Advanced MPI Capabilities

VSCSE Webinar (May 6-8, 2014)

Day 3

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda

Karen Tomko
Ohio Supercomputer Center
E-mail: ktomko@osc.edu
http://www.osc.edu/~ktomko

Plans for Thursday

- Tuesday, May 6 – MPI-3 Additions to the MPI Spec
  - Updates to the MPI One-Sided Communication Model (RMA)
  - Non-Blocking Collectives
  - MPI Tools Interface
- Wednesday, May 7 – MPI/PGAS Hybrid Programming
  - MVAPICH2-X: Unified runtime for MPI+PGAS
  - MPI+OpenSHMEM
  - MPI+UPC
- Thursday, May 8 – MPI for many-core processor
  - MVAPICH2-GPU: CUDA-aware MPI for NVidia GPU
  - MVAPICH2-MIC Design for Clusters with InfiniBand and Intel Xeon Phi

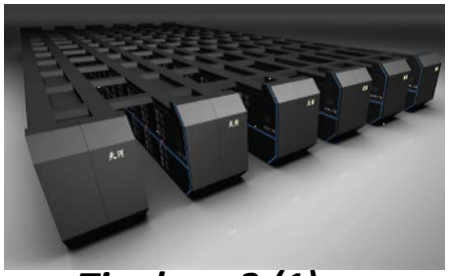
Drivers of Modern HPC Cluster Architectures

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



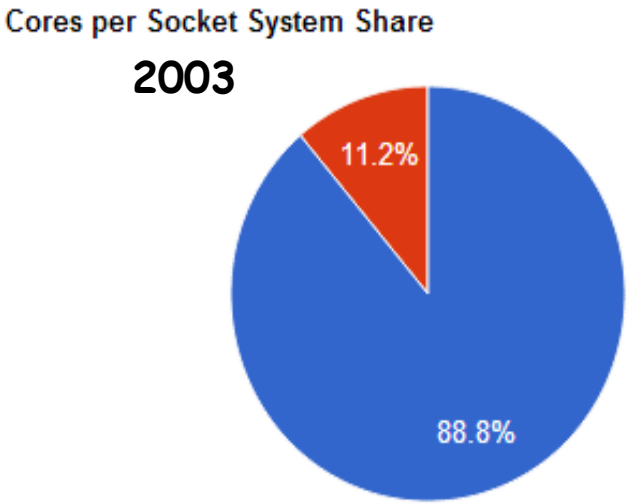
Tianhe – 2 (1)

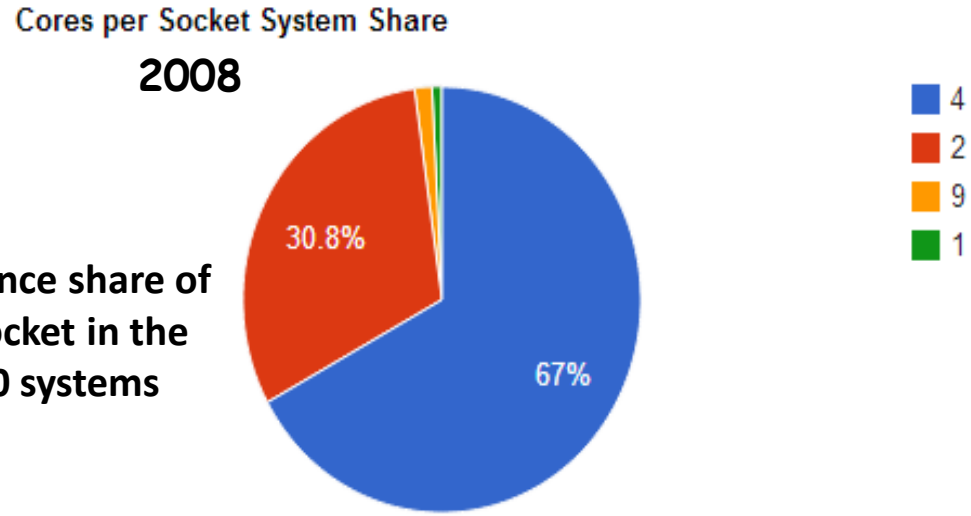
Titan (2)

Stampede (6)

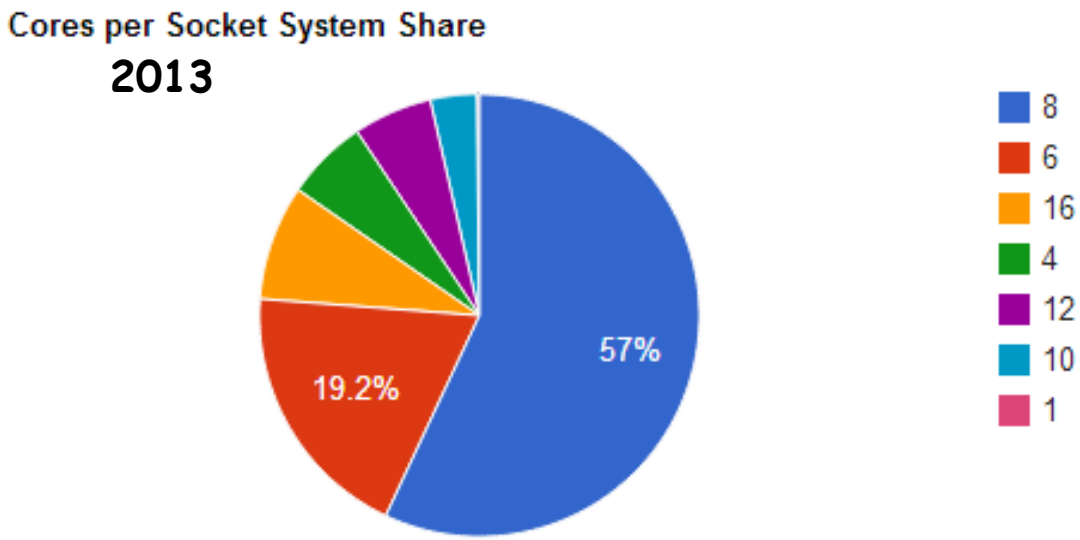
Tianhe – 1A (10)

