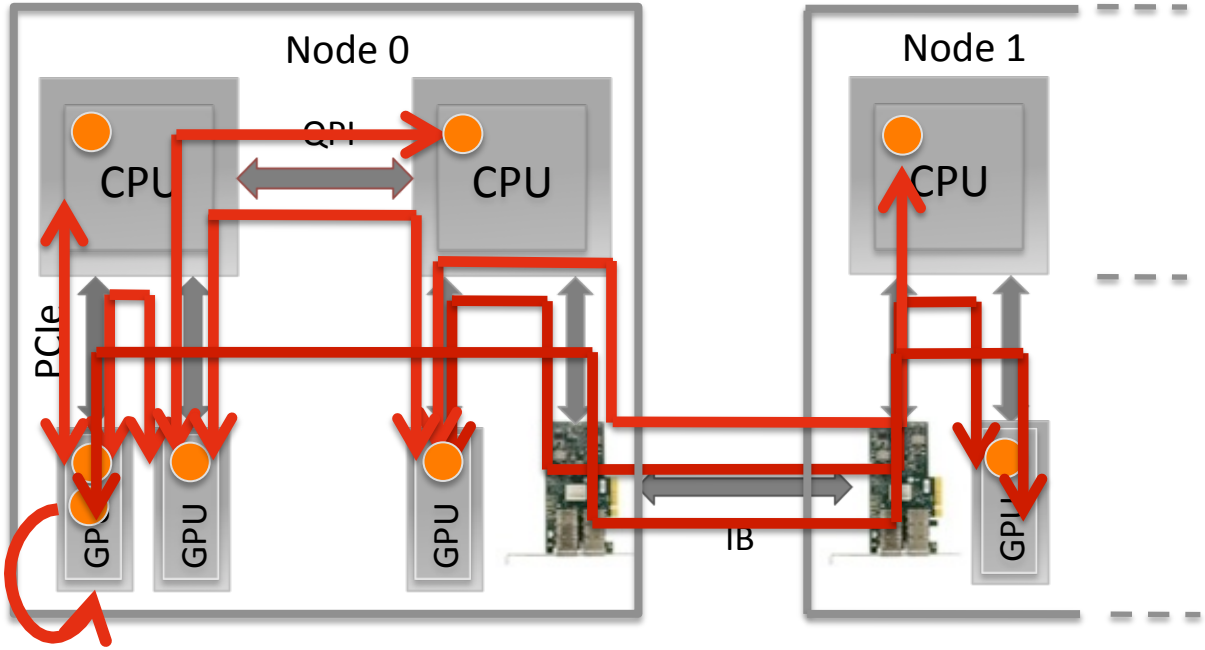
Models can be mapped on different types of systems

e.g. Distributed Shared Memory (DSM), MPI within a node, etc.

- **High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP and RDMA over Converged Enhanced Ethernet (RoCE)**
  - MVAPICH (MPI-1) ,MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Support for NVIDIA GPUs, Available since 2011
  - Used by more than 2,350 organizations (HPC Centers, Industry and Universities) in 75 countries
  - More than 241,000 downloads from OSU site directly
  - Empowering many TOP500 clusters (Nov '14 ranking)
    - 7th ranked 519,640-core cluster (Stampede) at TACC
    - 11th ranked 160,768-core cluster (Pleiades) at NASA
    - 15th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - Available with software stacks of many IB, HSE and server vendors including Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions

- Connected as PCIe devices – Flexibility but Complexity



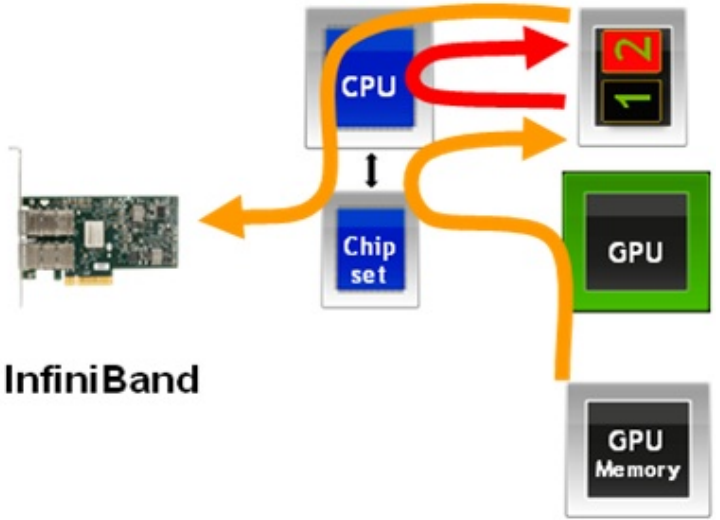
Memory buffers

1. Intra-GPU
2. Intra-Socket GPU-GPU
3. Inter-Socket GPU-GPU
4. Inter-Node GPU-GPU
5. Intra-Socket GPU-Host
6. Inter-Socket GPU-Host
7. Inter-Node GPU-Host
8. Inter-Node GPU-GPU with IB adapter on remote socket

and more . . .

- For each path different schemes: Shared_mem, IPC, GPUDirect RDMA, pipeline …
- Critical for runtimes to optimize data movement while hiding the complexity

- Many systems today want to use systems that have both GPUs and high-speed networks such as InfiniBand
- Problem: Lack of a common memory registration mechanism
  - Each device has to pin the host memory it will use
  - Many operating systems do not allow multiple devices to register the same memory pages
- Previous solution:
  - Use different buffers for each device and copy data



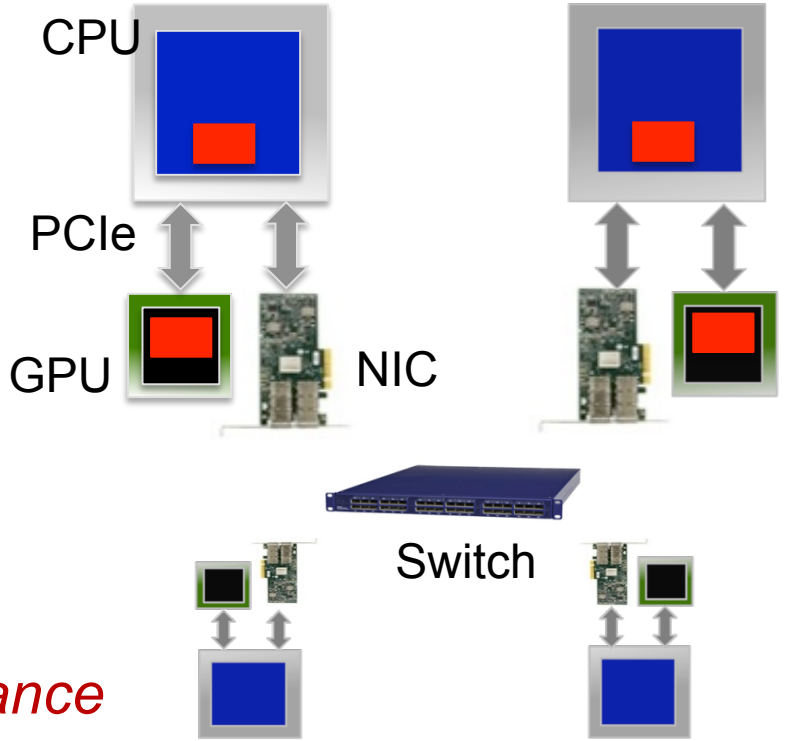
Naïve implementation with standard MPI and CUDA

At Sender:

```
cudaMemcpy(sbuf, sdev, . . .);
MPI_Send(sbuf, size, . . .);
```

At Receiver:

```
MPI_Recv(rbuf, size, . . .);
cudaMemcpy(rdev, rbuf, . . .);
```



