MVAPICH2: A High Performance MPI Library for NVIDIA GPU Clusters with InfiniBand

Presentation at GTC 2013

by

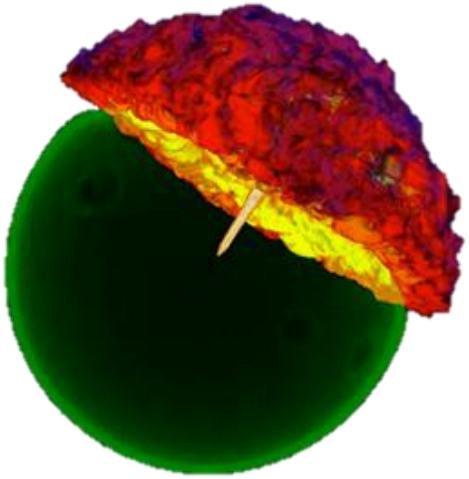
Dhabaleswar K. (DK) Panda

The Ohio State University

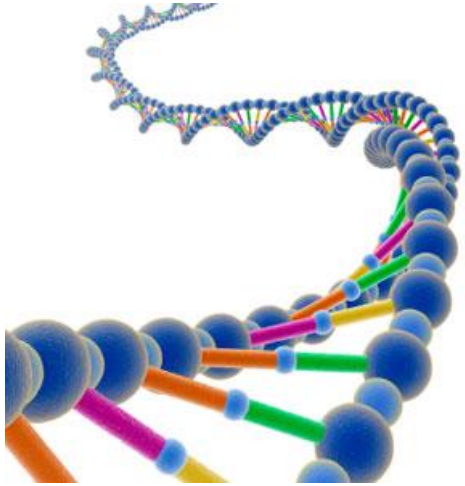
E-mail: panda@cse.ohio-state.edu

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

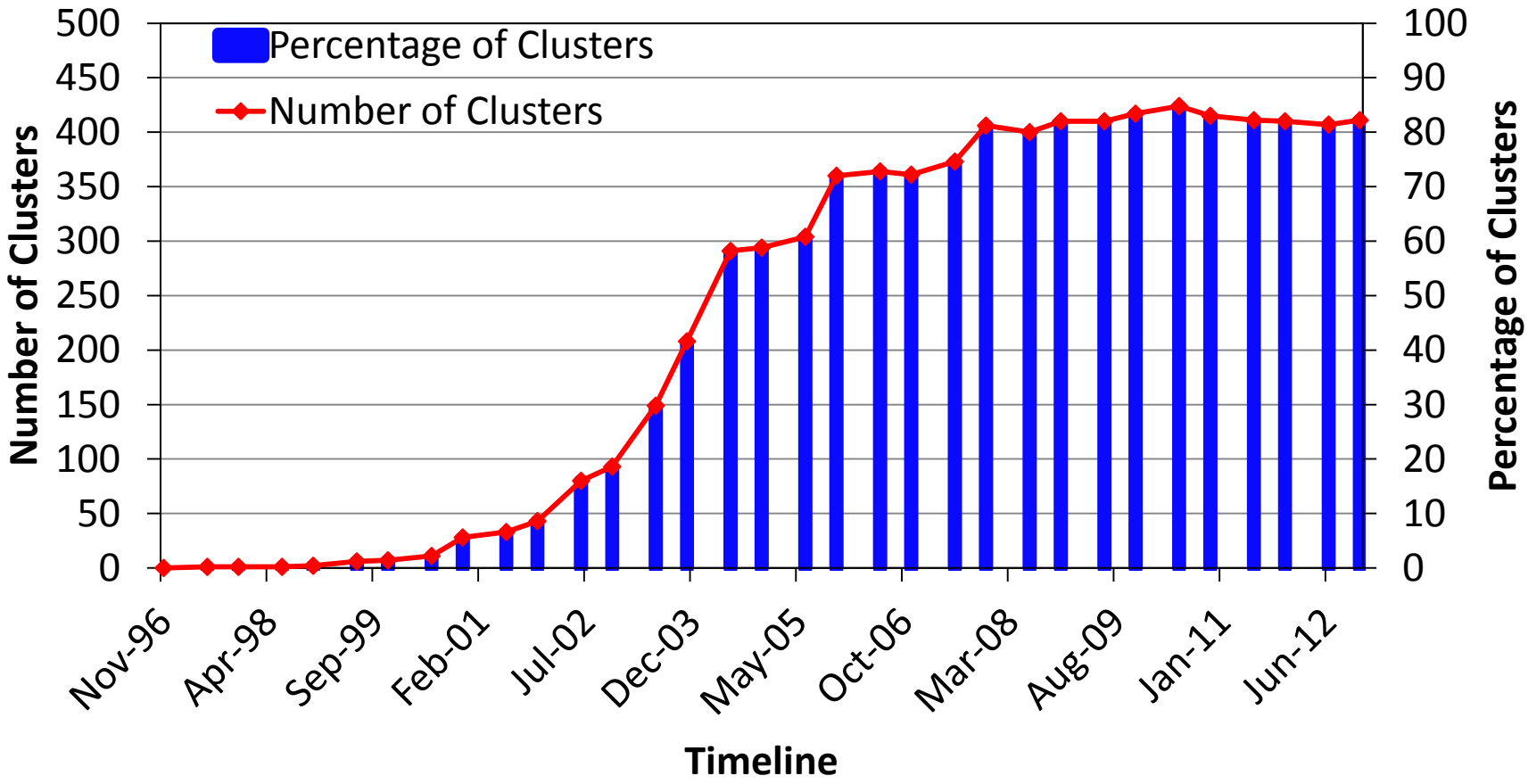
Current and Next Generation HPC Systems and Applications



- Growth of High Performance Computing (HPC)
  - Growth in processor performance
    - Chip density doubles every 18 months
  - Growth in commodity networking
    - Increase in speed/features + reducing cost
  - Growth in accelerators (NVIDIA GPUs)



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



Large-scale InfiniBand Installations

- 224 IB Clusters (44.8%) in the November 2012 Top500 list (http://www.top500.org)
- Installations in the Top 20 (9 systems), Two use NVIDIA GPUs

147, 456 cores (Super MUC) in Germany (6th)	125,980 cores (Pleiades) at NASA/Ames (14th)
204,900 cores (Stampede) at TACC (7th)	70,560 cores (Helios) at Japan/IFERC (15th)
77,184 cores (Curie thin nodes) at France/CEA (11th)	**73,278 cores (Tsubame 2.0) at Japan/GSIC (17th)**
120, 640 cores (Nebulae) at China/NSCS (12th)	138,368 cores (Tera-100) at France/CEA (20th)
72,288 cores (Yellowstone) at NCAR (13th)	

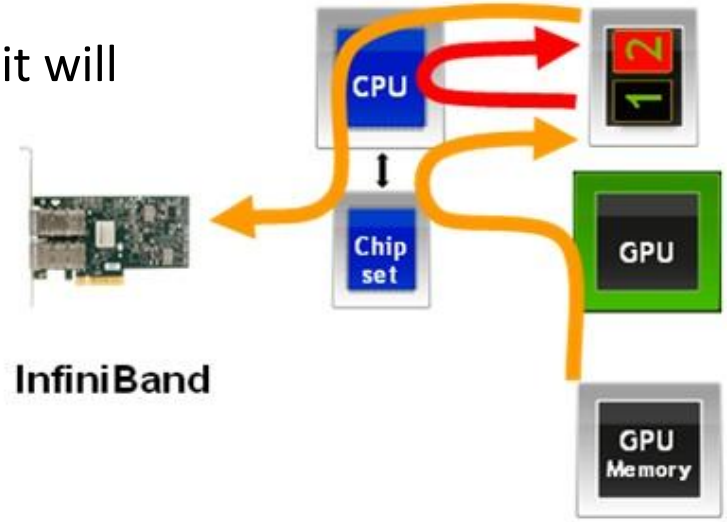
- 54 of the InfiniBand clusters (in TOP500) house accelerators/co-processors and 42 of them have NVIDIA GPUs

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

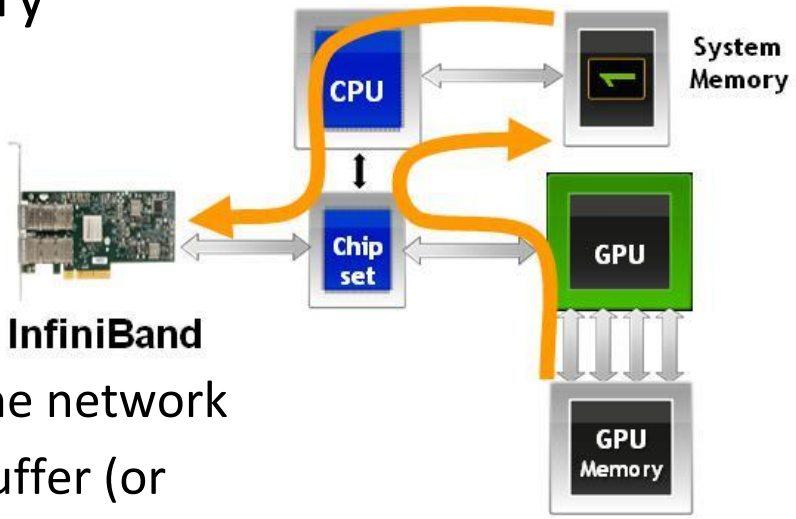
InfiniBand + GPU systems (Past)

- 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 buffer for each device and copy data



GPU-Direct

- 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*



Sample Code - Without MPI integration

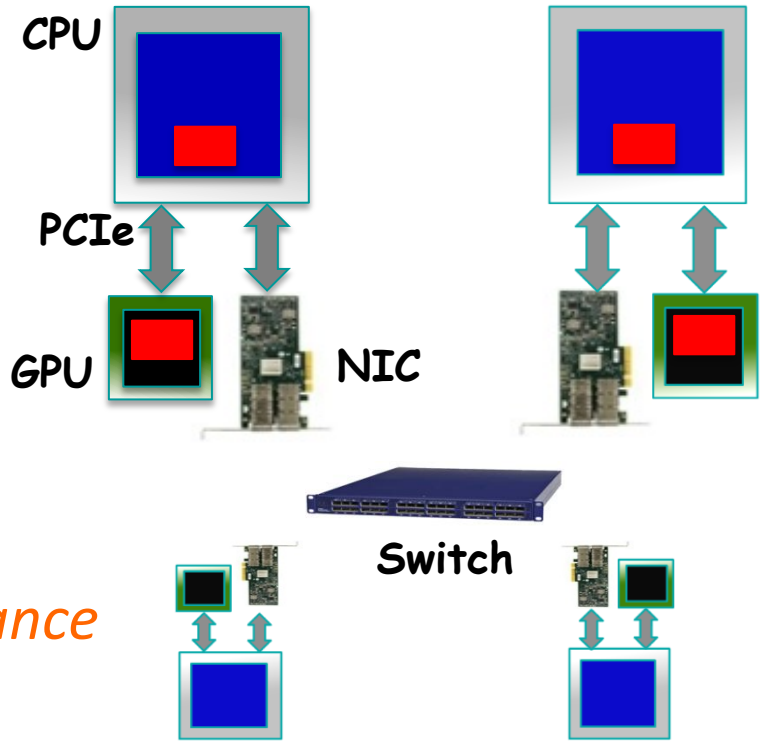
- Naïve implementation with standard MPI and CUDA

At Sender:

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

At Receiver:

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



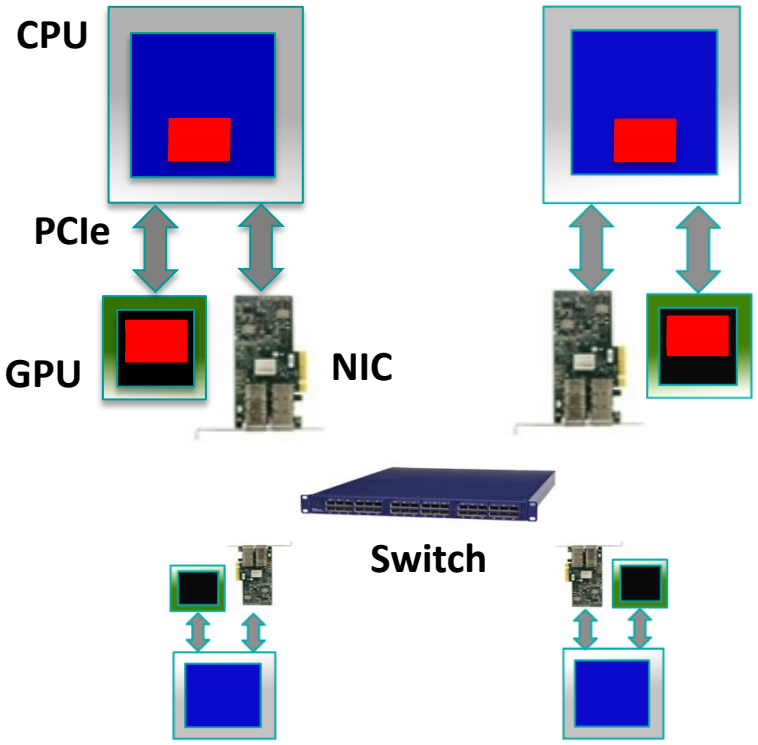
- *High Productivity and Poor Performance*

Sample Code – User Optimized Code

- 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*

Can this be done within MPI Library?