The Multi-core Era

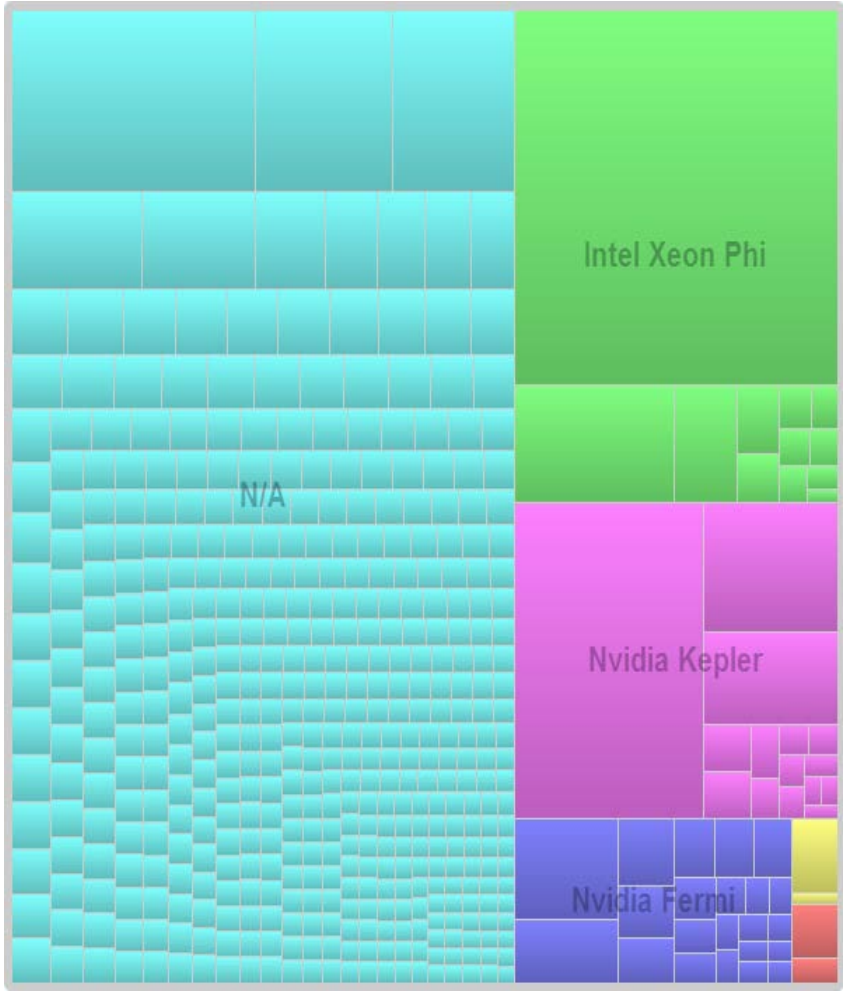




Performance share of cores/socket in the Top500 systems



Advent of Accelerators



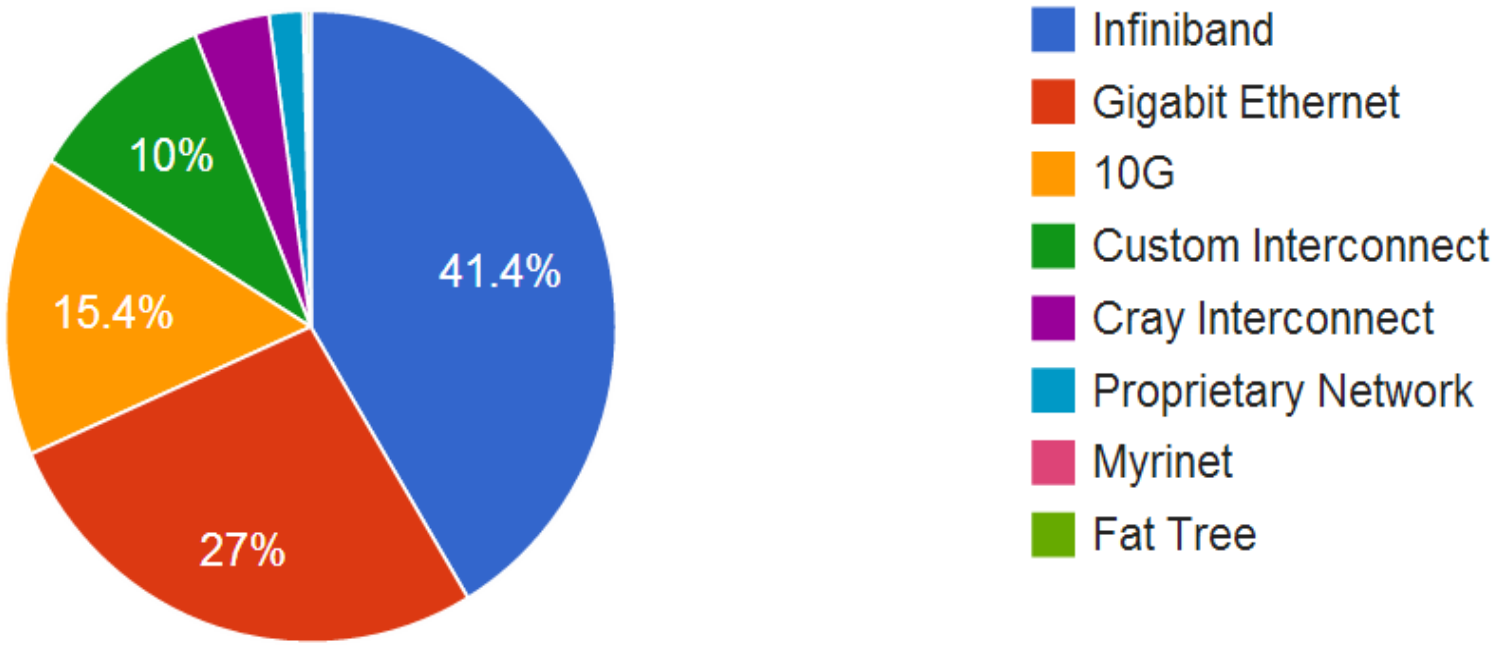
Performance share of accelerators in the Top500 systems