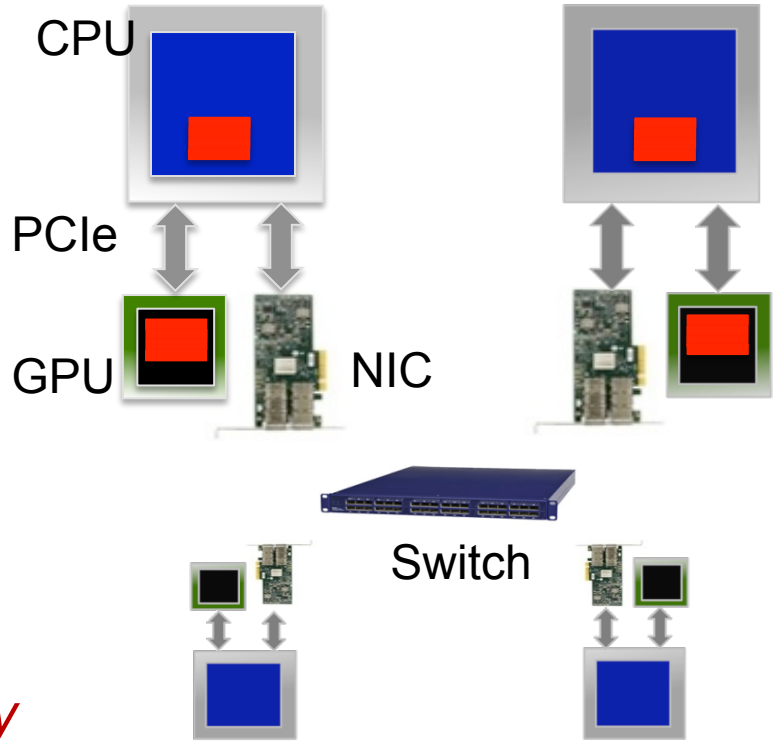
High Productivity and Poor Performance

Pipelining at user level with non-blocking MPI and CUDA interfaces

Code at Sender side (and repeated at Receiver side)

At Sender:

```
for (j = 0; j < pipeline_len; j++)
cudaMemcpyAsync(sbuf + j * blk, sdev + j *
        blksz,. . .);
for (j = 0; j < pipeline_len; j++) {
        while (result != cudaSucess) {
              result = cudaStreamQuery(…);
               if(j > 0) MPI_Test(…);
        }
        MPI_Isend(sbuf + j * block_sz, blksz . . .);
}
MPI_Waitall();
```



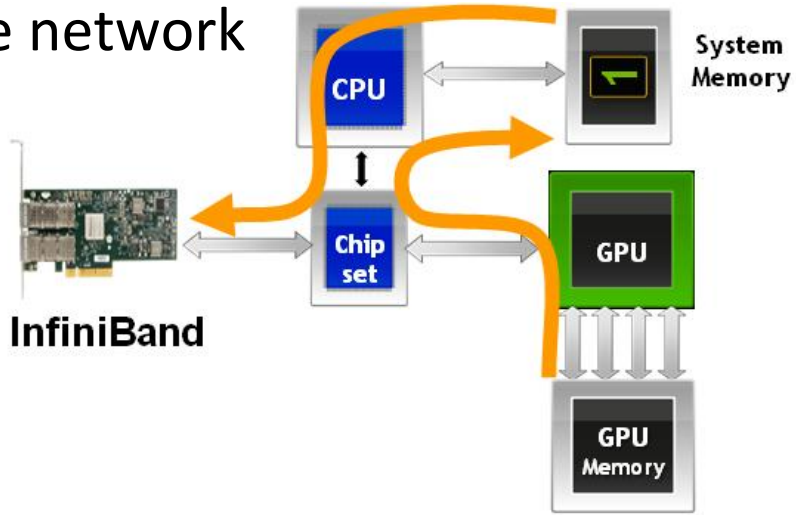
User-level copying may not match with internal MPI design

High Performance and Poor Productivity

- Support GPU to GPU communication through standard MPI interfaces
  - e.g. enable MPI_Send, MPI_Recv from/to GPU memory
- Provide high performance without exposing low level details to the programmer
  - Pipelined data transfer which automatically provides optimizations inside MPI library without user tuning
- A new Design incorporated in MVAPICH2 to support this functionality

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection) MPI and OpenACC
- Conclusions

- Collaboration between Mellanox and NVIDIA to converge on one memory registration technique
- Both devices register a common host buffer
- GPU copies data to this buffer, and the network adapter can directly read from this buffer (or vice-versa)



- Note that GPU-Direct does not allow you to bypass host memory

- MVAPICH2-GPU: standard MPI interfaces used
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU
- with RDMA transfers

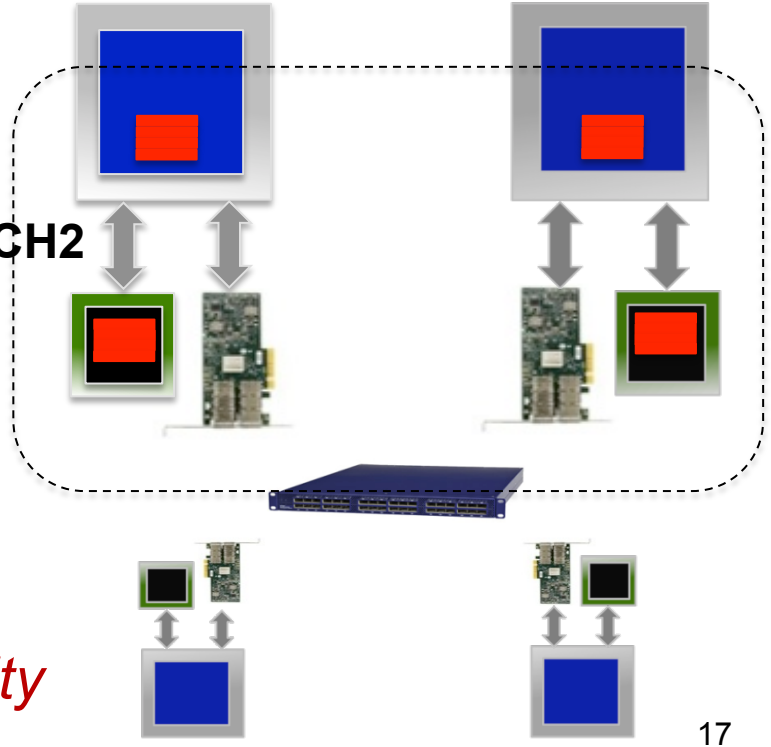
At Sender:

```
MPI_Send(s_device, size, …);
```