- 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 was incorporated in MVAPICH2 to support this functionality**

MVAPICH2/MVAPICH2-X Software

- 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
  - Used by more than 2,000 organizations (HPC Centers, Industry and Universities) in 70 countries
  - More than 160,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 7th ranked 204,900-core cluster (Stampede) at TACC
    - 14th ranked 125,980-core cluster (Pleiades) at NASA
    - 17th ranked 73,278-core cluster (Tsubame 2.0) at Tokyo Institute of Technology
    - 75th ranked 16,896-core cluster (Keenland) at GaTech
    - 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

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

Sample Code – MVAPICH2-GPU

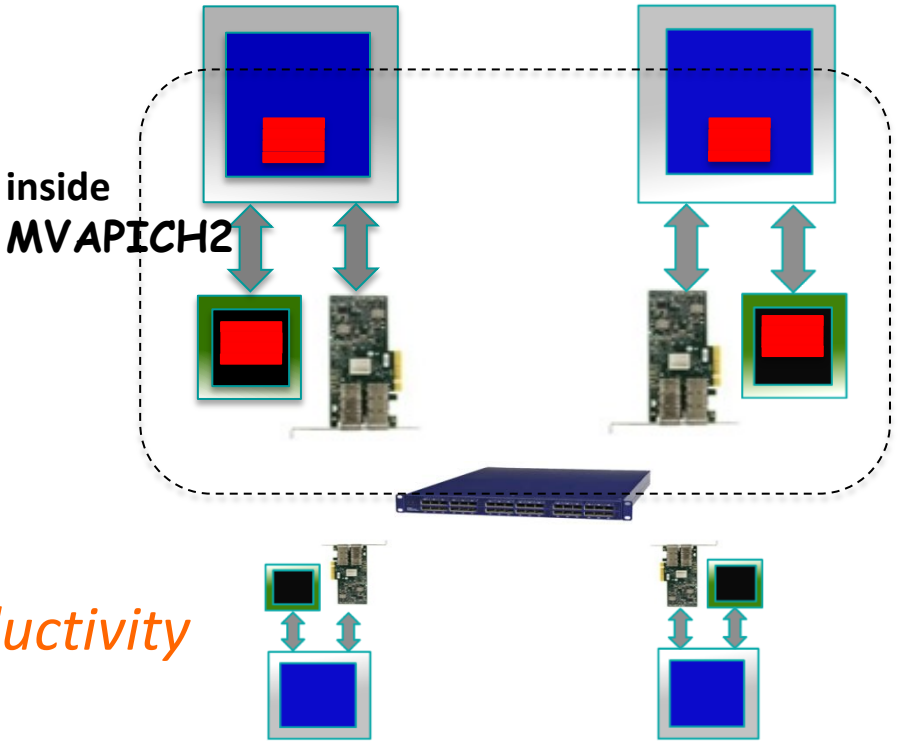
- MVAPICH2-GPU: standard MPI interfaces used
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_device, size, …);
```

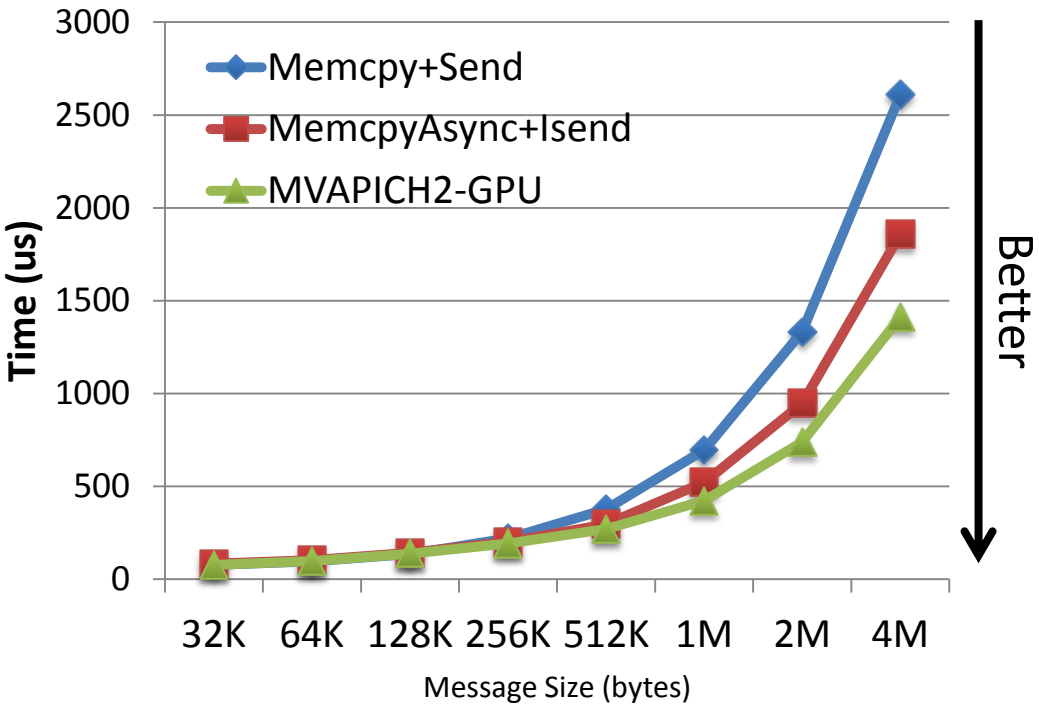
At Receiver:

```
MPI_Recv(r_device, size, …);
```



- *High Performance and High Productivity*

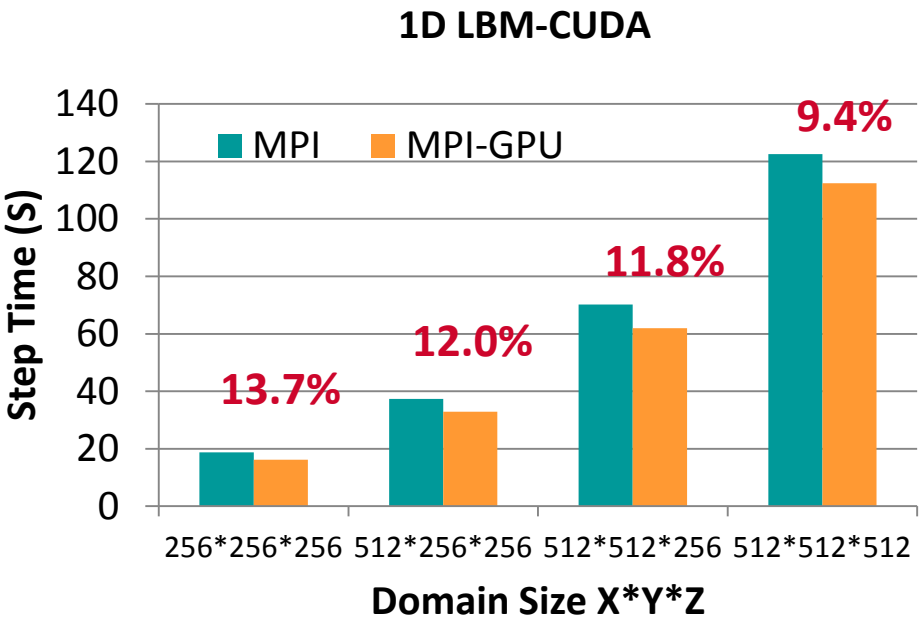
MPI Two-sided Communication

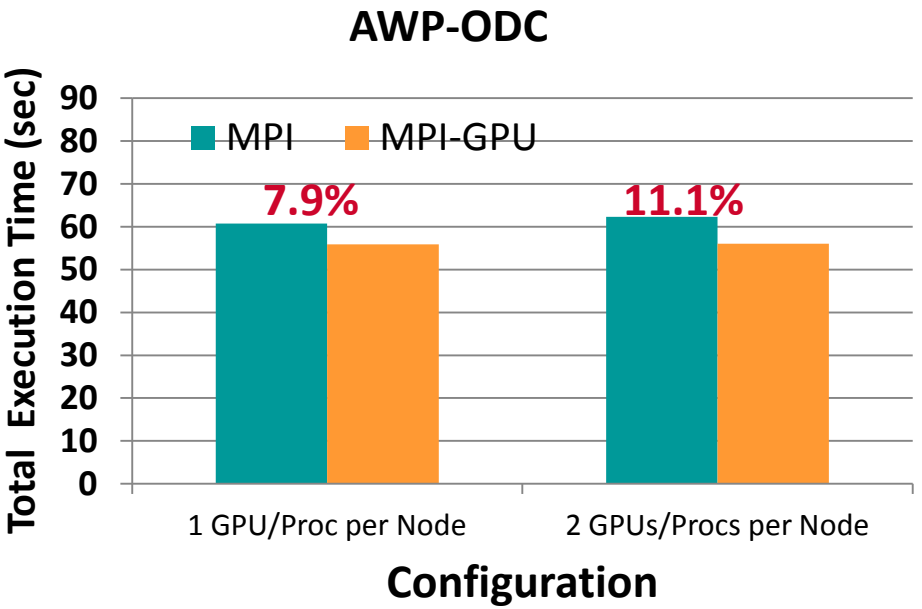


- 45% improvement compared with a naïve user-level implementation (Memcpy+Send), for 4MB messages
- 24% improvement compared with an advanced user-level implementation (MemcpyAsync+Isend), for 4MB messages

H. Wang, S. Potluri, M. Luo, A. Singh, S. Sur and D. K. Panda, MVAPICH2-GPU: Optimized GPU to GPU Communication for InfiniBand Clusters, ISC '11

Application-Level Evaluation (LBM and AWP-ODC)





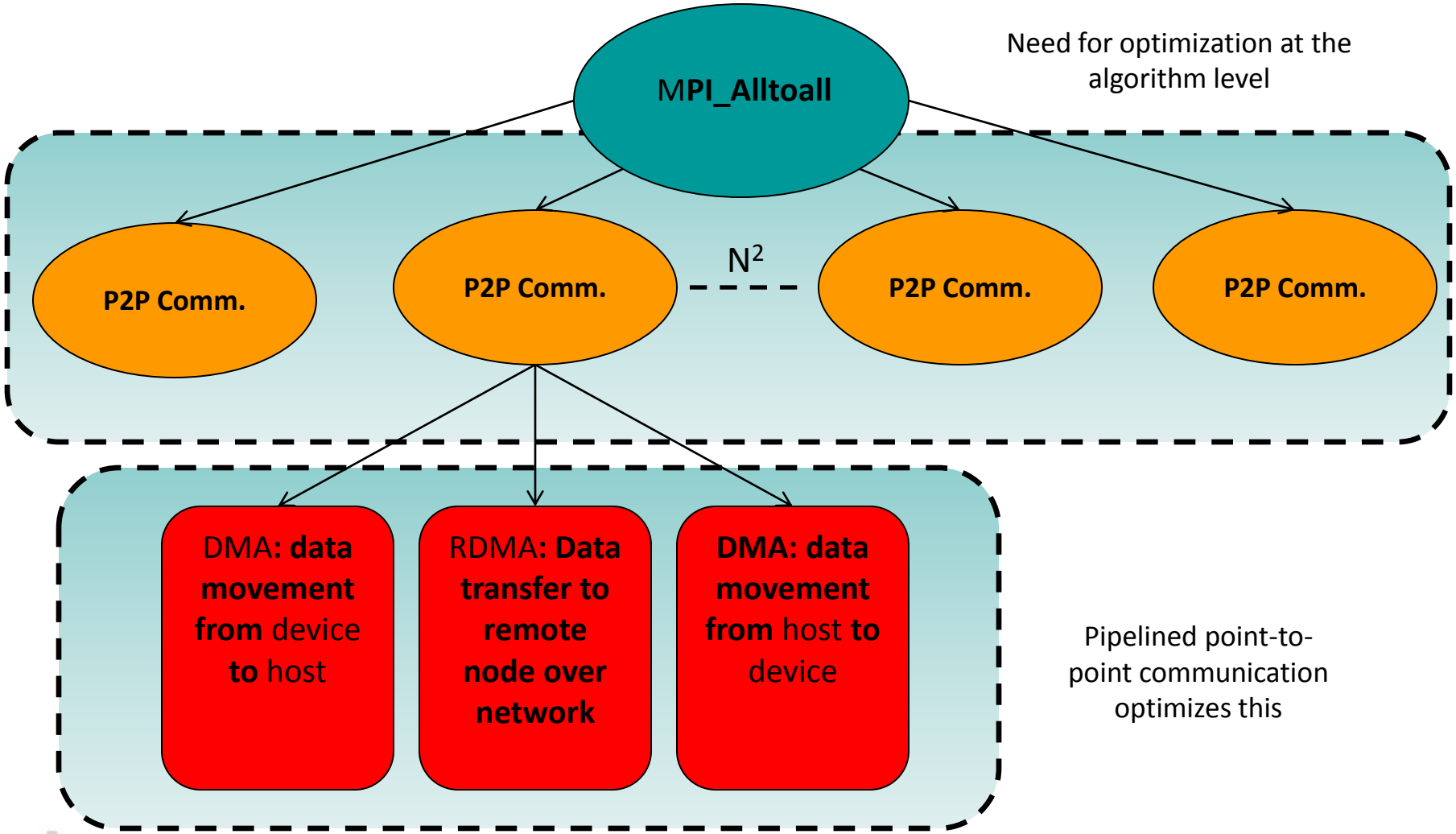
- **LBM-CUDA** (Courtesy: Carlos Rosale, TACC)
  - Lattice Boltzmann Method for multiphase flows with large density ratios
  - **1D LBM-CUDA:** one process/GPU per node, 16 nodes, 4 groups data grid
- **AWP-ODC** (Courtesy: Yifeng Cui, SDSC)
  - A seismic modeling code, Gordon Bell Prize finalist at 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 MT26428 adapter and 48 GB of main memory

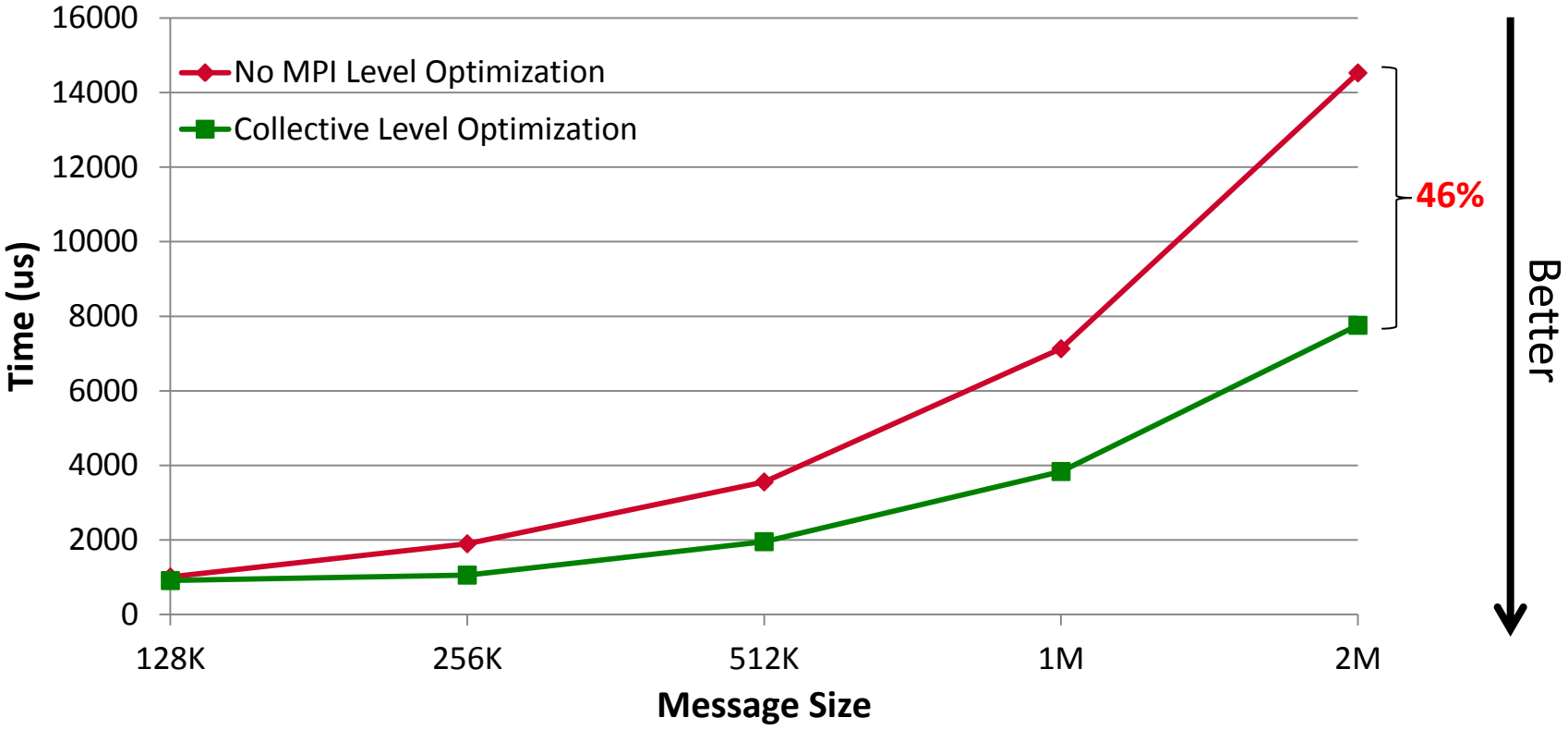
Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

Optimizing Collective Communication



Alltoall Latency Performance (Large Messages)



- 8 node Westmere cluster with NVIDIA Tesla C2050 and IB QDR

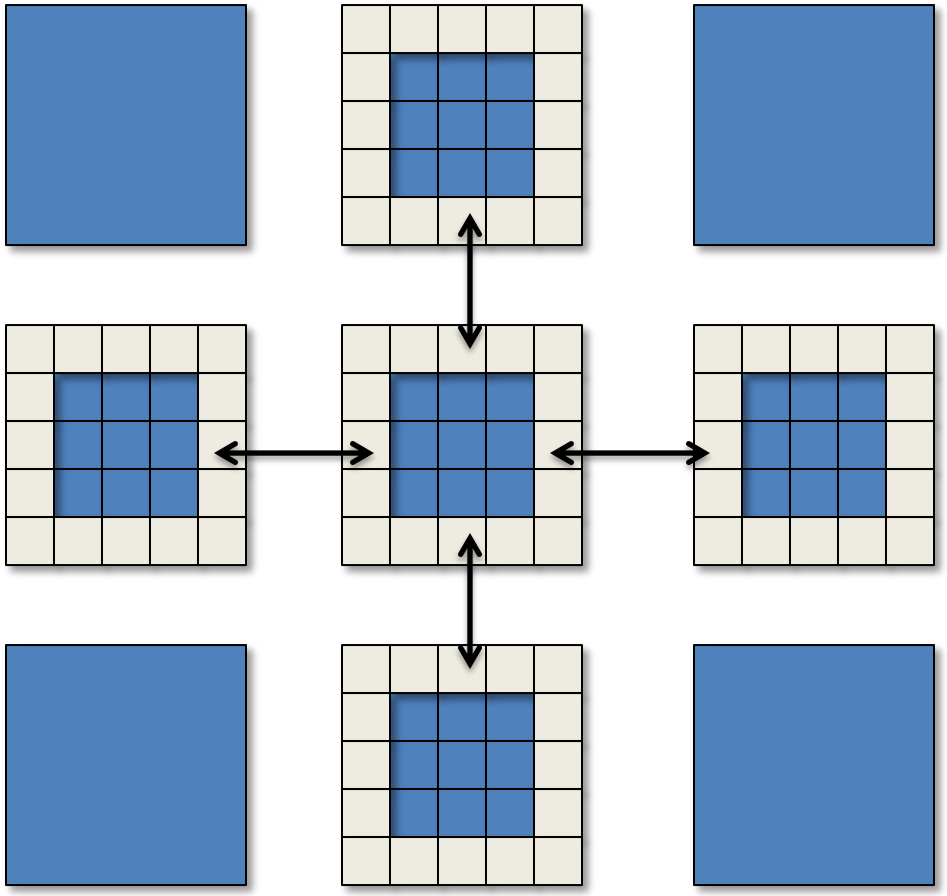
A. Singh, S. Potluri, H. Wang, K. Kandalla, S. Sur and D. K. Panda, MPI Alltoall Personalized Exchange on GPGPU Clusters: Design Alternatives and Benefits, Workshop on Parallel Programming on Accelerator Clusters (PPAC '11), held in conjunction with Cluster '11, Sept. 2011

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype Processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

Non-contiguous Data Exchange

Halo data exchange



- 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

Datatype Support in MPI

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

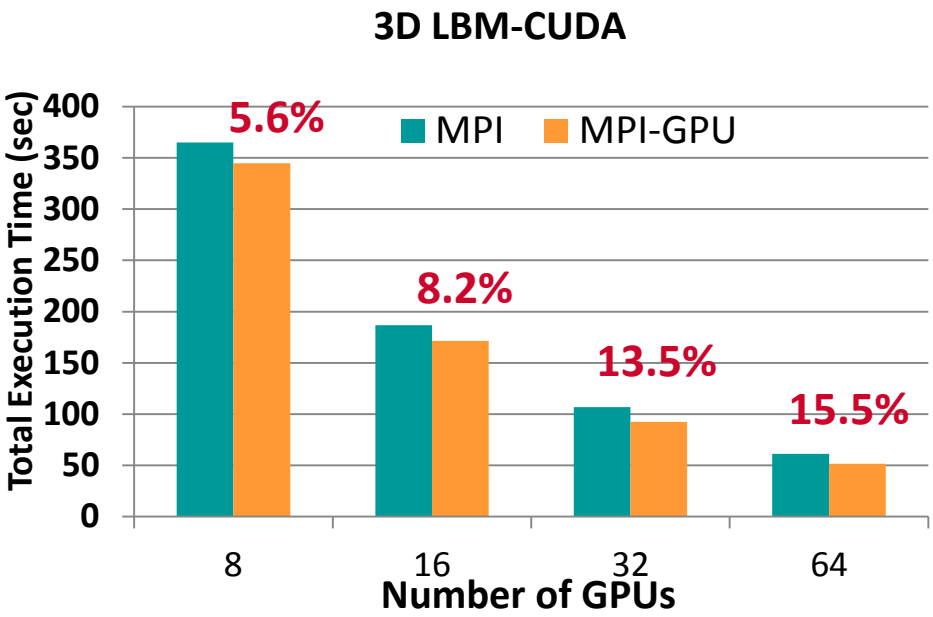
At Sender:

```
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);
```

- What will happen if the non-contiguous data is in the GPU device memory?
- Enhanced MVAPICH2
  - Use data-type specific CUDA Kernels to pack data in chunks
  - Pipeline pack/unpack, CUDA copies, and RDMA transfers

H. Wang, S. Potluri, M. Luo, A. Singh, X. Ouyang, S. Sur and D. K. Panda, Optimized Non-contiguous MPI Datatype Communication for GPU Clusters: Design, Implementation and Evaluation with MVAPICH2, IEEE Cluster '11, Sept. 2011

Application-Level Evaluation (LBMGPU-3D)



- LBM-CUDA (Courtesy: Carlos Rosale, TACC)
  - Lattice Boltzmann Method for multiphase flows with large density ratios
  - **3D LBM-CUDA: one process/GPU per node, 512x512x512 data grid, up to 64 nodes**

- Oakley cluster at OSC: two hex-core Intel Westmere processors, two NVIDIA Tesla M2070, one Mellanox IB QDR MT26428 adapter and 48 GB of main memory

MVAPICH2 1.8 and 1.9 Series

- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers

OSU MPI Micro-Benchmarks (OMB) 3.5 – 3.9 Releases

- A comprehensive suite of benchmarks to compare performance of different MPI stacks and networks
- Enhancements to measure MPI performance on GPU clusters
  - Latency, Bandwidth, Bi-directional Bandwidth
- Flexible selection of data movement between CPU(H) and GPU(D): D->D, D->H and H->D
- Extensions with OpenACC is added in 3.9 Release
- Available from http://mvapich.cse.ohio-state.edu/benchmarks
- Available in an integrated manner with MVAPICH2 stack

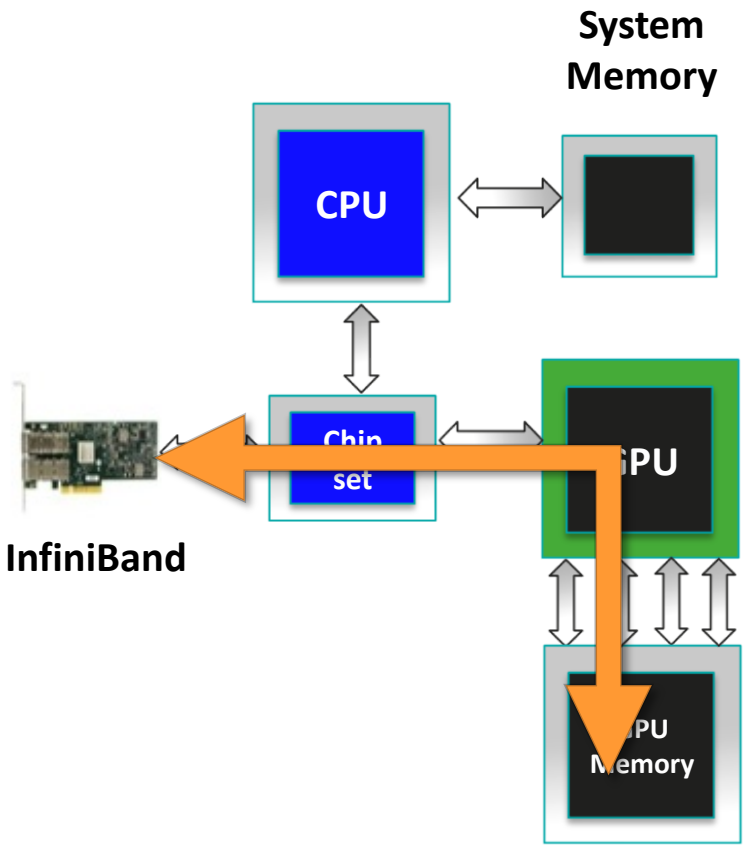
• D. Bureddy, H. Wang, A. Venkatesh, S. Potluri and D. K. Panda, OMB-GPU: A Micro-benchmark suite for Evaluating MPI Libraries on GPU Clusters, EuroMPI 2012, September 2012.