18.1%

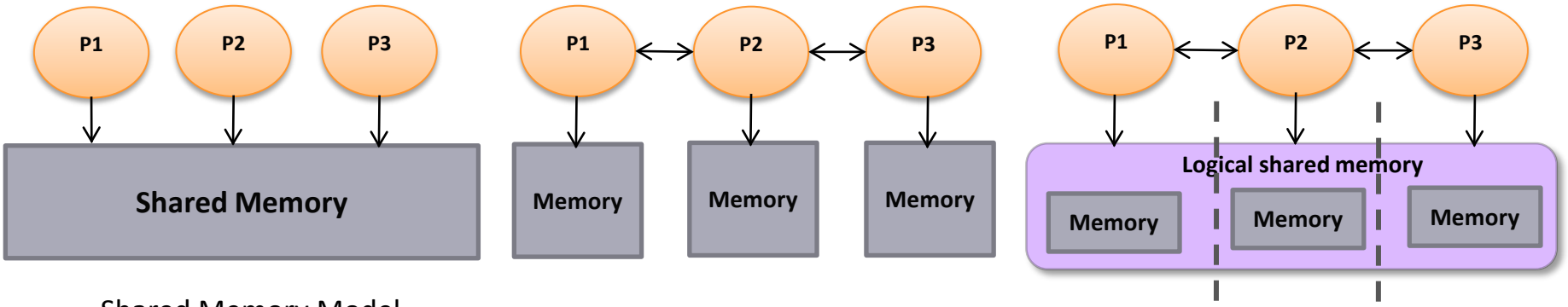
16.2%

State-of-art in Interconnects

Performance share of Interconnects in the Top500 systems



Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

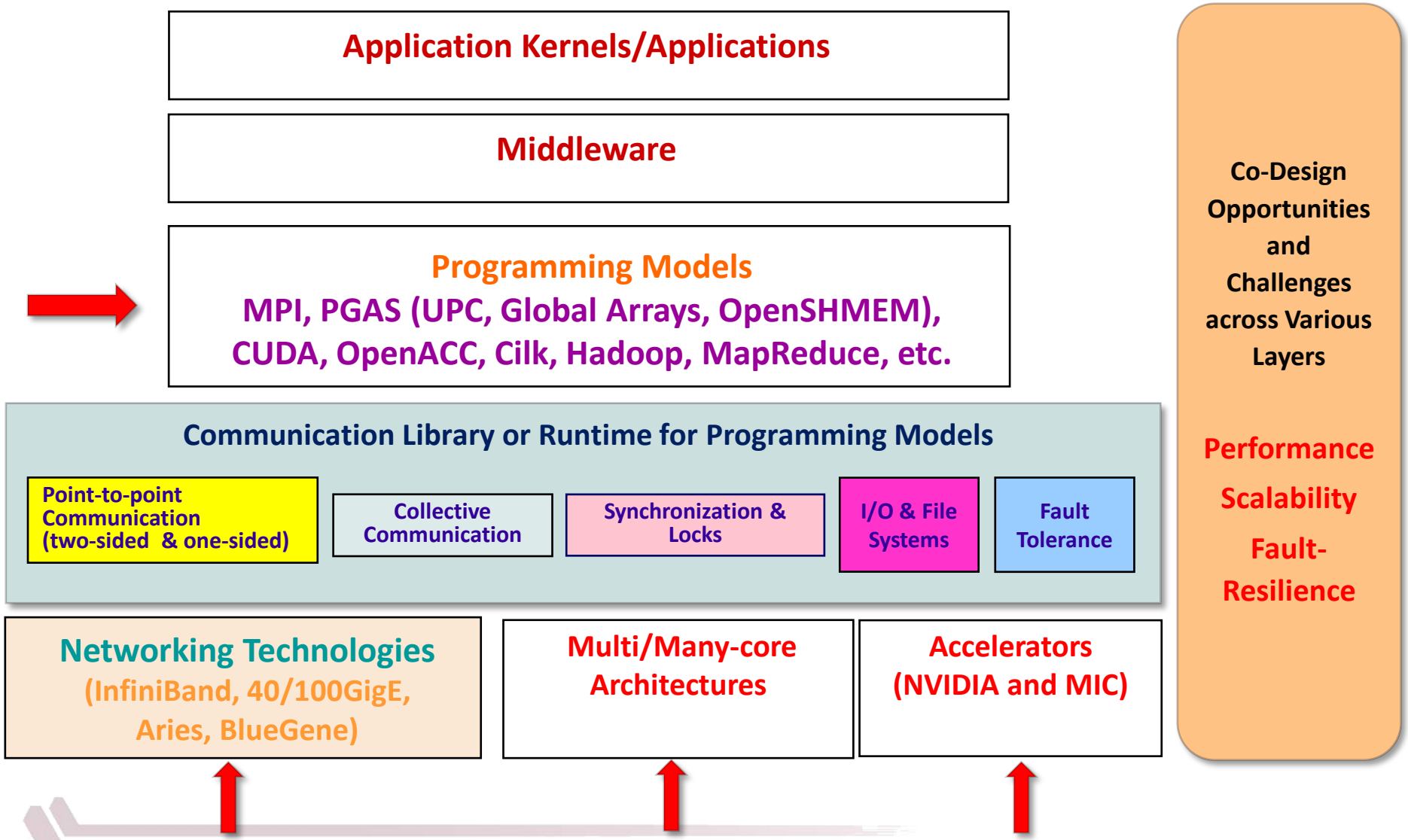
MPI (Message Passing Interface)