At Receiver:

```
MPI_Recv(r_device, size, …);
```

inside MVAPICH2

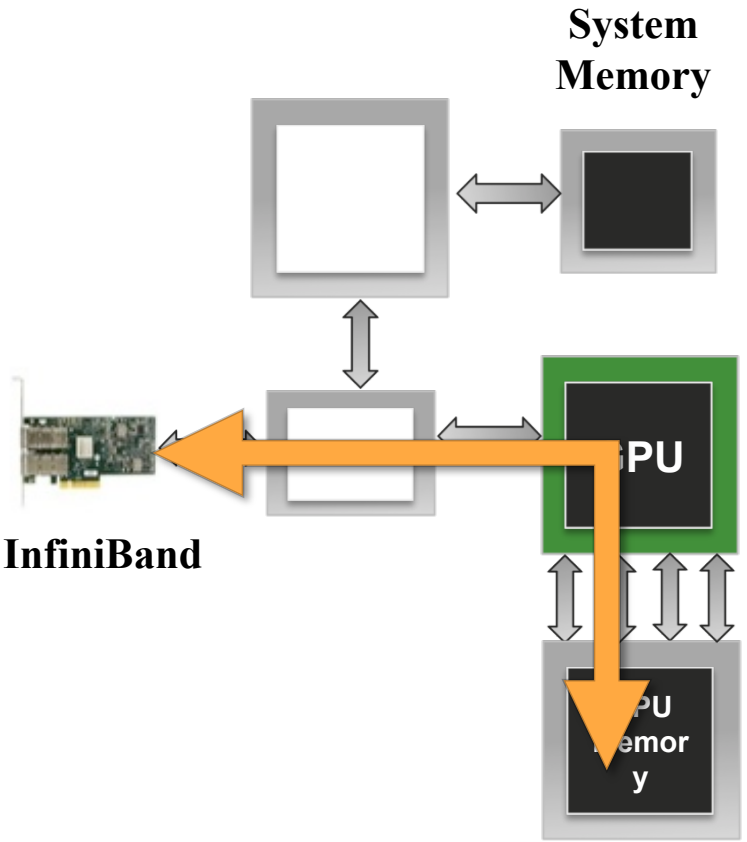


- *High Performance and High Productivity*

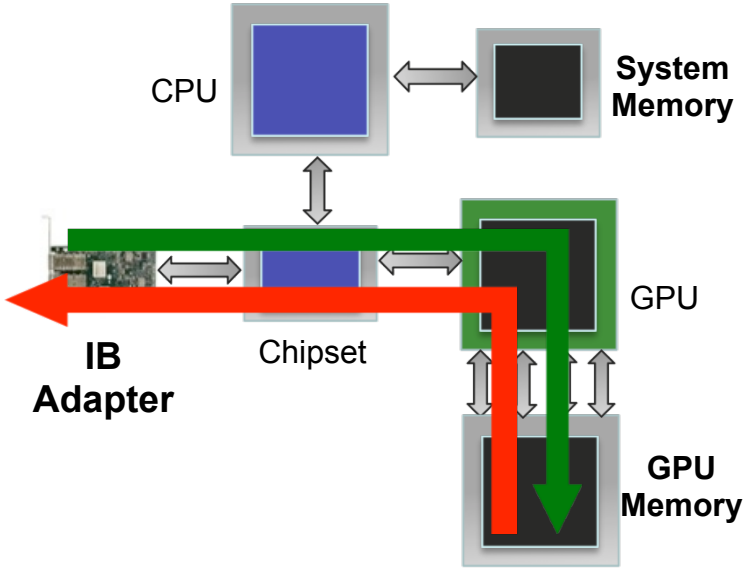
Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions

- Network adapter can directly read/ write data from/to GPU device memory
- Avoids copies through the host
- Fastest possible communication between GPU and IB HCA
- Allows for better asynchronous communication
- OFED with GDR support is under development by Mellanox and NVIDIA



- MVAPICH2 design extended to use GPUDirect RDMA
- Hybrid design using GPU-Direct RDMA
  - GPUDirect RDMA and Host-based pipelining
  - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters



SNB E5-2670

P2P write: 5.2 GB/s
P2P read: < 1.0 GB/s

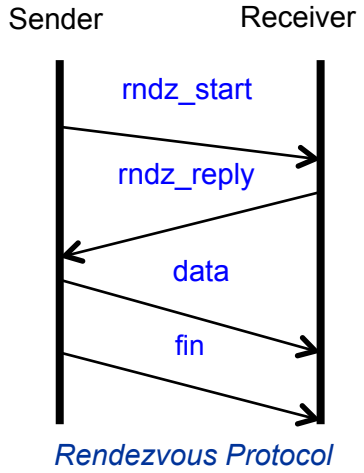
IVB E5-2680V2

P2P write: 6.4 GB/s
P2P read: 3.5 GB/s

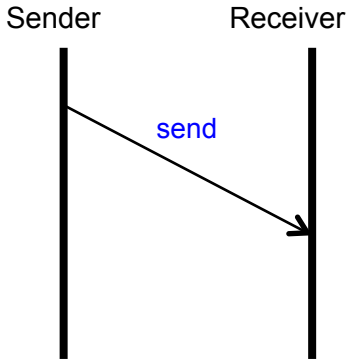
- MVAPICH2-2.1a with GDR support can be downloaded from https://mvapich.cse.ohio-state.edu/download/#mv2gdr/
- System software requirements
  - Mellanox OFED 2.1 or later
  - NVIDIA Driver 331.20 or later
  - NVIDIA CUDA Toolkit 6.0 or later
  - Plugin for GPUDirect RDMA
  - http://www.mellanox.com/page/products_dyn?product_family=116
  - Strongly Recommended : use the new GDRCOPY module from NVIDIA
    - https://github.com/NVIDIA/gdrcopy
- Has optimized designs for point-to-point communication using GDR
- Contact MVAPICH help list with any questions related to the package
  - mvapich-help@cse.ohio-state.edu

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - More Optimizations
- MPI and OpenACC
- Conclusions

- Current MPI design using GPUDirect RDMA uses Rendezvous protocol
  - Has higher latency for small messages
- Can eager protocol be supported to improve performance for small messages?
- Two schemes proposed and used
  - Loopback (using network adapter to copy data)
  - Fastcopy/GDRCOPY (using CPU to copy data)