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype Processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

GPU-Direct RDMA with CUDA 5.0

- Fastest possible communication between GPU and other PCI-E devices
- Network adapter can directly read/write data from/to GPU device memory
- Avoids copies through the host
- Allows for better asynchronous communication



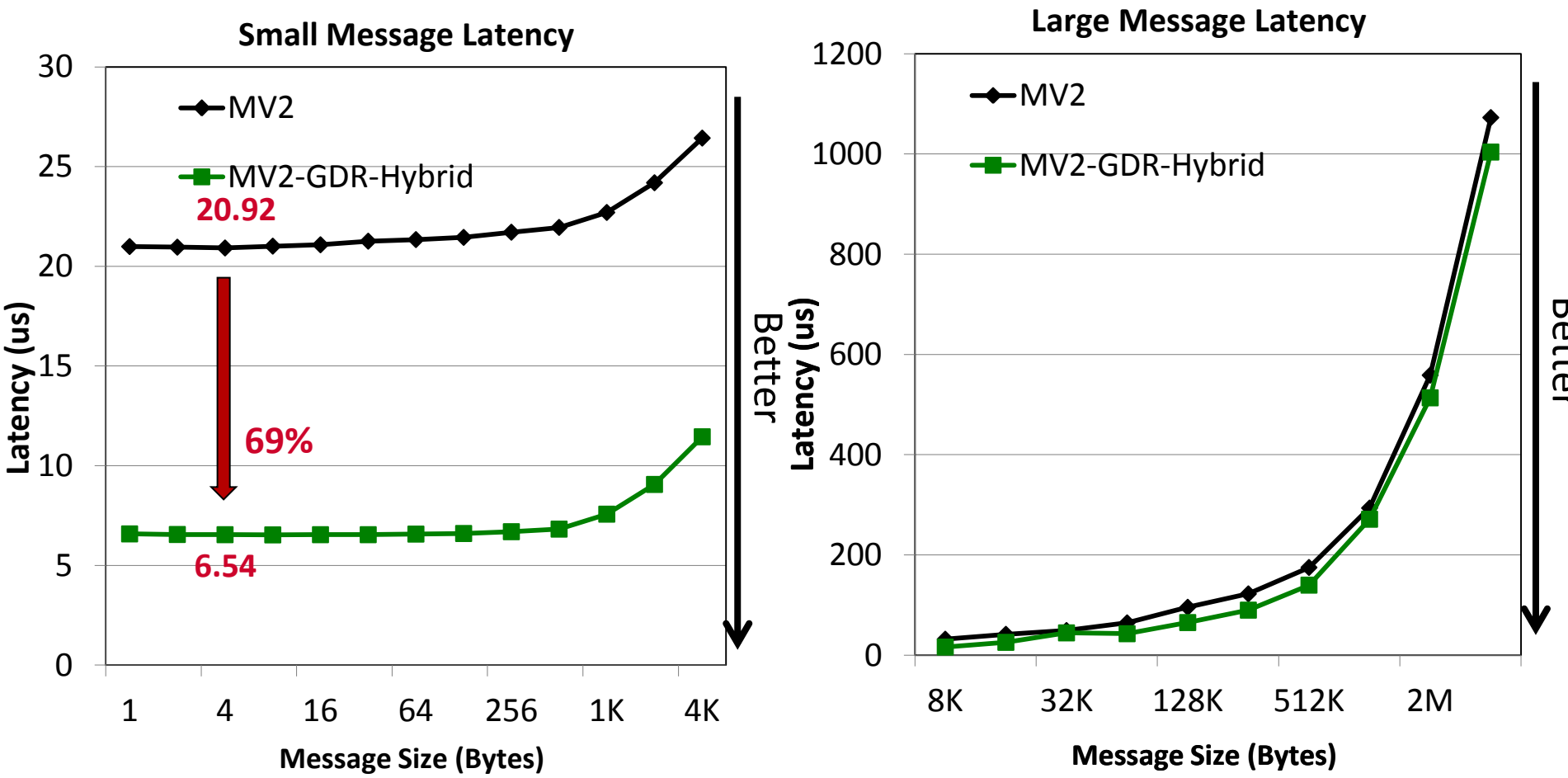
Initial Design of MVAPICH2 with GPU-Direct-RDMA

- Preliminary driver for GPU-Direct is under work by NVIDIA and Mellanox
- OSU has done an initial design of MVAPICH2 with the latest GPU-Direct-RDMA Driver

Preliminary Performance Evaluation of OSU-MVAPICH2 with GPU-Direct-RDMA

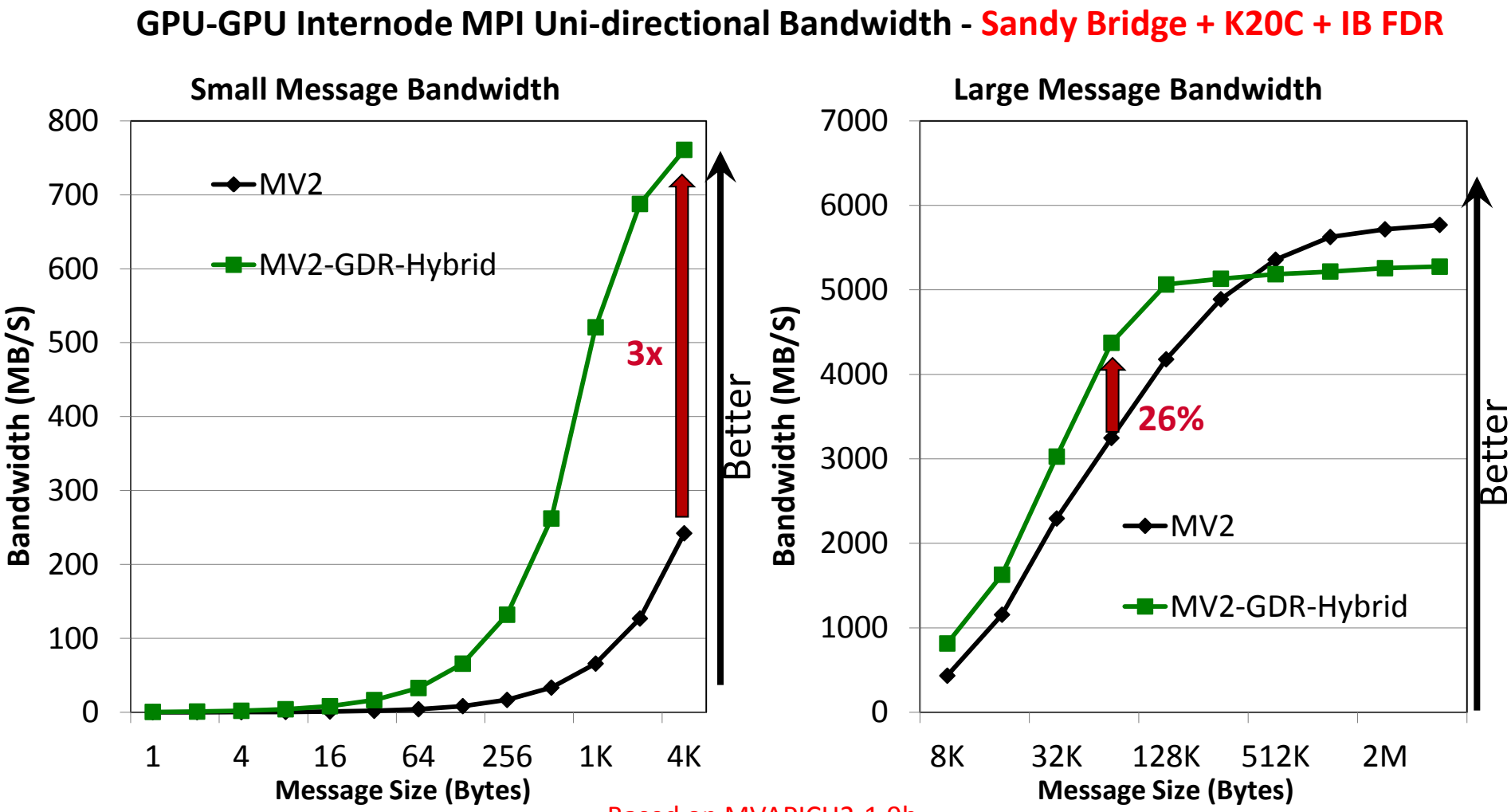
- Performance evaluation has been carried out on four platform configurations:
  - Sandy Bridge, IB FDR, K20C
  - WestmereEP, IB FDR, K20C
  - Sandy Bridge, IB QDR, K20C
  - WestmereEP, IB QDR, K20C

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA



Based on MVAPICH2-1.9b
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

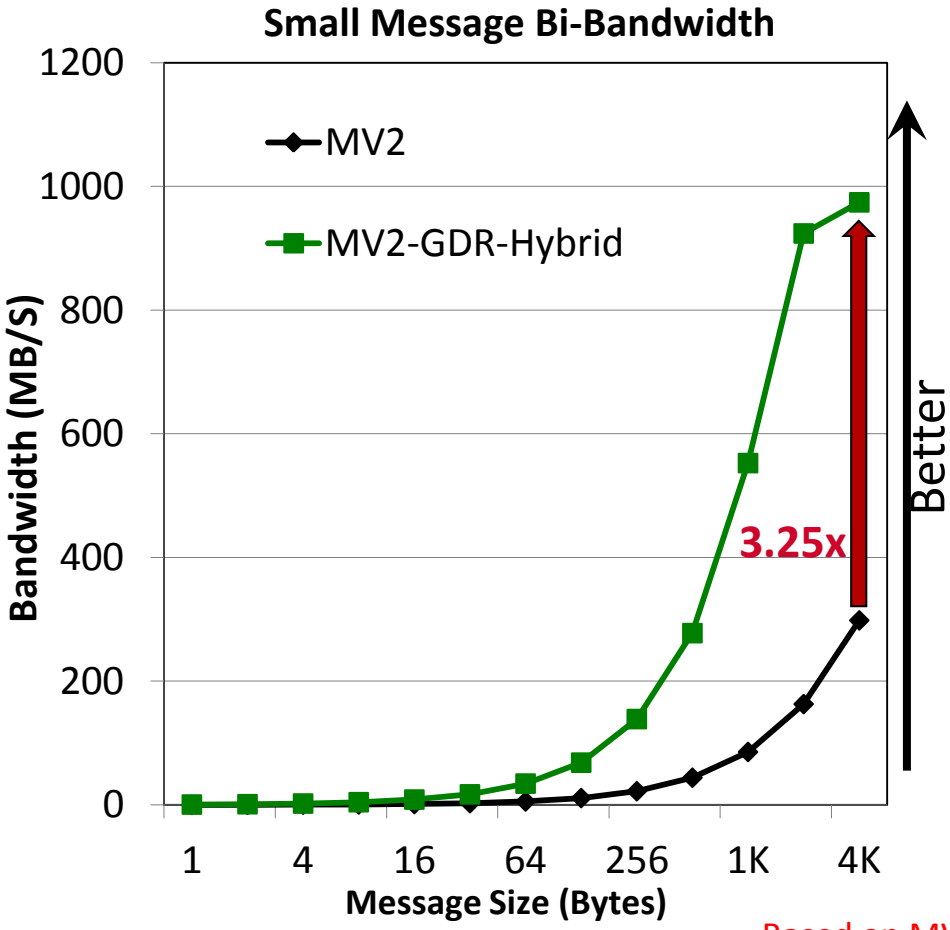
Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

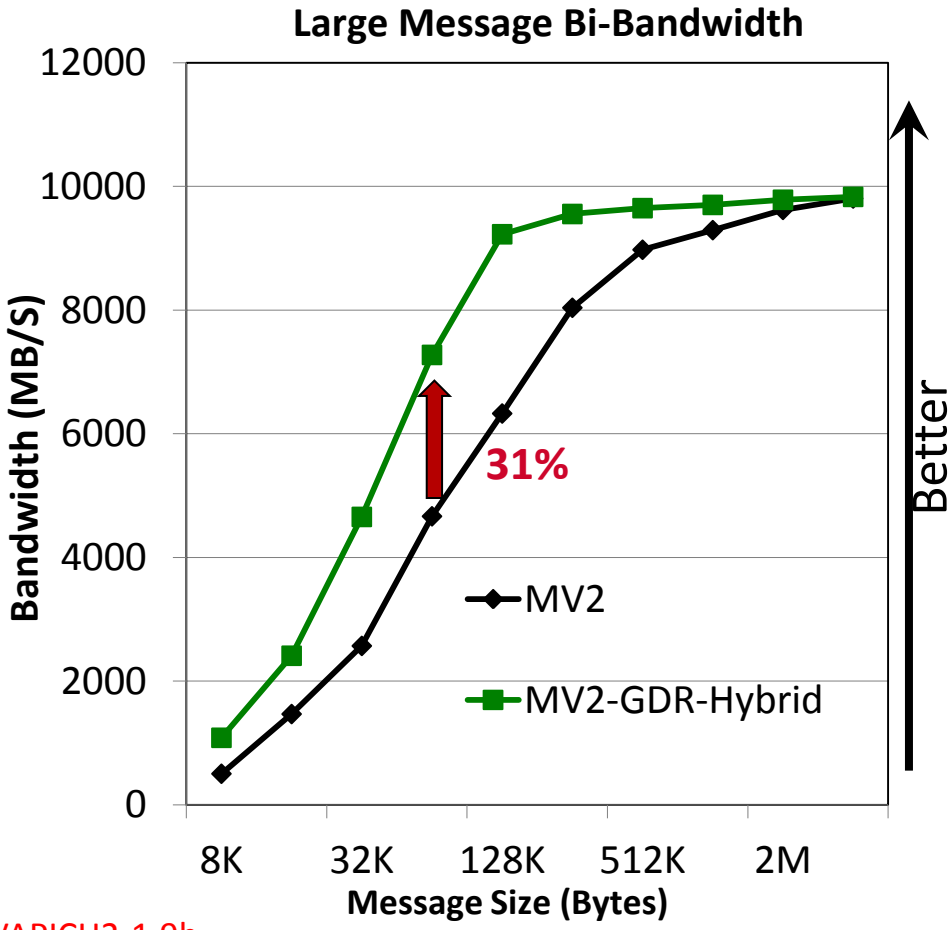


Based on MVAPICH2-1.9b
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

GPU-GPU Internode MPI Bi-directional Bandwidth - Sandy Bridge + K20C + IB FDR



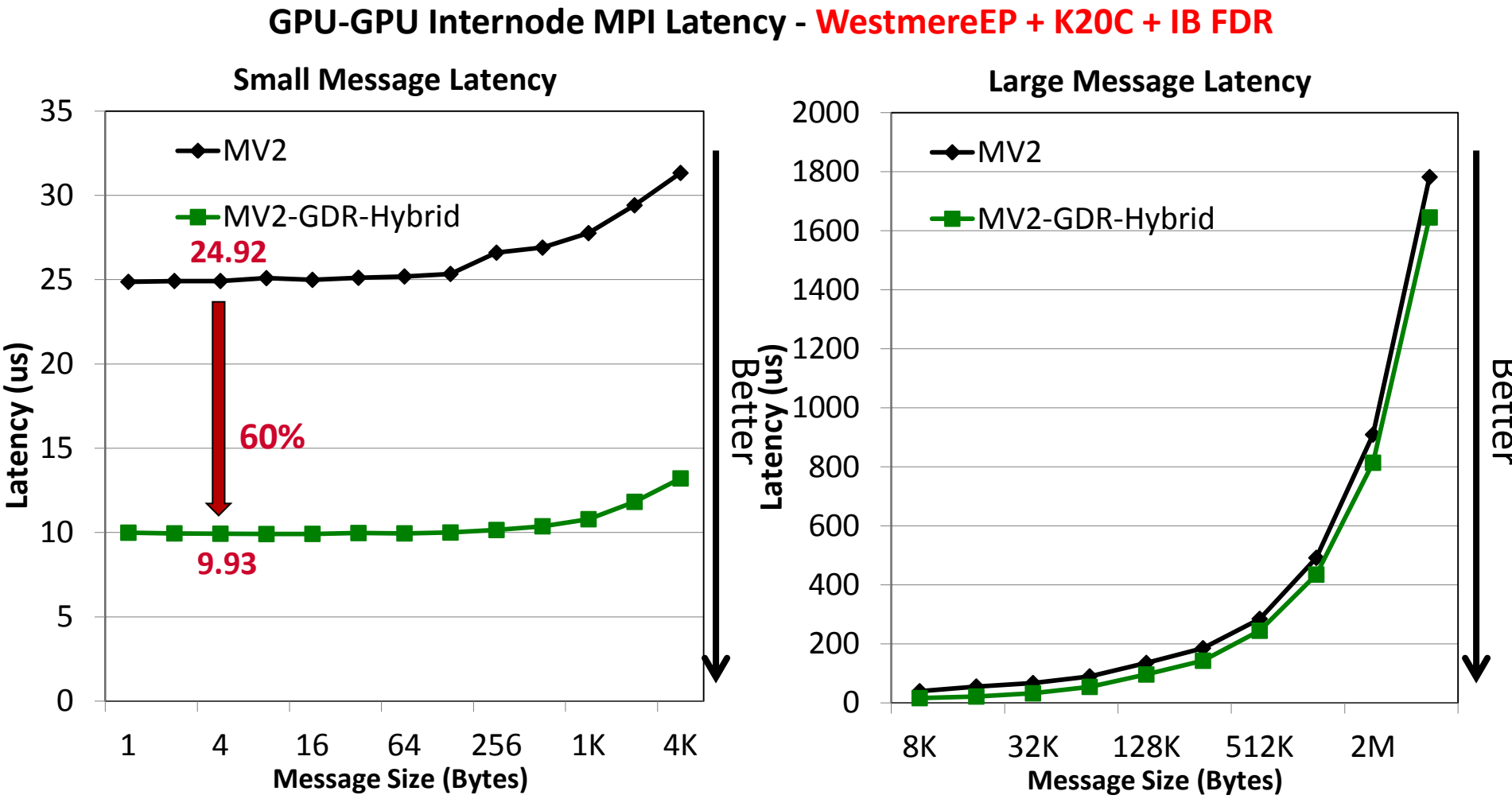


Based on MVAPICH2-1.9b
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance Evaluation of OSU-MVAPICH2 with GPU-Direct-RDMA

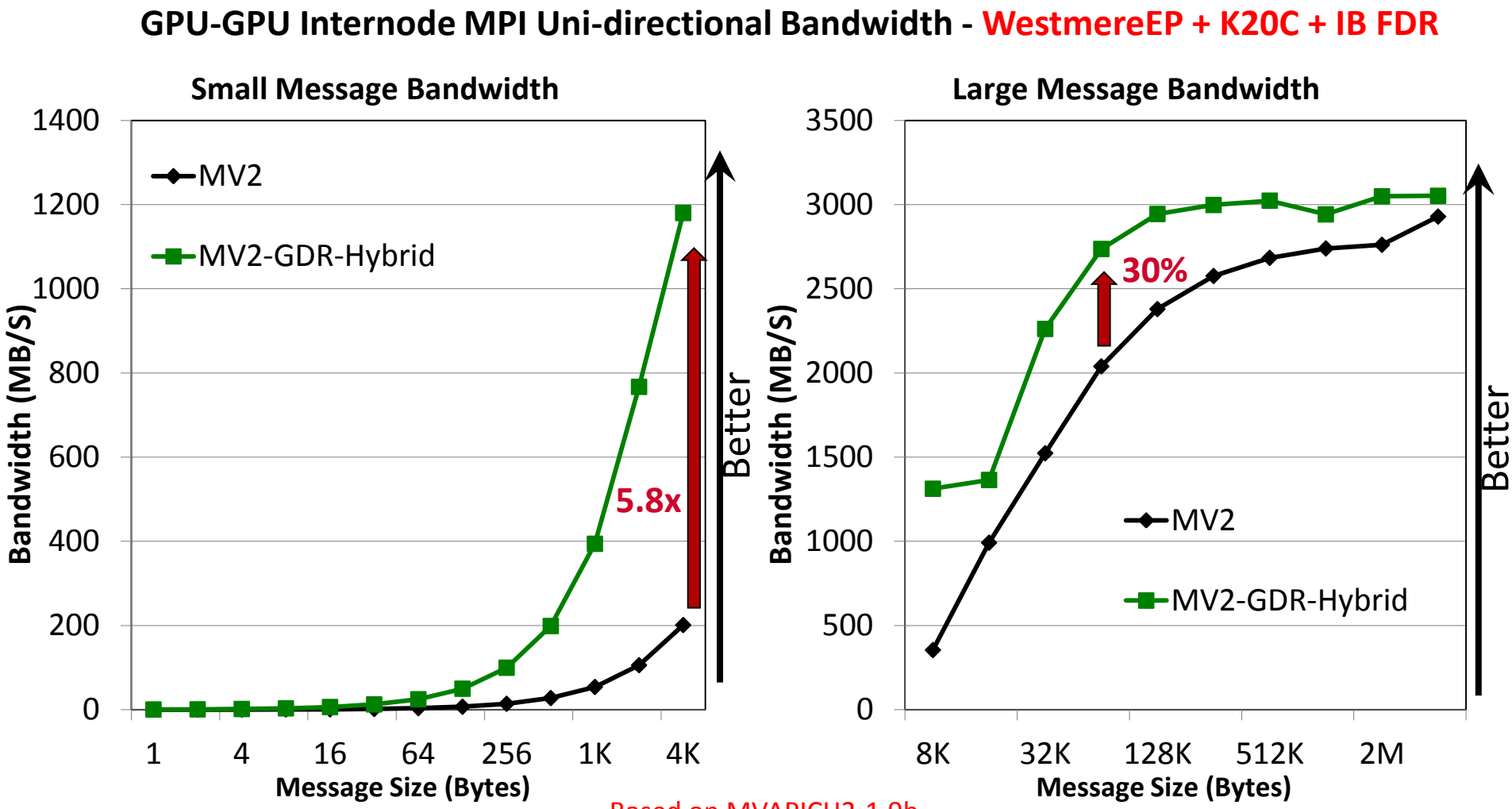
- Performance evaluation has been carried out on four platform configurations:
  - Sandy Bridge, IB FDR, K20C
  - WestmereEP, IB FDR, K20C
  - Sandy Bridge, IB QDR, K20C