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.
- We concentrated on MPI during Day 1
- PGAS and Hybrid MPI+PGAS during Day 2

Designing Software Libraries for Multi-Petaflop and Exaflop Systems: Challenges

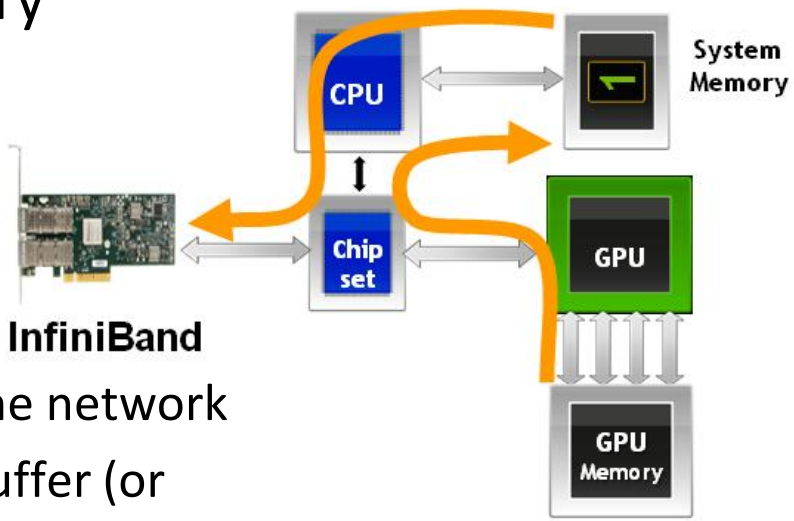


Presentation Overview

- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- MVAPICH2 and OpenACC
- Programming Models for Intel MIC
- MVAPICH2-MIC Designs

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*
- *Latest GPUDirect RDMA achieves this*

Sample Code - Without MPI integration

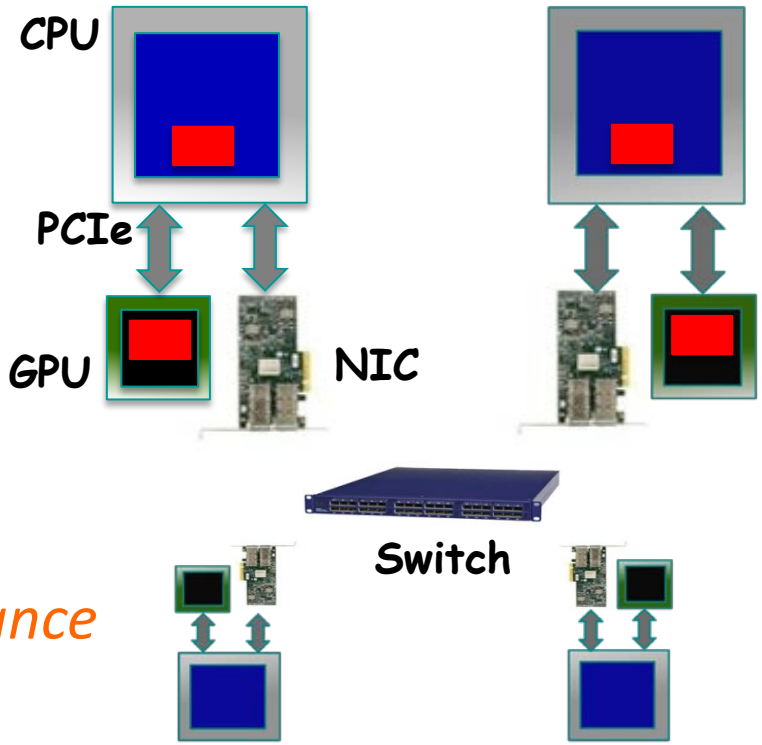
- Naïve implementation with standard MPI and CUDA