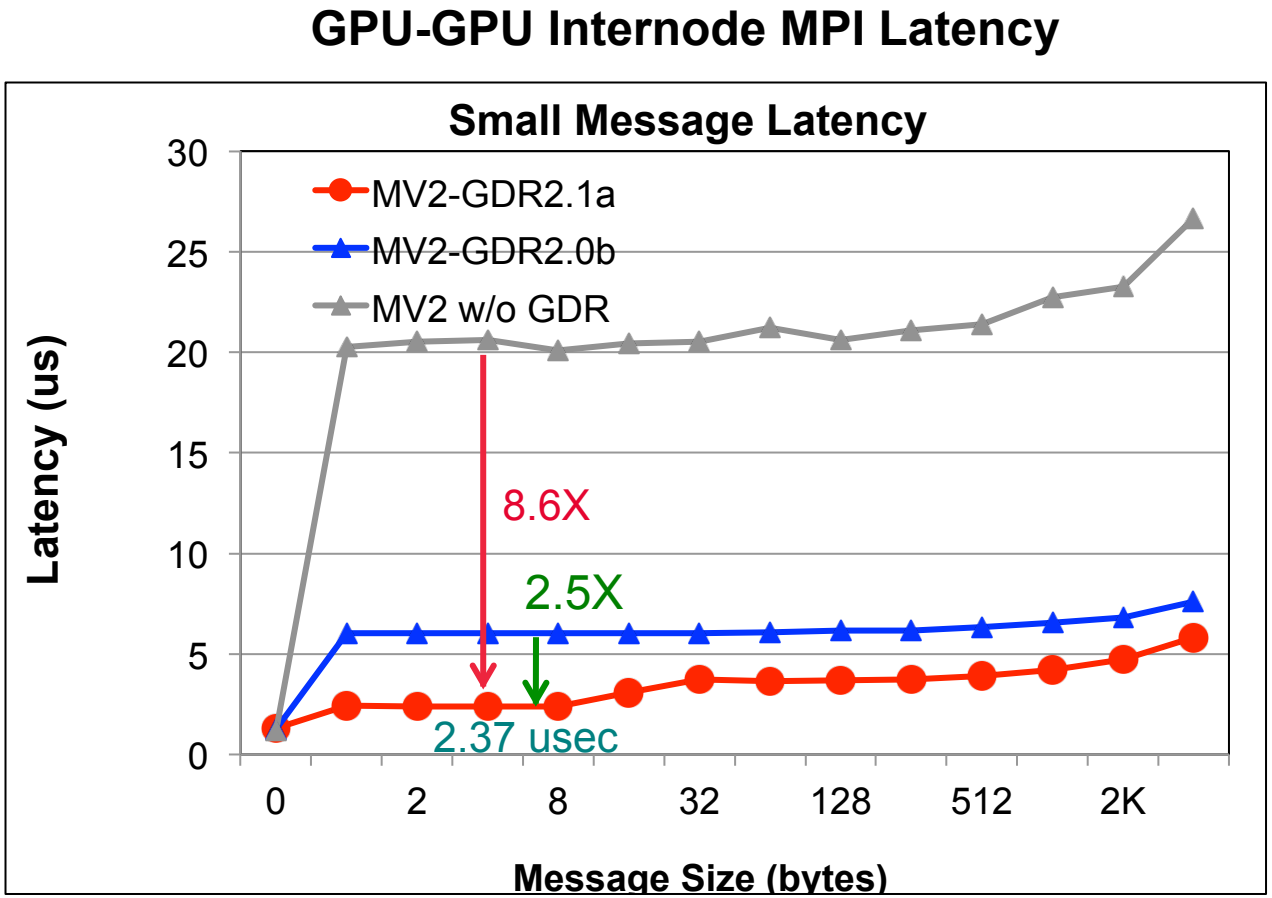


Rendezvous Protocol



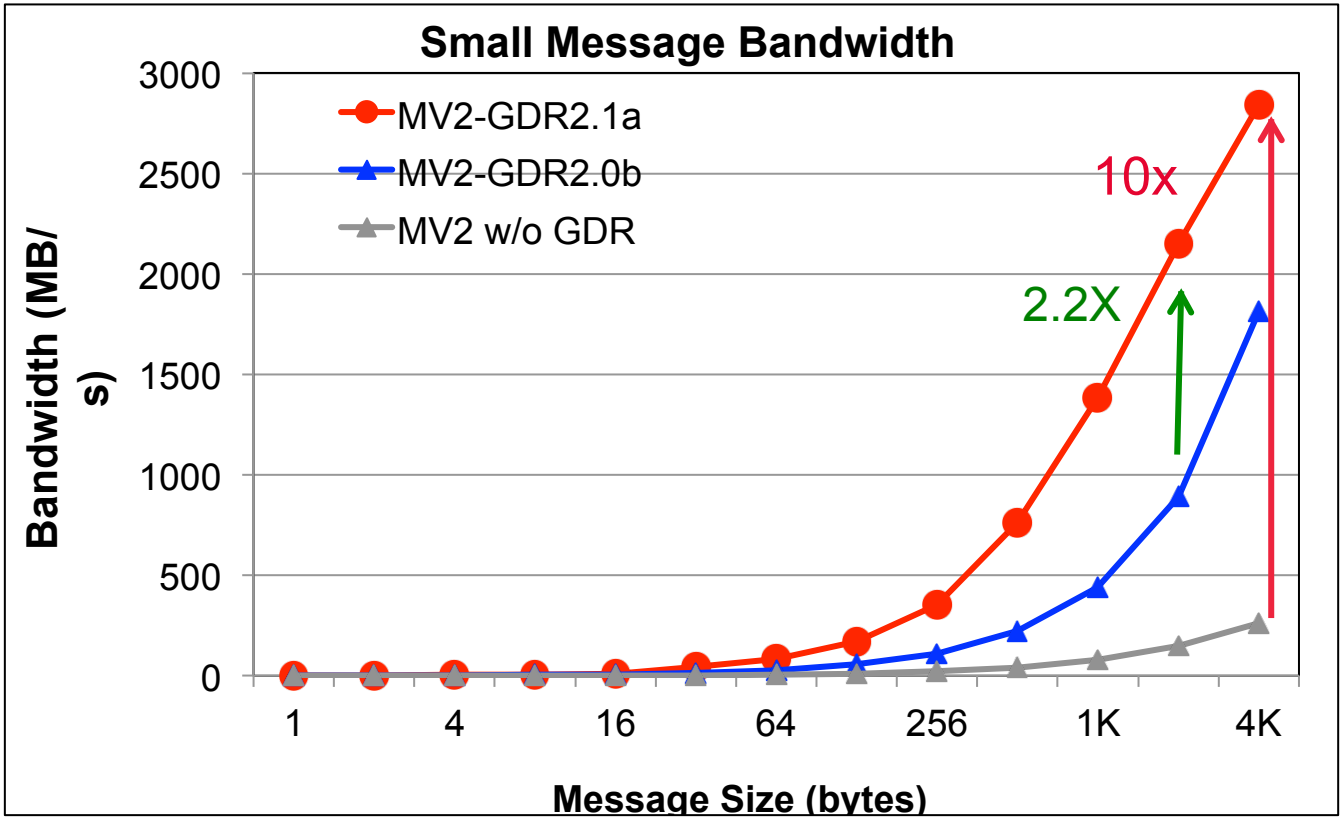
Eager Protocol

R. Shi, S. Potluri, K. Hamidouche M. Li, J. Perkins D. Rossetti and D. K. Panda, Designing Efficient Small Message Transfer Mechanism for Inter-node MPI Communication on InfiniBand GPU Clusters IEEE International Conference on High Performance Computing (HiPC'2014).



Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA

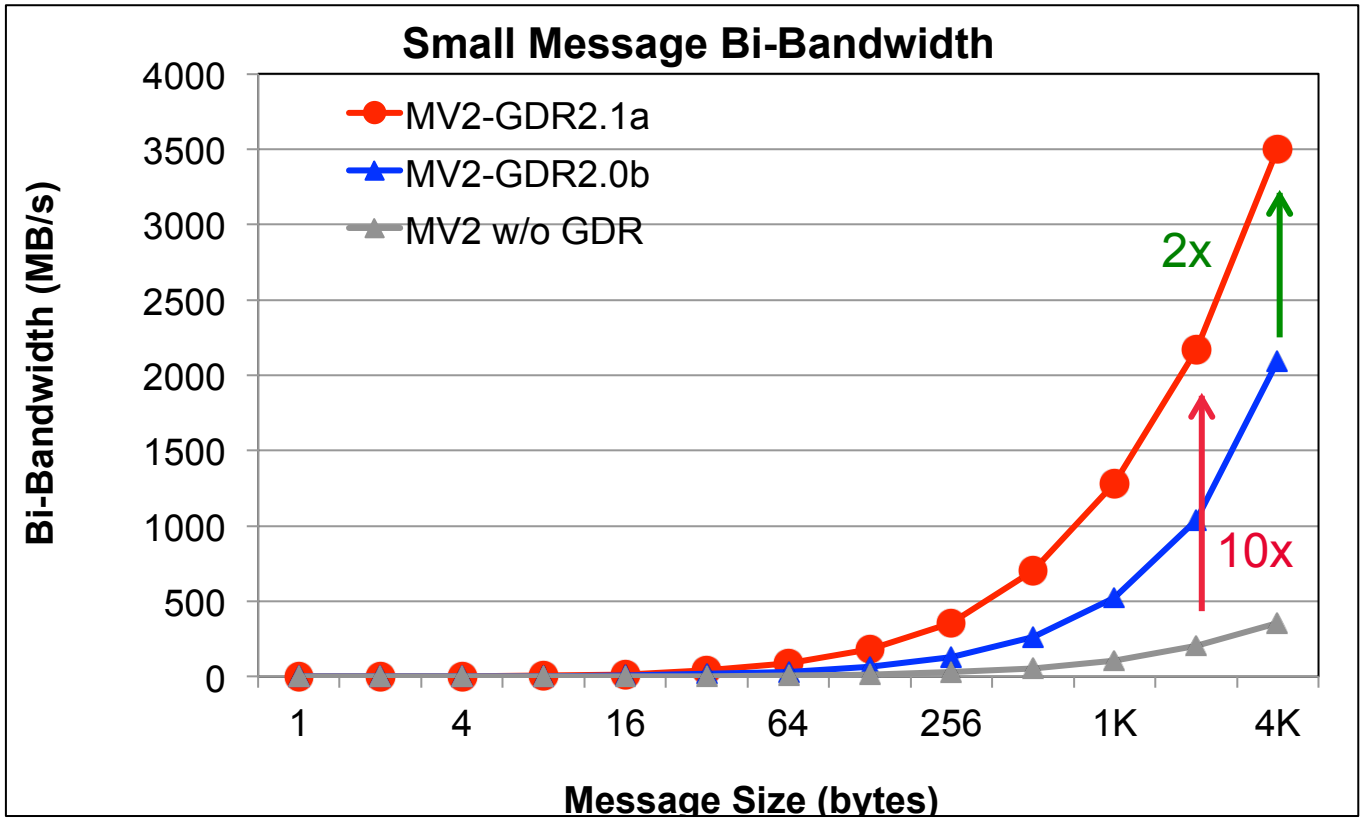
GPU-GPU Internode MPI Uni-Directional Bandwidth



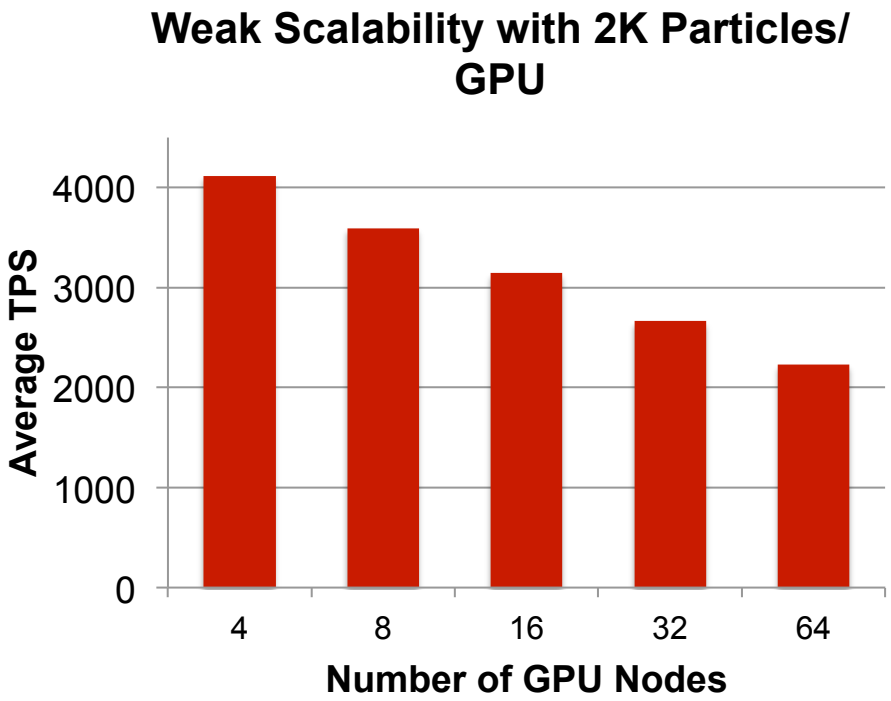
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA

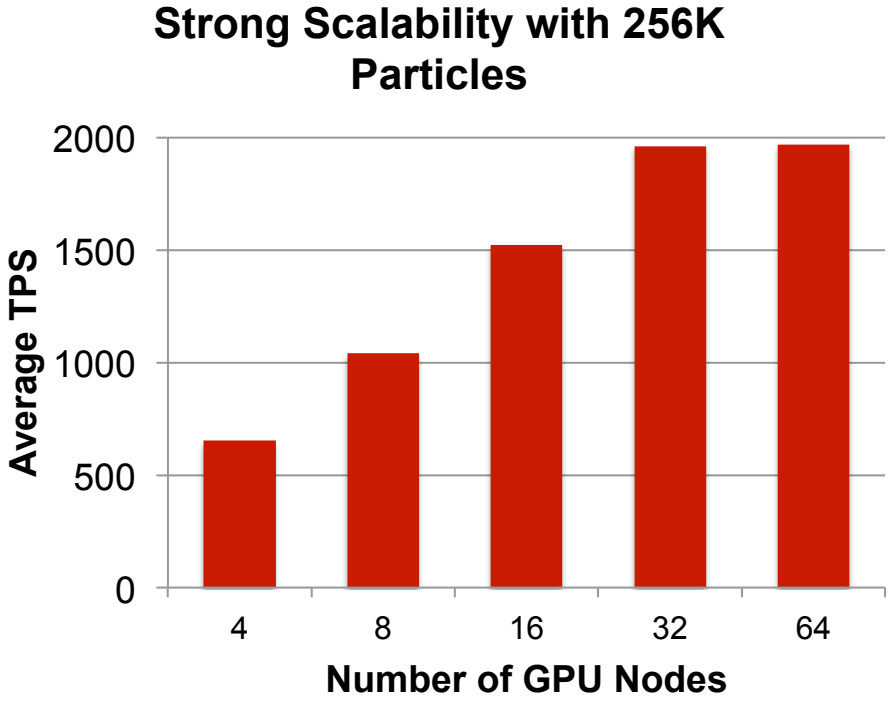
Performance of MVAPICH2 with GPU-Direct-RDMA: Bi-Bandwidth

GPU-GPU Internode MPI Bi-directional Bandwidth

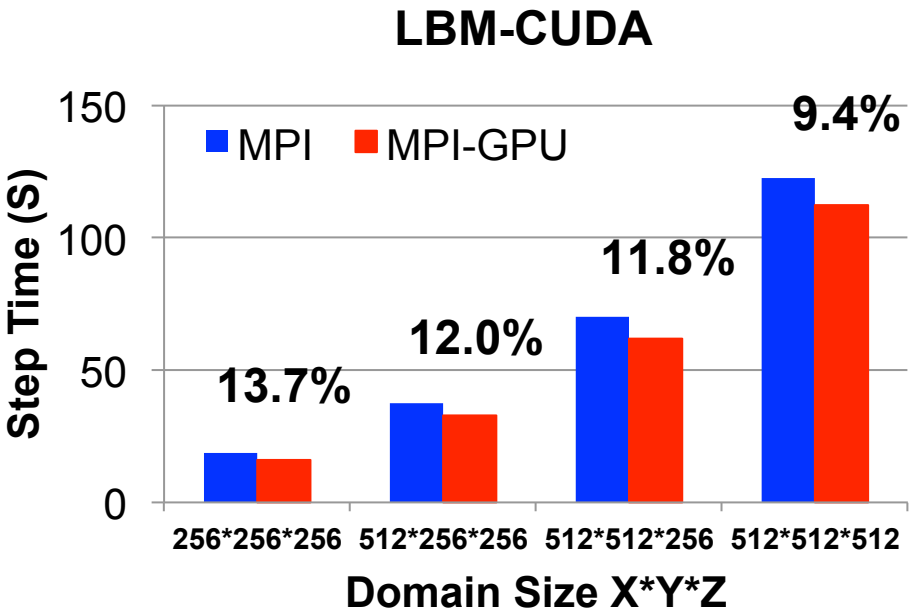


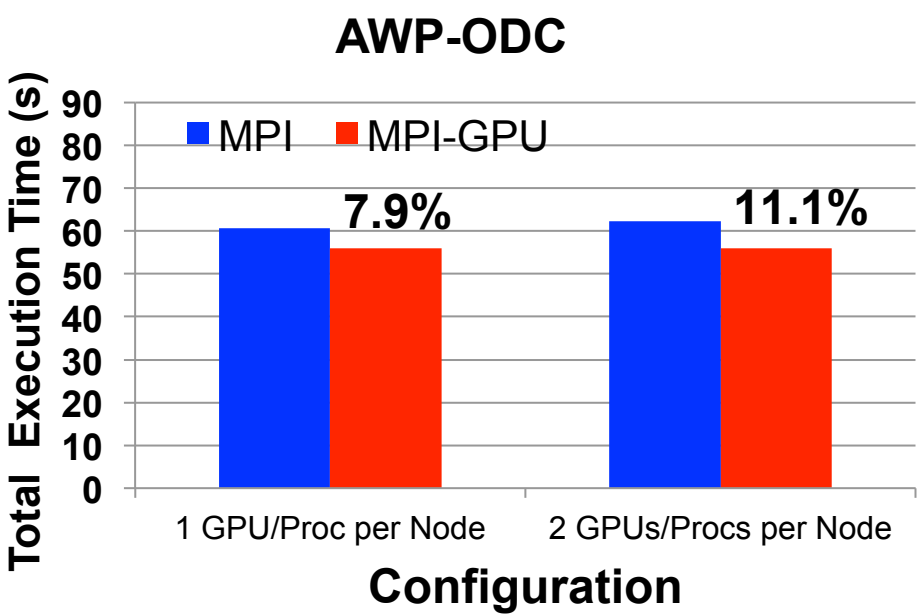
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA





- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384





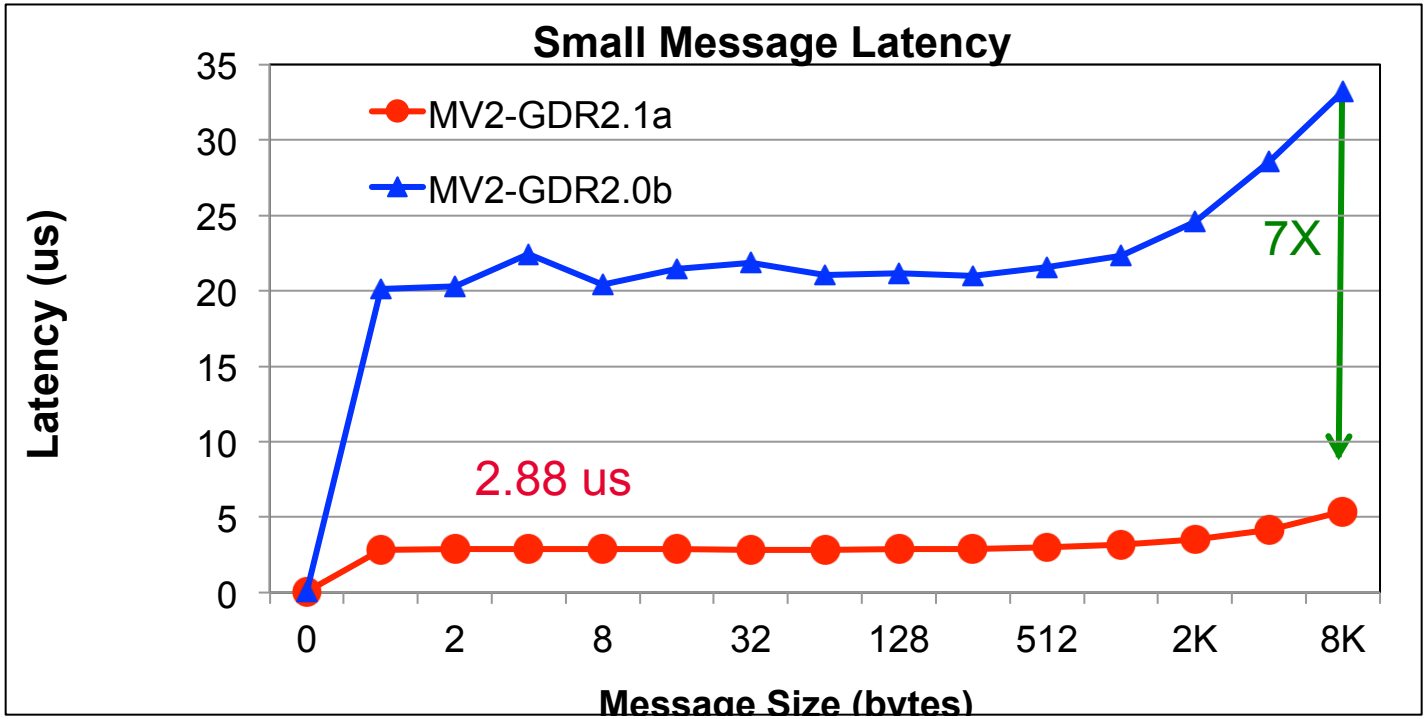
- LBM-CUDA (Courtesy: Carlos Rosale, TACC)
  - Lattice Boltzmann Method for multiphase flows with large density ratios
  - One process/GPU per node, 16 nodes
- AWP-ODC (Courtesy: Yifeng Cui, SDSC)
  - Seismic modeling code, Gordon Bell Prize finalist at SC 2010
  - 128x256x512 data grid per process, 8 nodes
- Oakley cluster at OSC: two hex-core Intel Westmere processors, two NVIDIA Tesla M2070, one Mellanox IB QDR

Outline

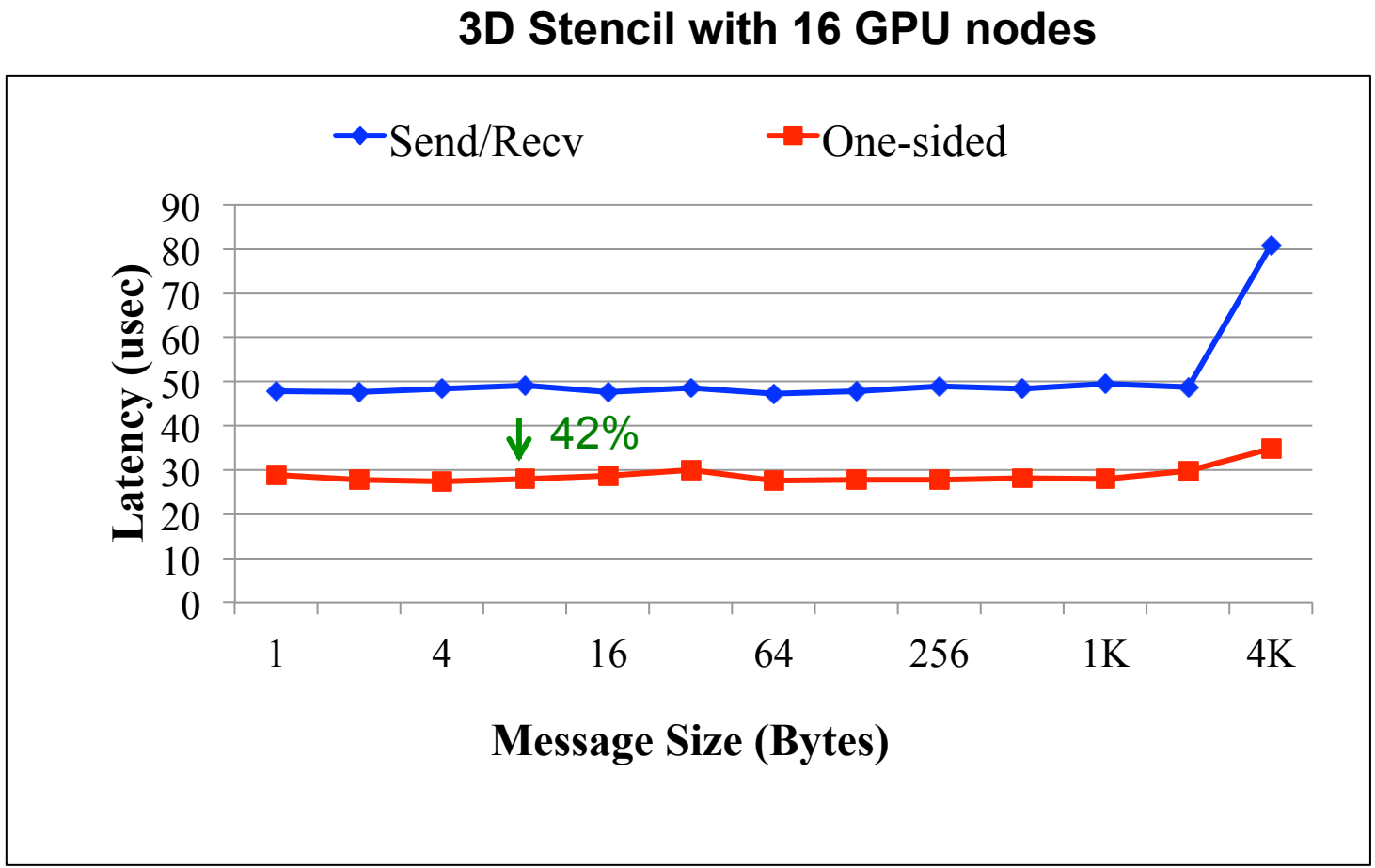
- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions

GPU-GPU Internode MPI Put latency (RMA put operation Device to Device)

MPI-3 RMA provides flexible synchronization and completion primitives

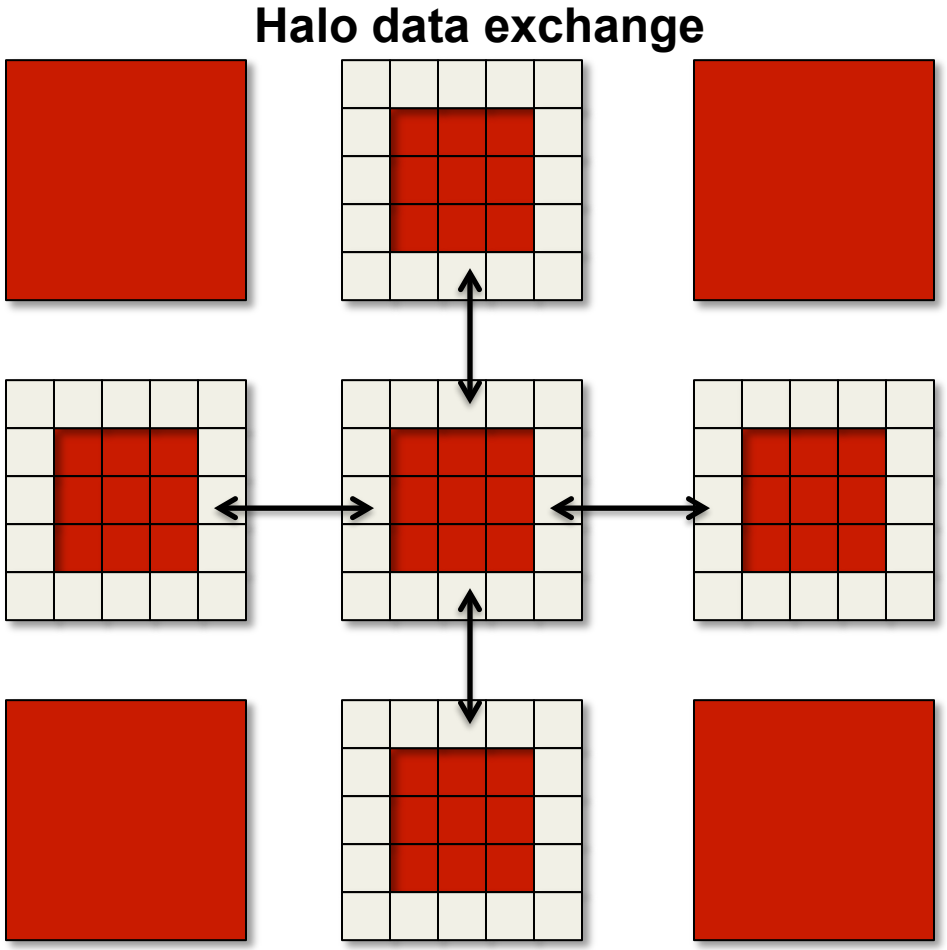


Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA



Based on MVAPICH2-2.0b
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.1 with GPUDirect-RDMA Plugin