  - WestmereEP, IB QDR, K20C

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA



Based on MVAPICH2-1.9b
WestmereEP (E5645) node with 12 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

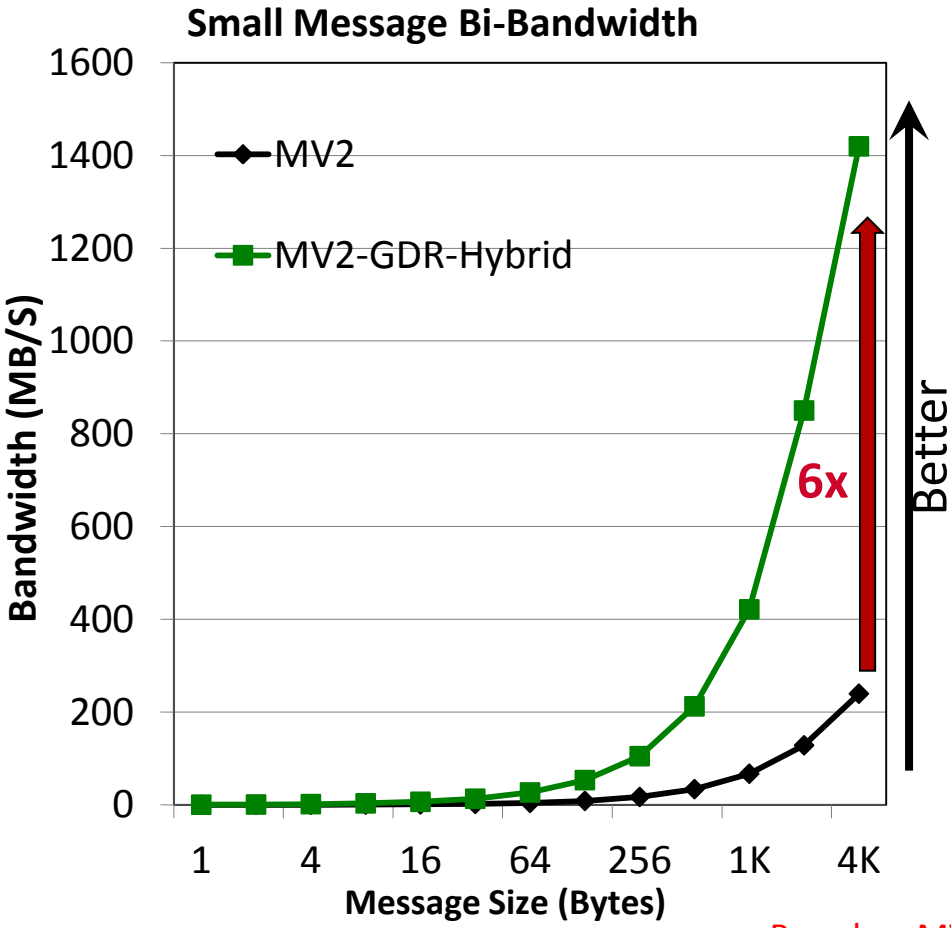
Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

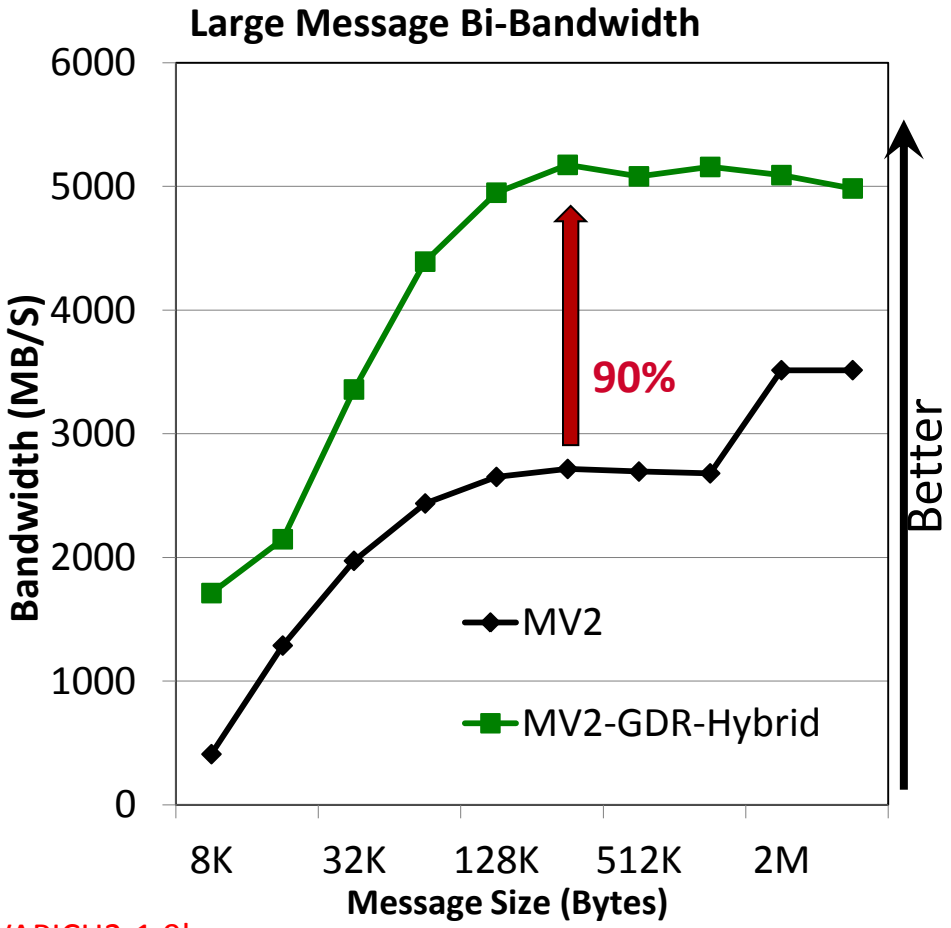


Based on MVAPICH2-1.9b
WestmereEP (E5645) node with 12 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

GPU-GPU Internode MPI Bi-directional Bandwidth - WestmereEP + K20C + IB FDR



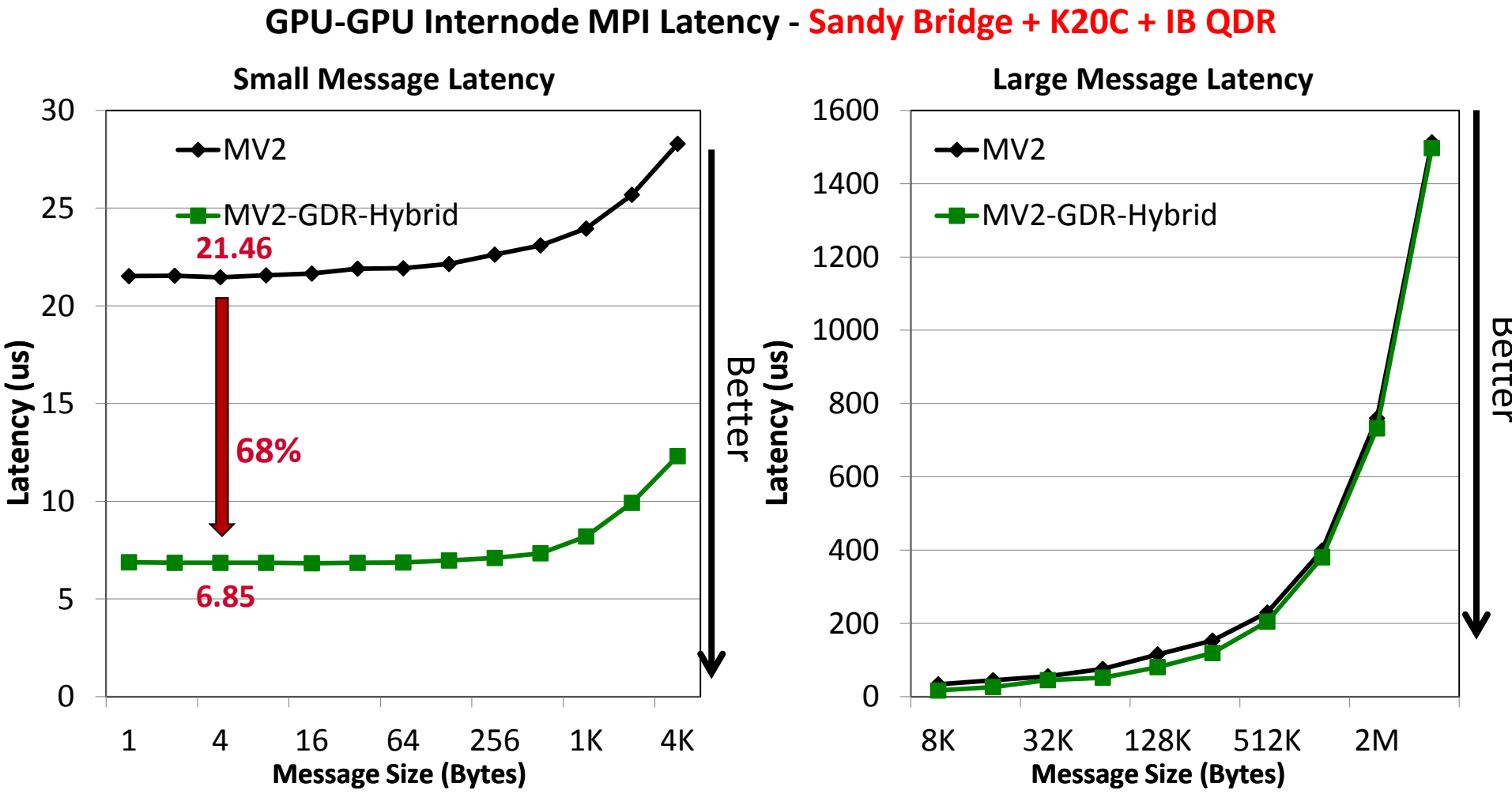


Based on MVAPICH2-1.9b
WestmereEP (E5645) node with 12 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance Evaluation of OSU-MVAPICH2 with GPU-Direct-RDMA

- Performance evaluation has been carried out on four platform configurations:
  - Sandy Bridge, IB FDR, K20C
  - WestmereEP, IB FDR, K20C
  - Sandy Bridge, IB QDR, K20C
  - WestmereEP, IB QDR, K20C

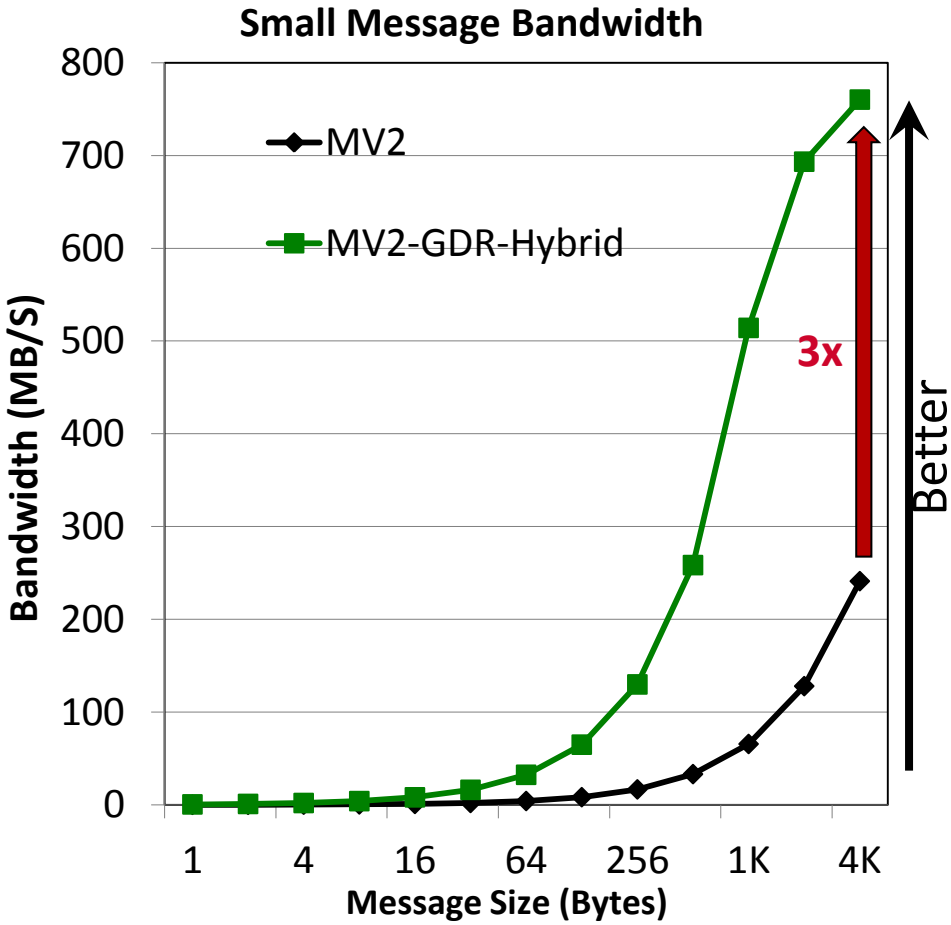
Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

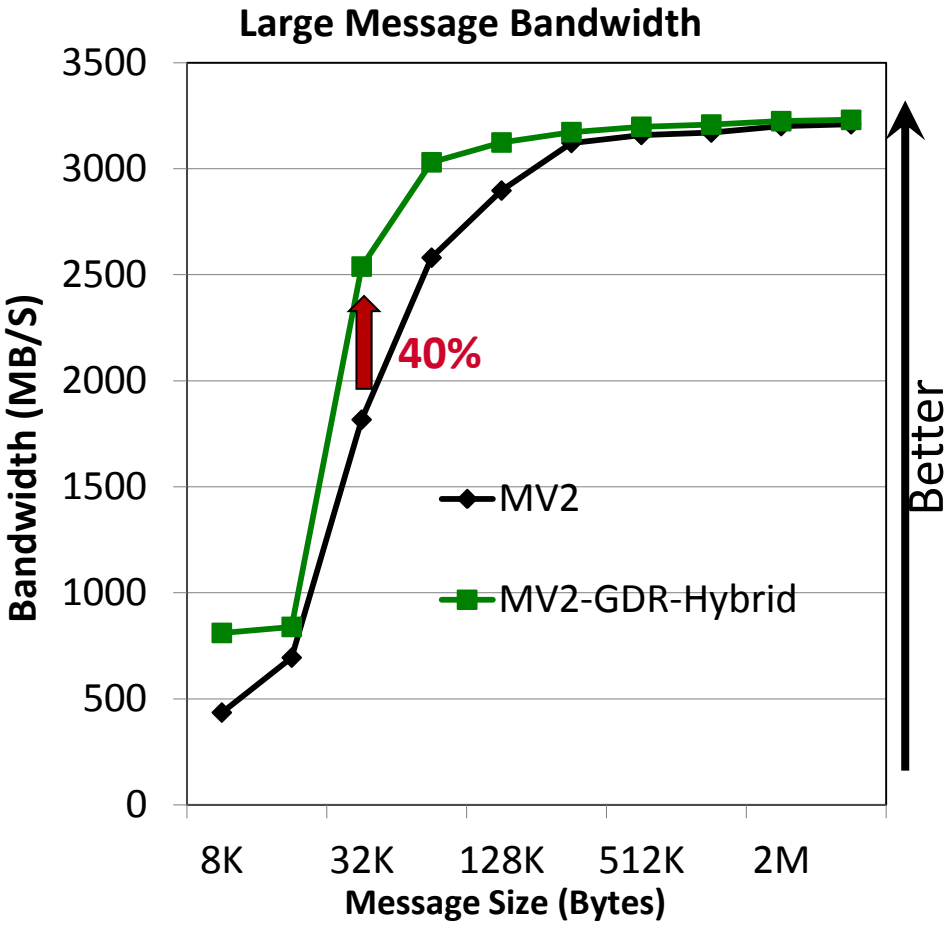


Based on MVAPICH2-1.9b
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-2 QDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

GPU-GPU Internode MPI Uni-directional Bandwidth - Sandy Bridge + K20C + IB QDR



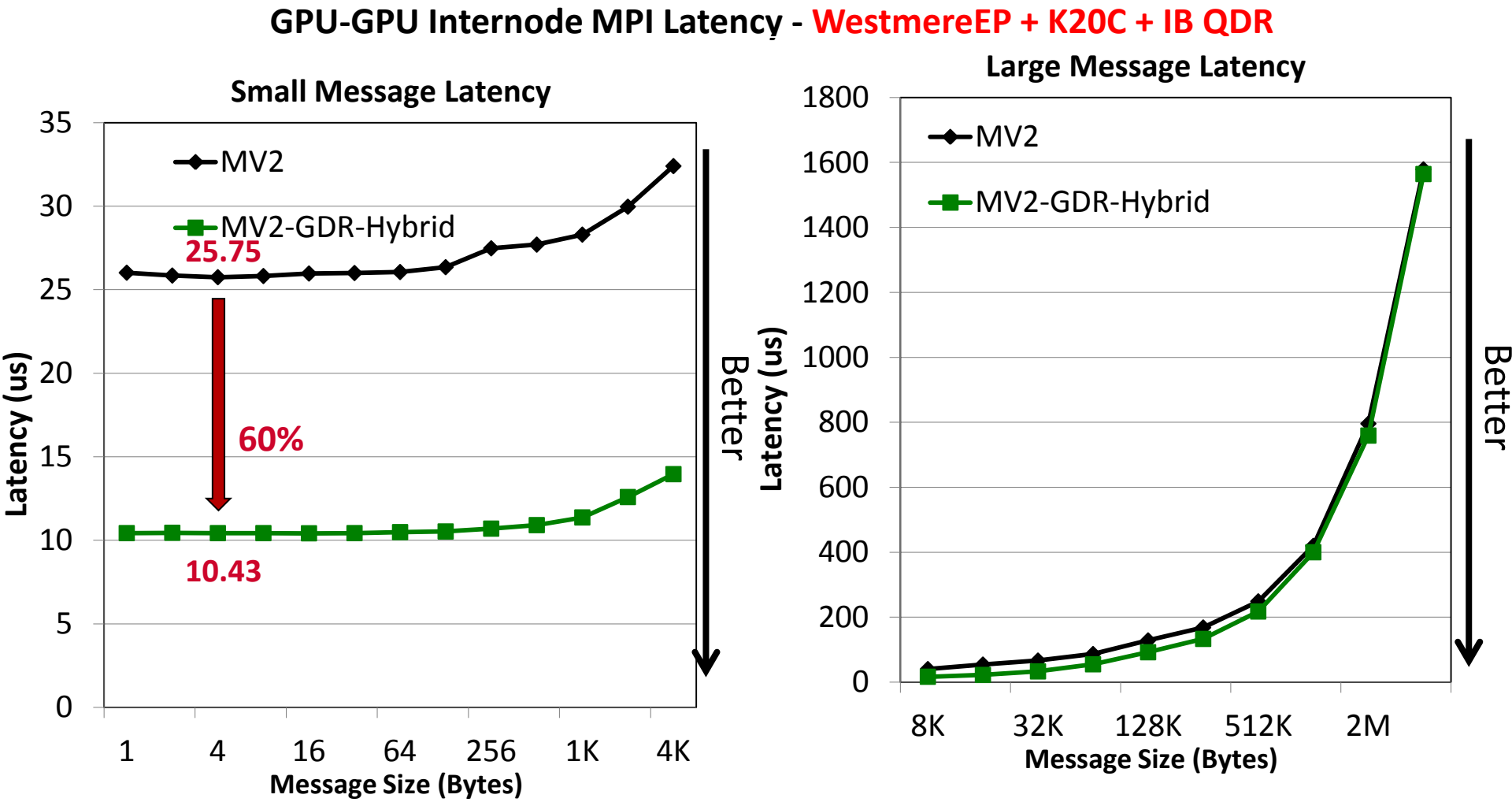


Based on MVAPICH2-1.9b
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-2 QDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance Evaluation of OSU-MVAPICH2 with GPU-Direct-RDMA

- Performance evaluation has been carried out on four platform configurations:
  - Sandy Bridge, IB FDR, K20C
  - WestmereEP, IB FDR, K20C
  - Sandy Bridge, IB QDR, K20C
  - WestmereEP, IB QDR, K20C

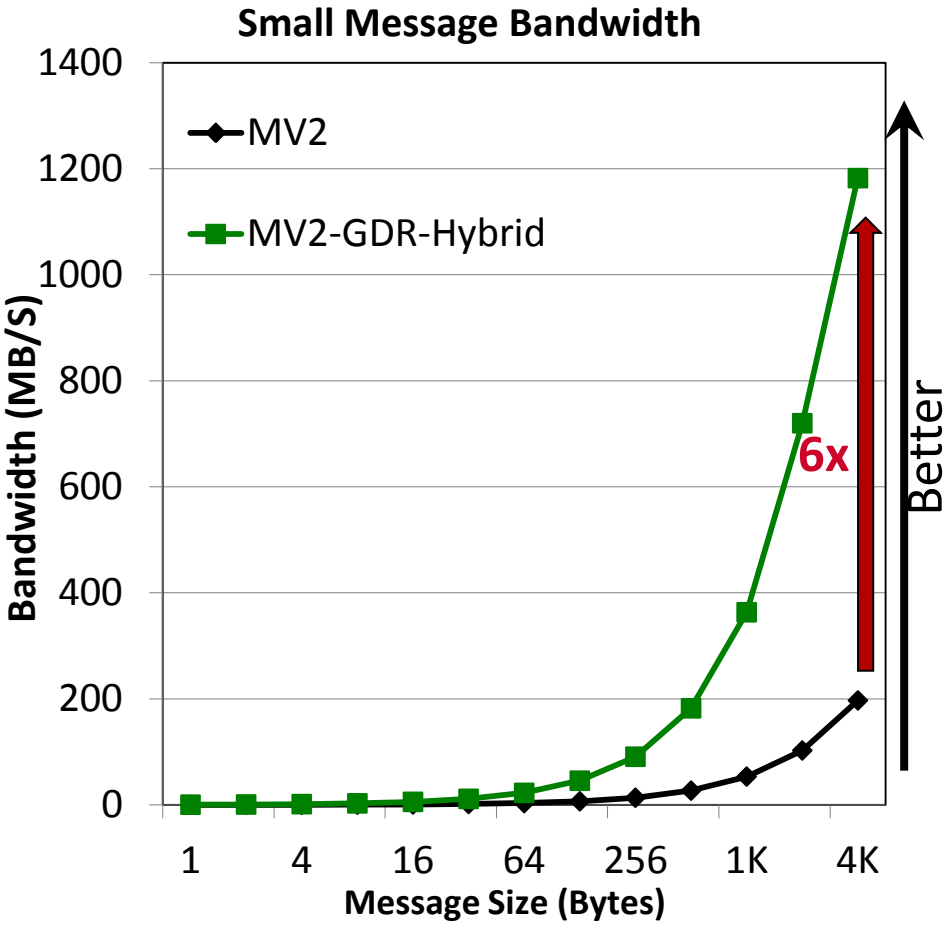
Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

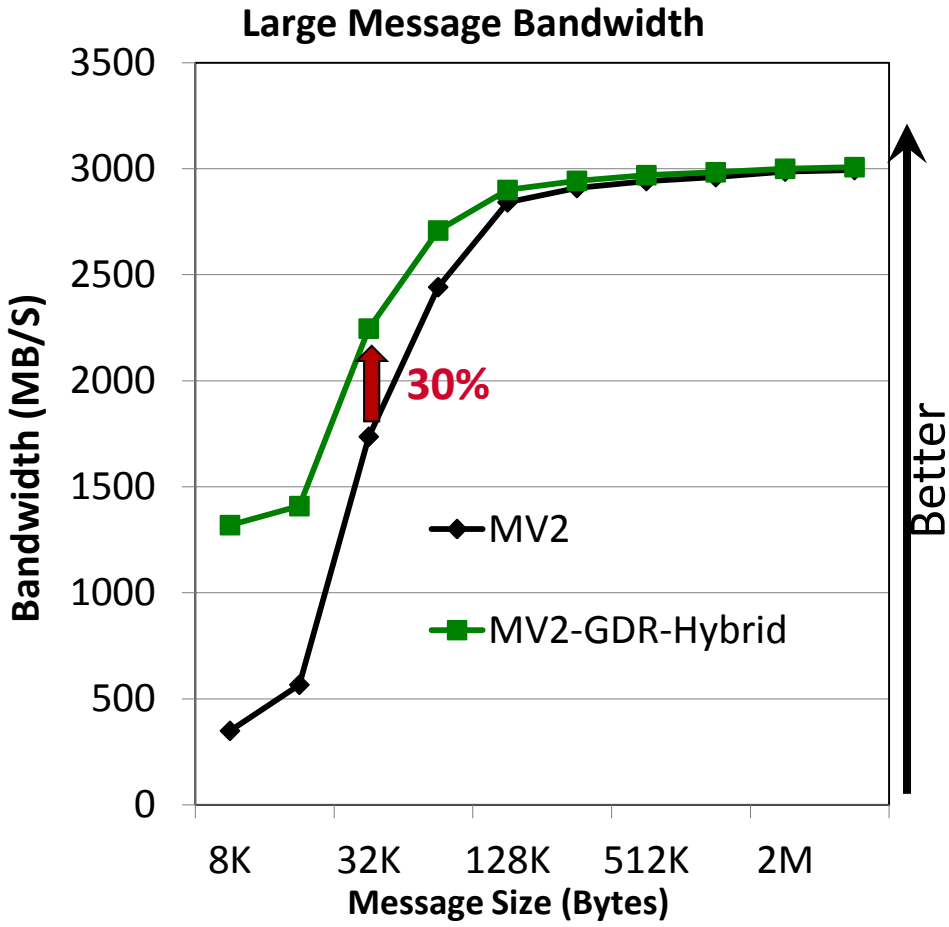


Based on MVAPICH2-1.9b
WestmereEP (E5645) node with 12 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-2 QDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA

GPU-GPU Internode MPI Uni-directional Bandwidth - WestmereEP + K20C + IB QDR





Based on MVAPICH2-1.9b
WestmereEP (E5645) node with 12 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-2 QDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

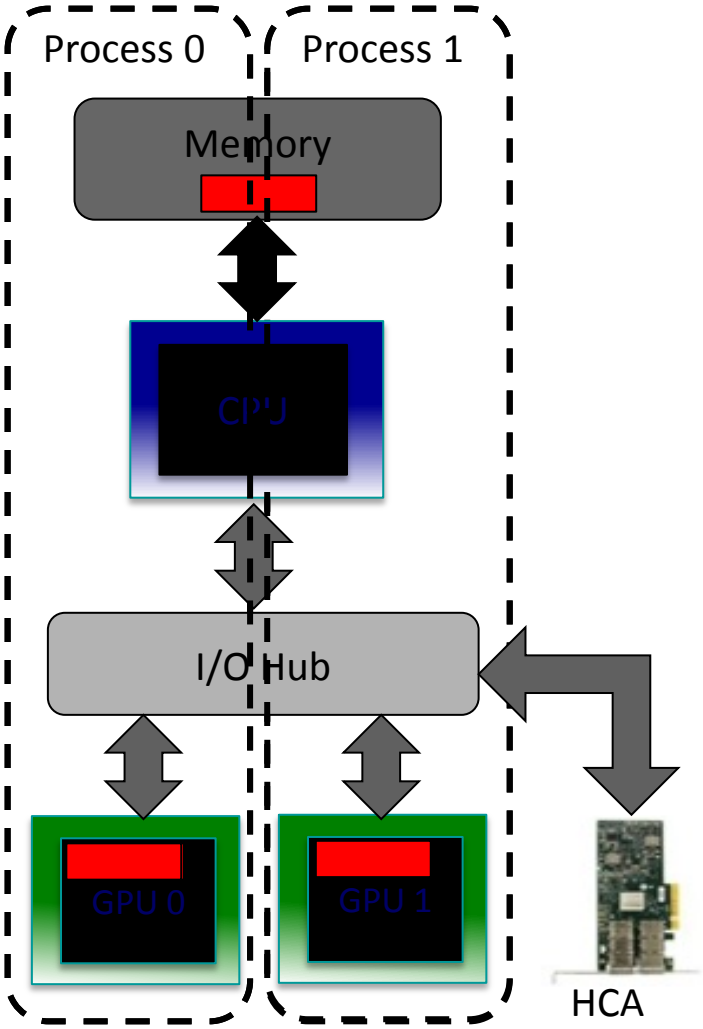
MVAPICH2 Release with GPUDirect RDMA Hybrid

- Further tuning and optimizations (such as collectives) to be done
- GPUDirect RDMA support in Open Fabrics Enterprise Distribution (OFED) is expected during Q2 ‘13 (according to Mellanox)
- MVAPICH2 release with GPUDirect RDMA support will be timed accordingly

Outline

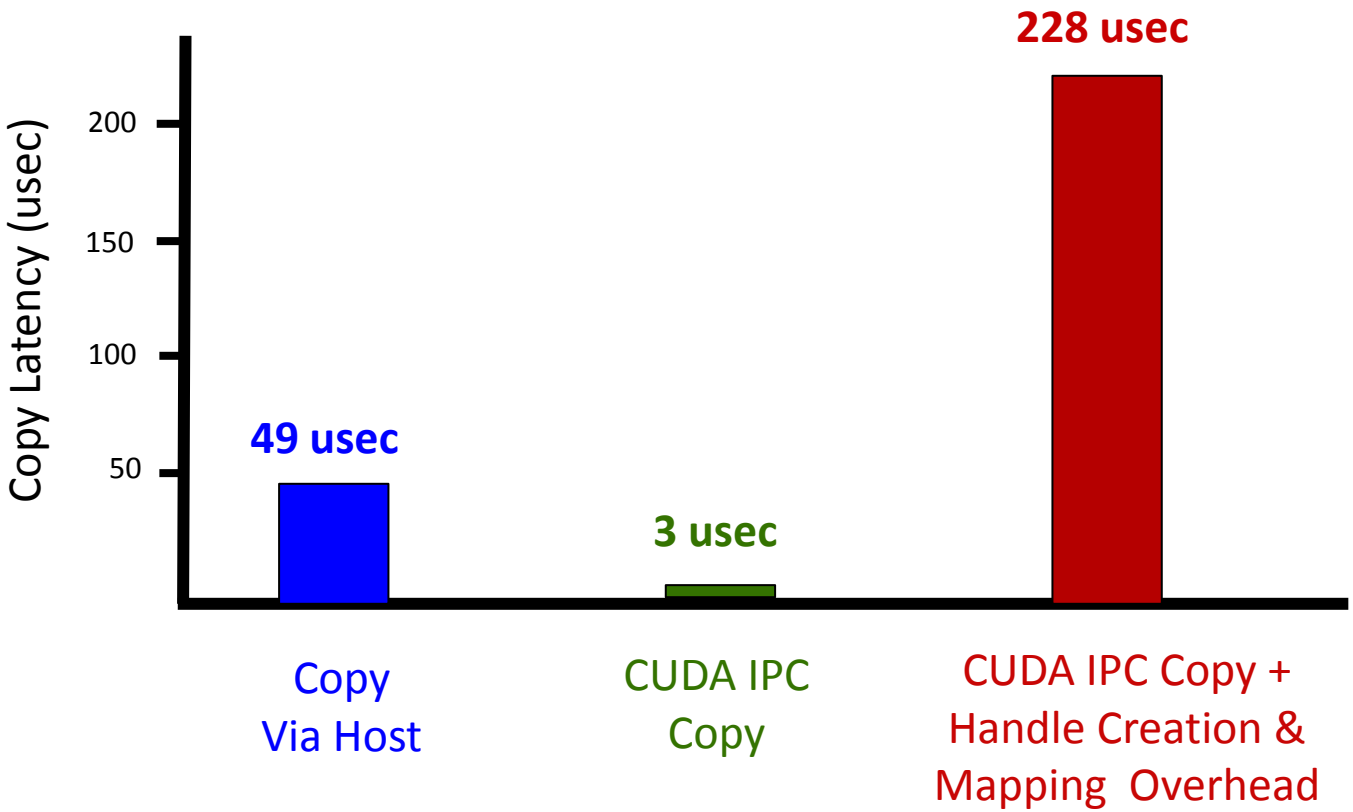
- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype Processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

Multi-GPU Configurations



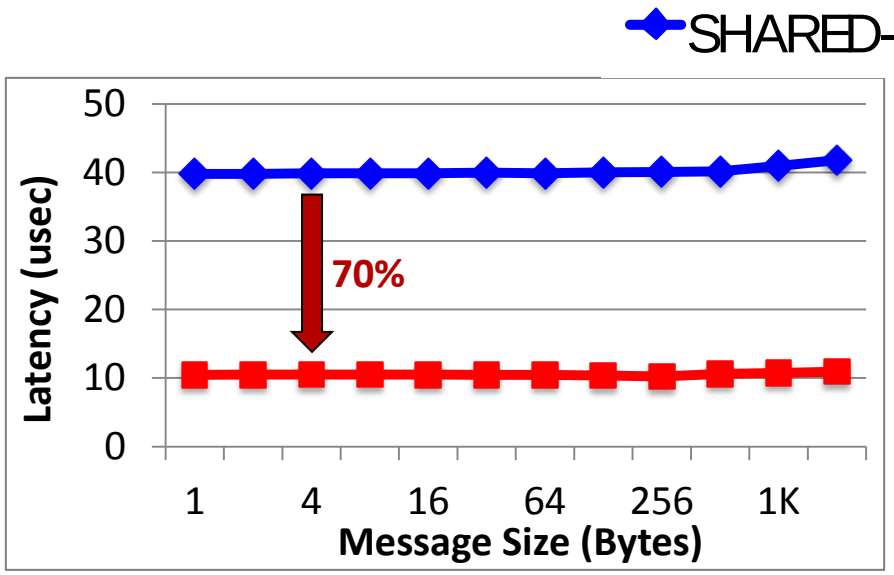
- Multi-GPU node architectures are becoming common
- Until CUDA 3.2
  - Communication between processes staged through the host
  - Shared Memory (pipelined)
  - Network Loopback [asynchronous)
- CUDA 4.0
  - Inter-Process Communication (IPC)
  - Host bypass
  - Handled by a DMA Engine
  - Low latency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

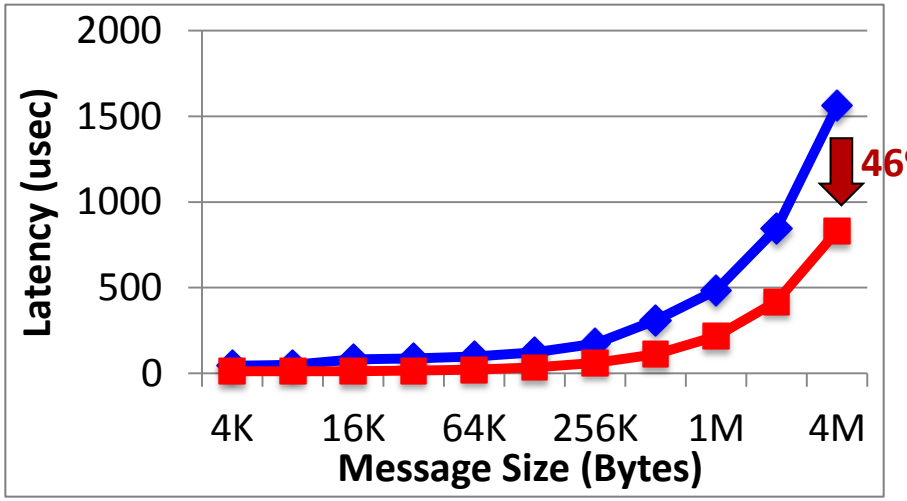
Comparison of Costs

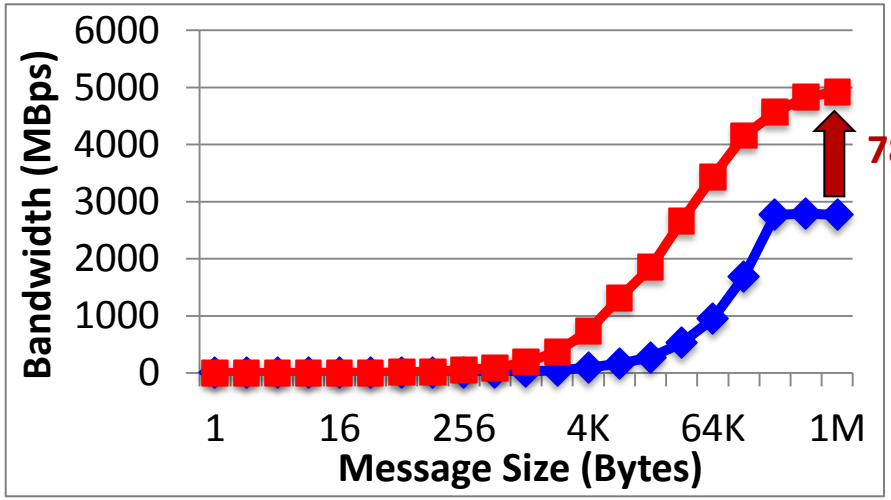


- Comparison of bare copy costs between two processes on one node, each using a different GPU (outside MPI)