At Sender:

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

At Receiver:

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



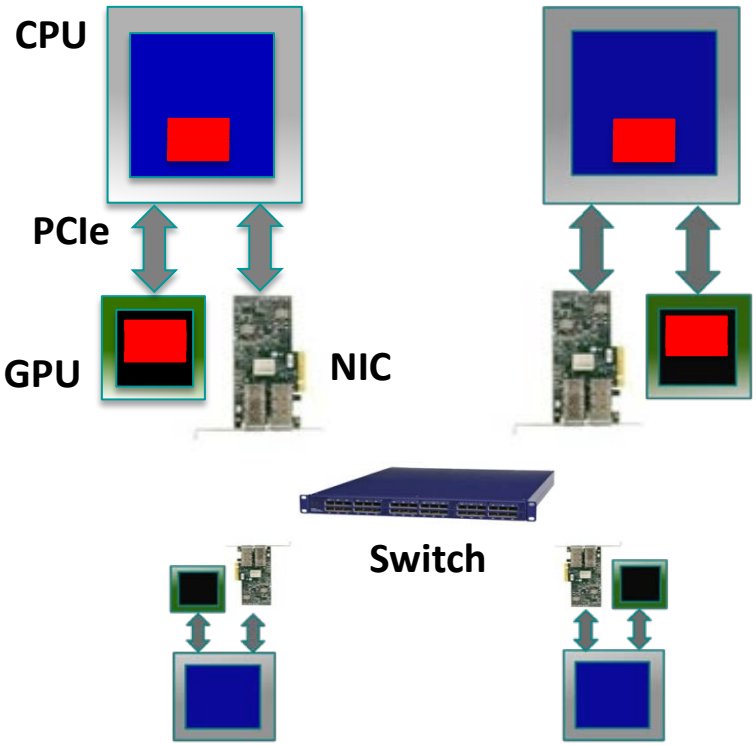
- *High Productivity and Poor Performance*

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 incorporated in MVAPICH2 to support this functionality**

Sample Code – MVAPICH2-GPU

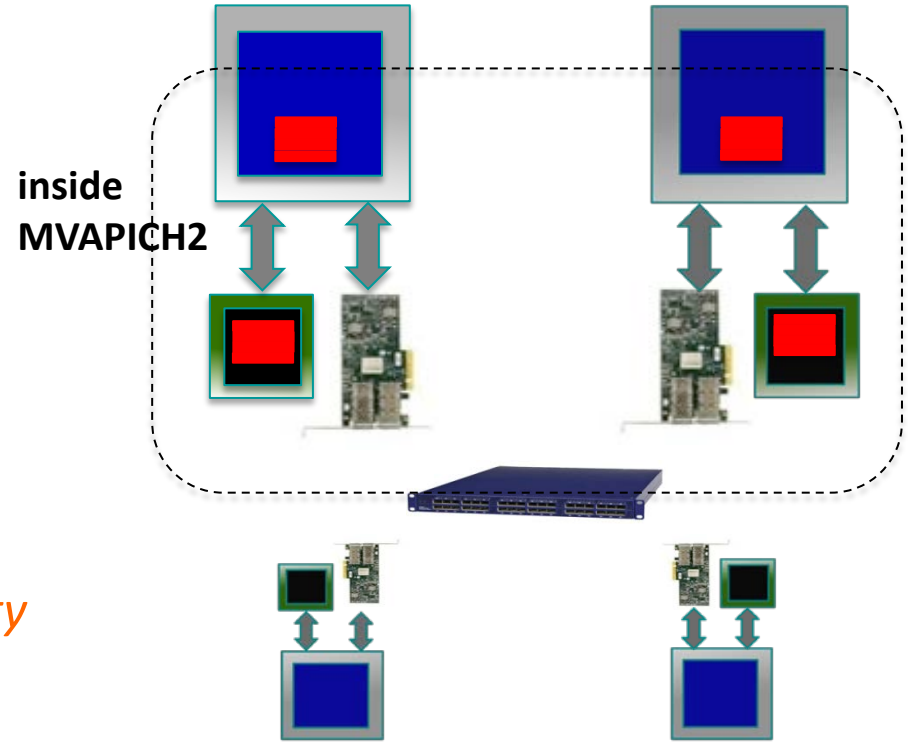
- MVAPICH2-GPU: standard MPI interfaces used
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_device, size, …);
```

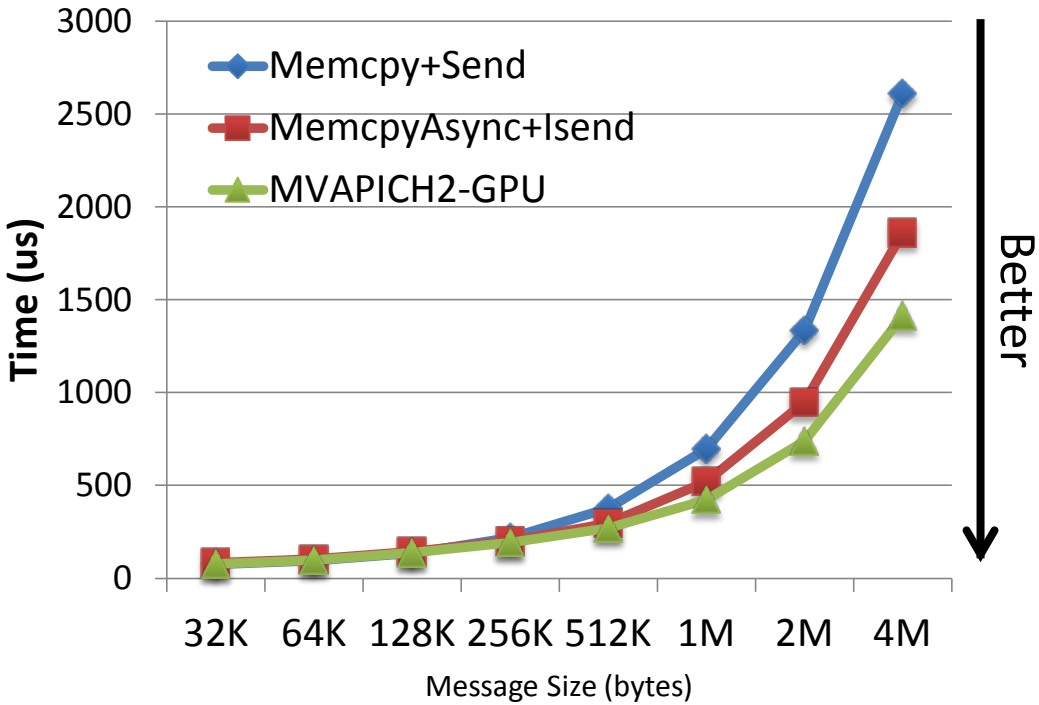
At Receiver:

```
MPI_Recv(r_device, size, …);
```



- *High Performance and High Productivity*
- **CUDA-Aware MPI**

MPI-Level Two-sided Communication



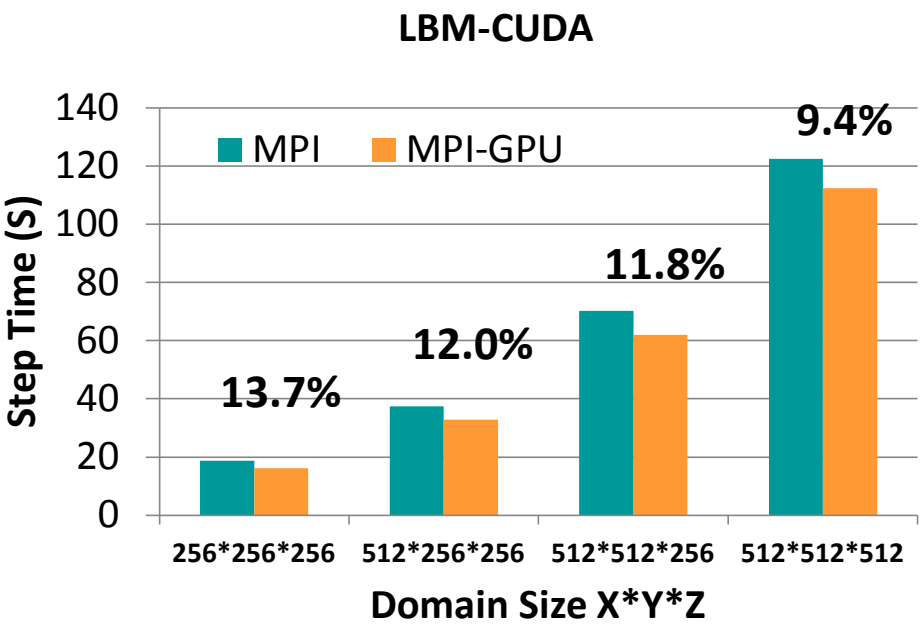
- 45% improvement compared with a naïve user-level implementation (Memcpy+Send), for 4MB messages
- 24% improvement compared with an advanced user-level implementation (MemcpyAsync+Isend), for 4MB messages

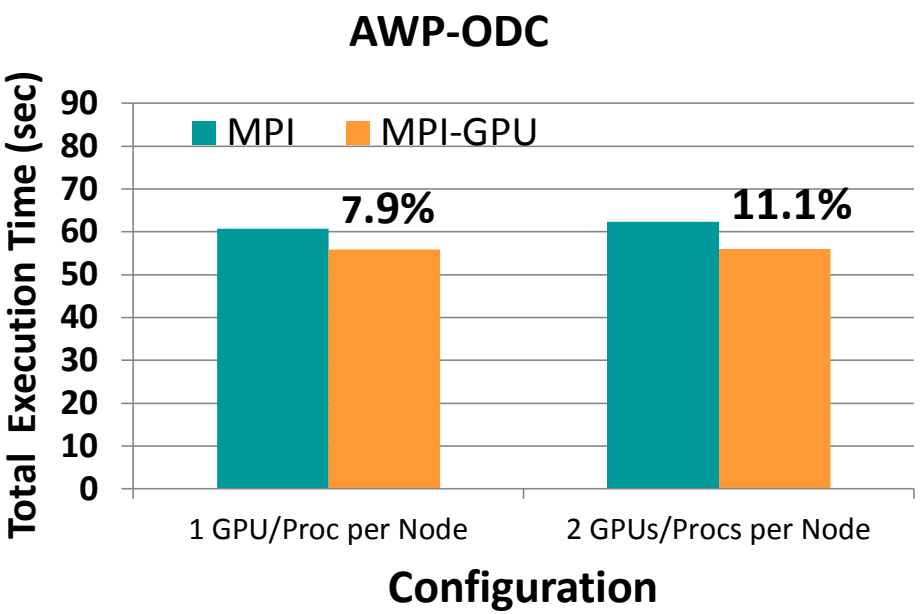
H. Wang, S. Potluri, M. Luo, A. Singh, S. Sur and D. K. Panda, MVAPICH2-GPU: Optimized GPU to GPU Communication for InfiniBand Clusters, ISC '11

MVAPICH2 1.8, 1.9, 2.0a-2.0rc1 Releases

- 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