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions



- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions
- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration

- Datatypes support in MPI
  - Operate on customized datatypes to improve productivity
  - Enable MPI library to optimize non-contiguous data

At Sender:

```
MPI_Type_vector (n_blocks, n_elements, stride, old_type, &new_type);
MPI_Type_commit(&new_type);
…
MPI_Send(s_buf, size, new_type, dest, tag, MPI_COMM_WORLD);
```

- Inside MVAPICH2
  - Use datatype specific CUDA Kernels to pack data in chunks
  - Efficiently move data between nodes using RDMA
  - Optimizes datatypes from GPU memories
  - Transparent to the user

H. Wang, S. Potluri, D. Bureddy, C. Rosales and D. K. Panda, GPU-aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation, IEEE Transactions on Parallel and Distributed Systems.

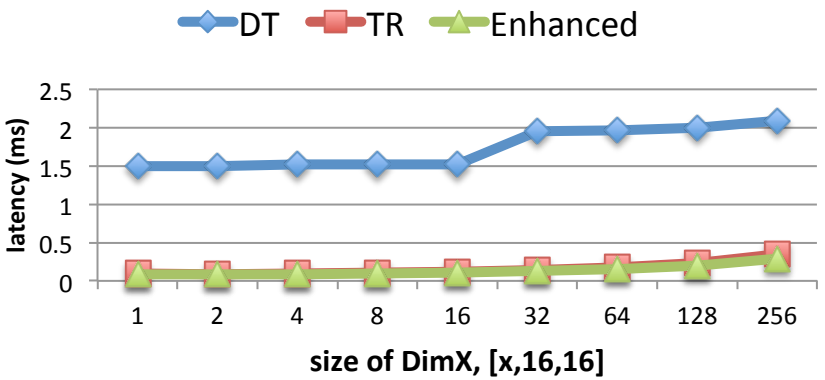
- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes
- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels
- Flexible set of parameters for users to tune kernels
  - Vector
    - MV2_CUDA_KERNEL_VECTOR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_VECTOR_YSIZE
  - Subarray
    - MV2_CUDA_KERNEL_SUBARR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_SUBARR_XDIM
    - MV2_CUDA_KERNEL_SUBARR_YDIM
    - MV2_CUDA_KERNEL_SUBARR_ZDIM
  - Indexed_block
    - MV2_CUDA_KERNEL_IDXBLK_XDIM

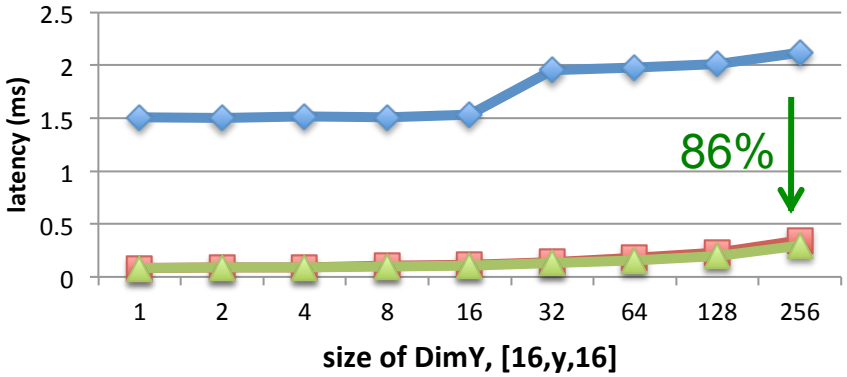
R. Shi, X. Lu, S. Potluri, K. Hamidouche, J. Zhang and D. K. Panda, HAND: A Hybrid Approach to Accelerate Non-contiguous Data Movement using MPI Datatypes on GPU Clusters International Conference on Parallel Processing (ICPP'14).

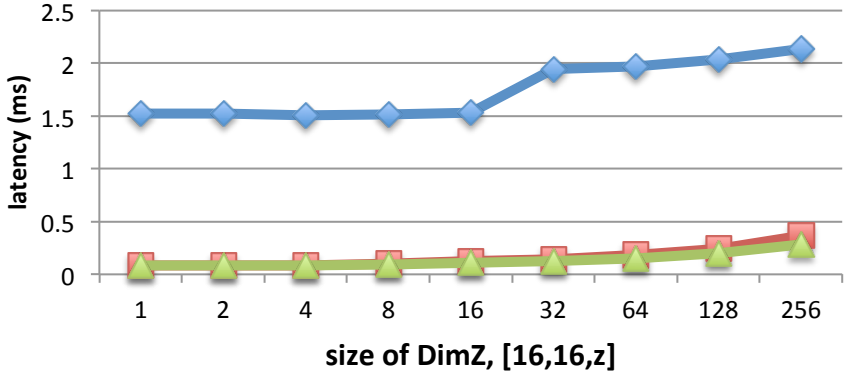
Performance of Stencil3D (3D subarray)

Stencil3D communication kernel on 2 GPUs with various X, Y, Z dimensions using MPI_Isend/Irecv

- DT: Direct Transfer, TR: Targeted Kernel
- Optimized design gains up to 15%, 15% and 22% compared to TR, and more than 86% compared to DT on X, Y and Z respectively