- MVAPICH2 takes advantage of CUDA IPC while hiding the handle creation and mapping overheads from the user

Two-sided Communication Performance



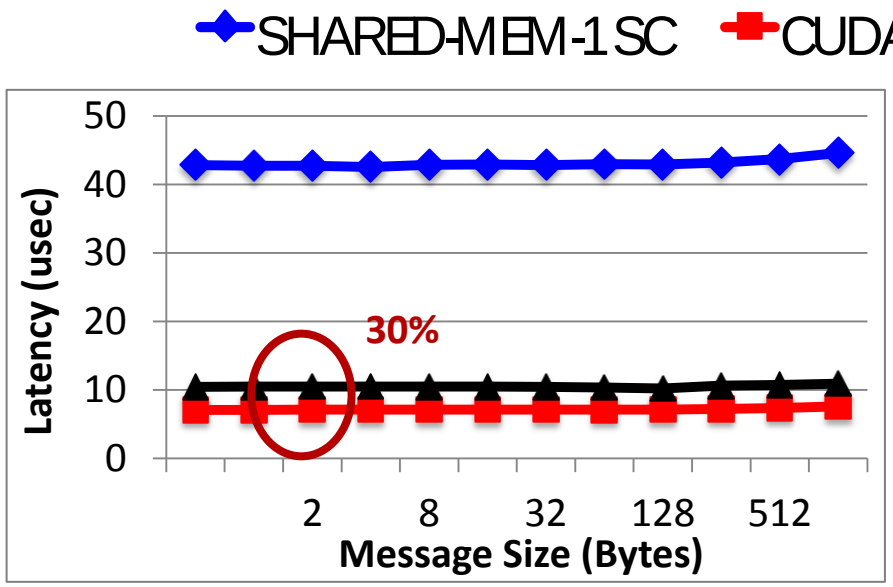


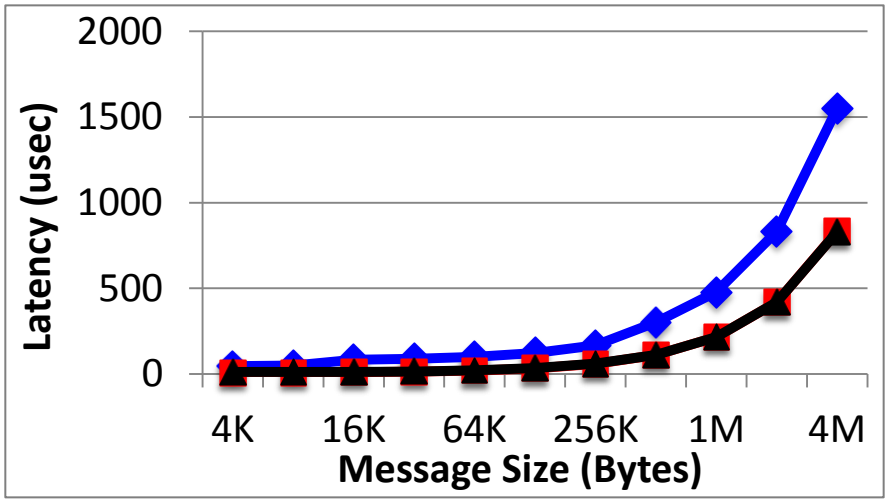


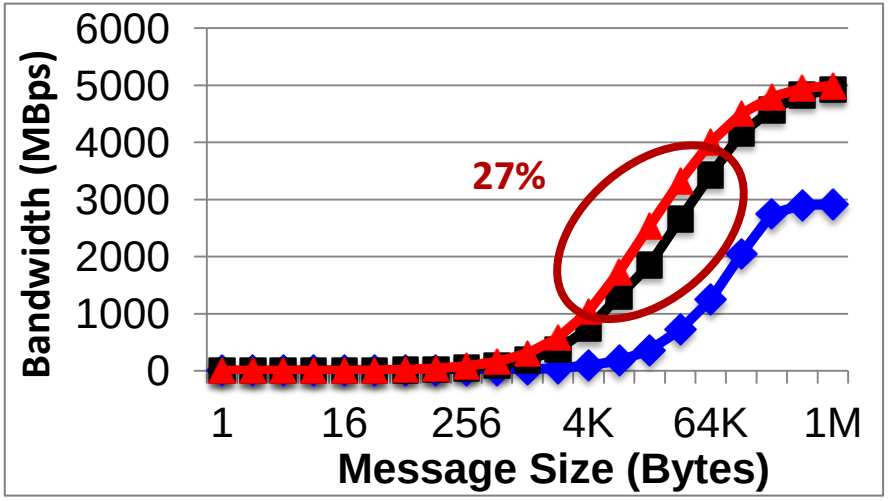
Already available in MVAPICH2 1.8 and 1.9

One-sided Communication Performance

(get + active synchronization vs. send/recv)







one-sided semantics harness better performance compared to two-sided semantics.

Support for one-sided communication from GPUs will be available in future releases of MVAPICH2

Outline

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU
  - Internode Communication
    - Point-to-point Communication
    - Collective Communication
    - MPI Datatype Processing
    - Using GPUDirect RDMA
  - Multi-GPU Configurations
- MPI and OpenACC
- Conclusion

OpenACC

- 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..**

Using MVAPICH2 with OpenACC 1.0

- 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);
```

Using MVAPICH2 with the new OpenACC 2.0

- 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
  - Expected to deliver 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);
```

Conclusions

- MVAPICH2 optimizes MPI communication on InfiniBand clusters with GPUs
- Point-to-point, collective communication and datatype processing are addressed
- Takes advantage of CUDA features like IPC and GPUDirect RDMA
- Optimizations under the hood of MPI calls, hiding all the complexity from the user
- High productivity and high performance

Web Pointers

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

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

MVAPICH Web Page

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

panda@cse.ohio-state.edu