Applications-Level Evaluation





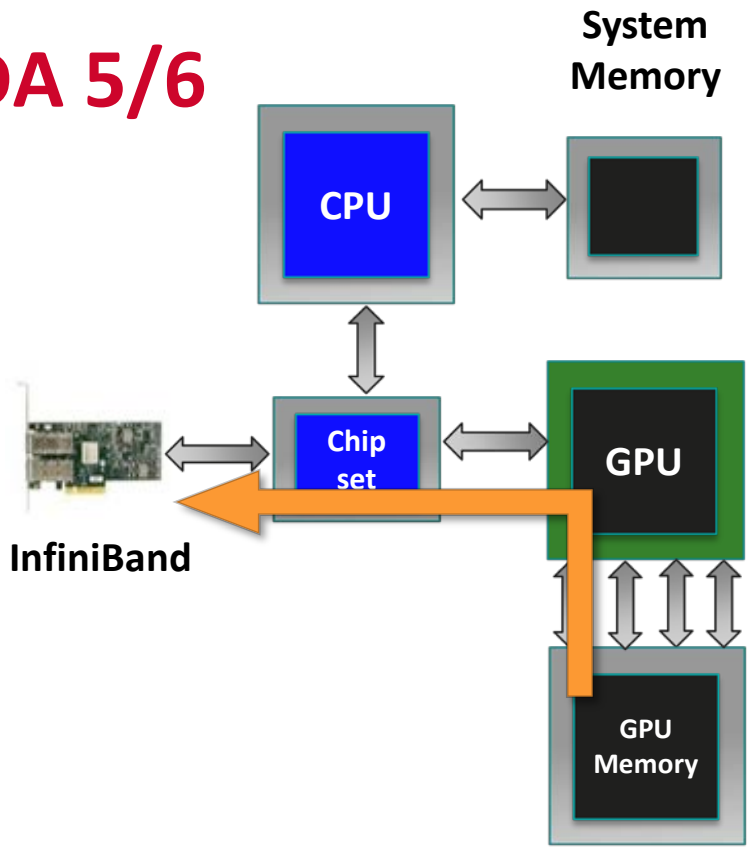
- LBM-CUDA (Courtesy: Carlos Rosale, TACC)
  - Lattice Boltzmann Method for multiphase flows with large density ratios
  - one process/GPU per node, 16 nodes
- AWP-ODC (Courtesy: Yifeng Cui, SDSC)
  - a seismic modeling code, Gordon Bell Prize finalist at SC 2010
  - 128x256x512 data grid per process, 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

Presentation Overview

- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
- MVAPICH2 and OpenACC
- Programming Models for Intel MIC
- MVAPICH2-MIC Designs

GPU-Direct RDMA with CUDA 5/6