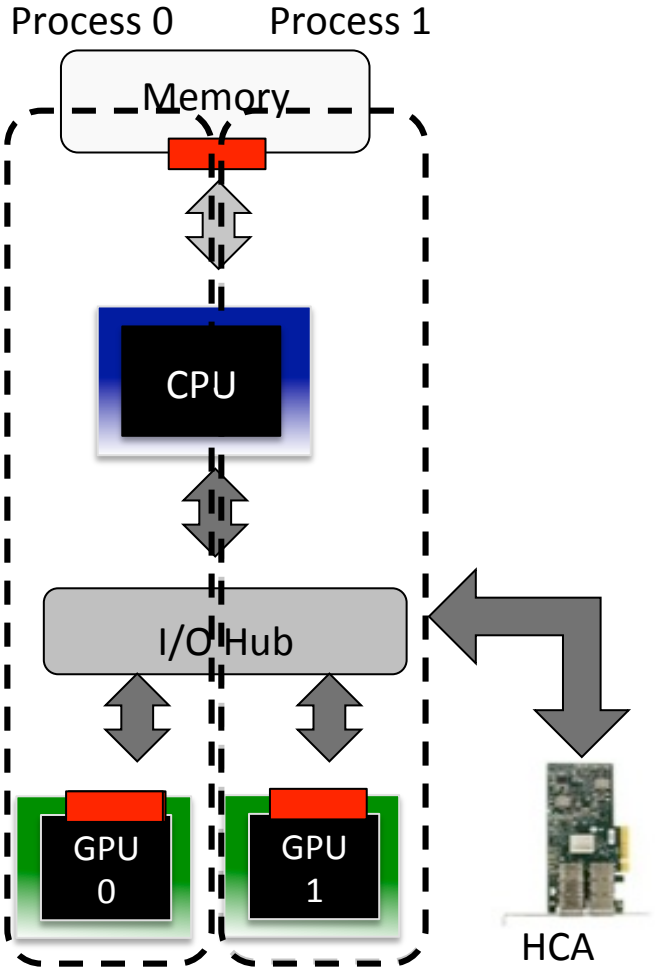




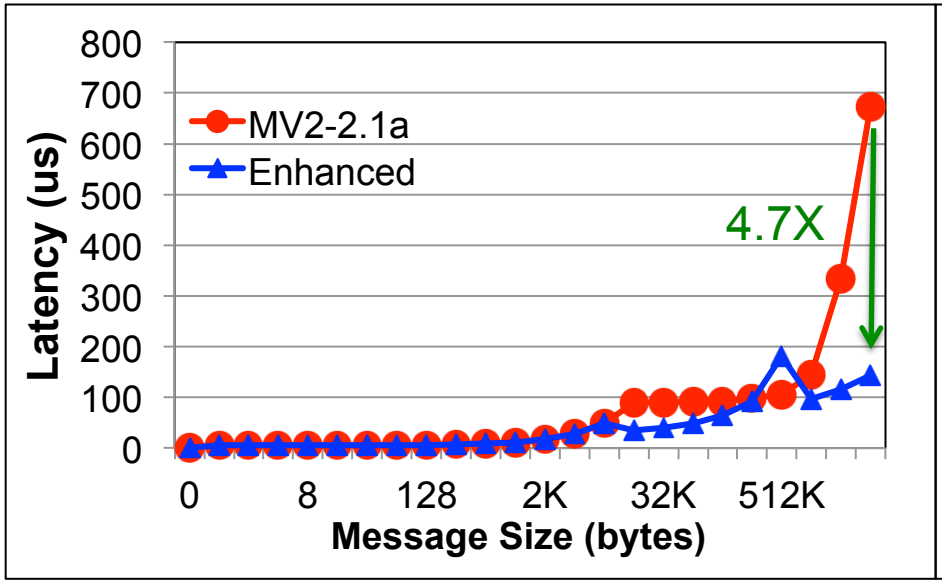


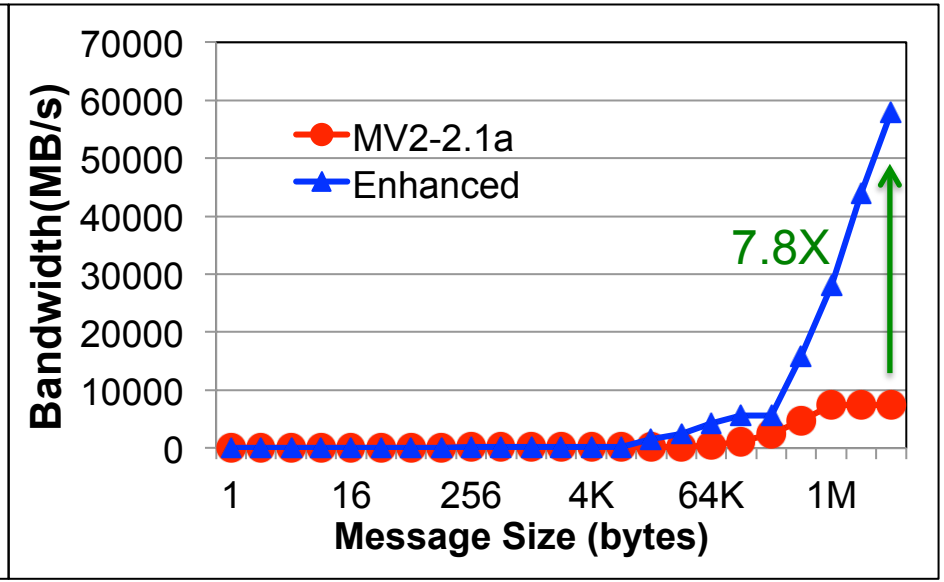
Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions



- 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 and later
  - Inter-Process Communication (IPC)
  - Host bypass
  - Handled by a DMA Engine
  - Low latency and Asynchronous
  - Requires creation, exchange and mapping of memory handles
  - Overhead





- Two Processes sharing the same K80 GPUs
- Proposed designs achieve 4.7X improvement in latency
- 7.8X improvement is delivered for Bandwidth
- Will be available with next releases of MVAPICH2-GDR

- MVAPICH2 1.8, 1.9 and other MPI libraries require device selection before MPI_Init
- This restriction has been removed with MVAPICH2 2.0 and later
- OSU micro-benchmarks still select device before MPI_Init for backward compatibility
- Uses node-level rank information exposed by launchers
- We provide a script which exports this node rank information to be used by the benchmark
  - Can be modified to work with different MPI libraries without modifying the benchmarks themselves
  - Sample script provided with OMB : "get_local_rank"

    export LOCAL_RANK=$MV2_COMM_WORLD_LOCAL_RANK

- Automatic detection of GPUs and IB placement:
  - Avoid the inter-sockets **QPI bottlenecks**
  - Selection of the nearest HCA and binding the process to the associate CPU set
  - Dynamic threshold **selection** between GDR and host-based transfers

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions

- OpenACC is gaining popularity
- Several sessions during GTC
- A set of compiler directives (#pragma)
- Offload specific loops or parallelizable sections in code onto accelerators

```
#pragma acc region
{
    for(i = 0; i < size; i++) {
        A[i] = B[i] + C[i];
    }
}
```

- Routines to allocate/free memory on accelerators

```
buffer = acc_malloc(MYBUFSIZE);
acc_free(buffer);
```

- Supported for C, C++ and Fortran
- Huge list of modifiers – **copy, copyout, private, independent, etc..**

- acc_malloc to allocate device memory
  - No changes to MPI calls
  - MVAPICH2 detects the device pointer and optimizes data movement
  - Delivers the same performance as with CUDA

```
A = acc_malloc(sizeof(int) * N);

……

#pragma acc parallel loop deviceptr(A) . . .
//compute for loop

MPI_Send (A, N, MPI_INT, 0, 1, MPI_COMM_WORLD);



……
acc_free(A);
```

- acc_deviceptr to get device pointer (in OpenACC 2.0)
  - Enables MPI communication from memory allocated by compiler when it is available in OpenACC 2.0 implementations
  - MVAPICH2 will detect the device pointer and optimize communication
  - Delivers the same performance as with CUDA

```
A = malloc(sizeof(int) * N);

……

#pragma acc data copyin(A) . . .
{

#pragma acc parallel loop . . .
//compute for loop

MPI_Send(acc_deviceptr(A), N, MPI_INT, 0, 1, MPI_COMM_WORLD);

}
……
free(A);
```

- OSU Micro-benchmarks (OMB) are widely used to compare performance of different MPI stacks and networks
- Enhancements to measure performance of MPI communication from GPU memory
  - Point-to-point: Latency, Bandwidth and Bi-directional Bandwidth
  - Collectives: Latency test for all collectives
- Support for CUDA and OpenACC
- Flexible selection of data movement between CPU(H) and GPU(D): D->D, D->H and H->D
- Available from http://mvapich.cse.ohio-state.edu/benchmarks
- Available in an integrated manner with MVAPICH2 stack

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU: CUDA-Aware MPI
- MVAPICH2-GPU with GPUDirect-RDMA (MVAPICH2-GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
  - Optimizations (multi-GPU, device selection, topology detection)
- MPI and OpenACC
- Conclusions

- MVAPICH2 optimizes MPI communication on InfiniBand clusters with GPUs
- Close partnership with NVIDIA during the last four years
- Provides optimized designs for point-to-point two-sided, one-sided communications, and datatype processing
- Takes advantage of CUDA features like IPC, GPUDirect RDMA and GDRCOPY
- Delivers
  - High performance
  - High productivity

With support for latest NVIDIA GPUs and InfiniBand Adapters

Acknowledgments

Dr. Davide Rossetti

Filippo Spiga and Stuart Rankin,
HPCS, University of Cambridge
(Wilkes Cluster)

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- A. Awan (Ph.D.)
- A. Bhat (M.S.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- N. Islam (Ph.D.)

Past Students

- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- E. Mancini
- S. Marcarelli
- J. Vienne

Past Post-Docs

- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo

Current Senior Research Associates

- K. Hamidouche
- X. Lu
- H. Subramoni

Current Post-Doc

- J. Lin

Current Programmer

- J. Perkins

Current Research Specialist

- M. Arnold

- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Research Scientist

- S. Sur

Past Programmers

- D. Bureddy

Contact

panda@cse.ohio-state.edu

hamidouc@cse.ohio-state.edu

MVAPICH

Network Based Computing Laboratory

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

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