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



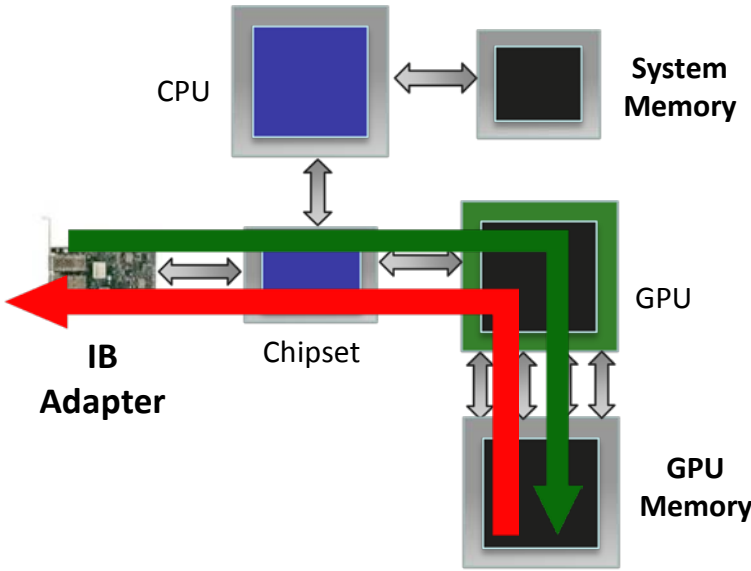
MVAPICH2-GDR 2.0b is available for users

GPUDirect RDMA (GDR) with CUDA

- Hybrid design using GPUDirect 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

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, Int'l Conference on Parallel Processing (ICPP '13)

SNB E5-2670 / IVB E5-2680V2



SNB E5-2670

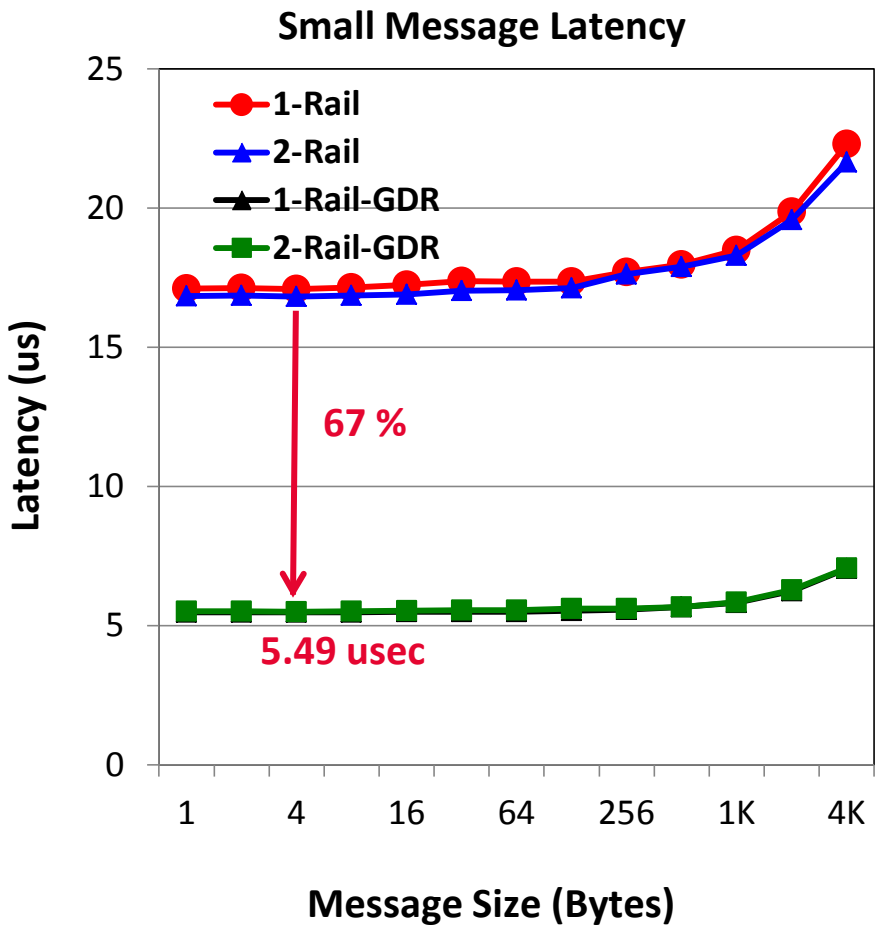
P2P write: 5.2 GB/s
P2P read: < 1.0 GB/s

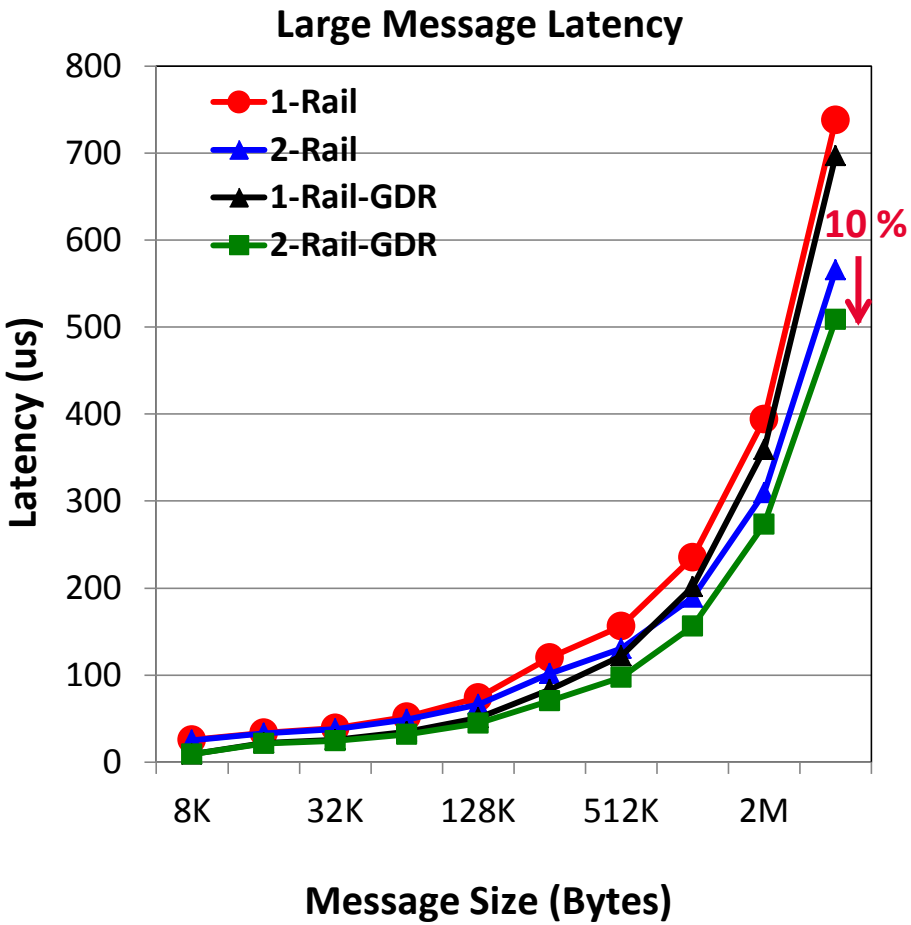
IVB E5-2680V2

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

Performance of MVAPICH2 with GPUDirect-RDMA: Latency

GPU-GPU Internode MPI Latency

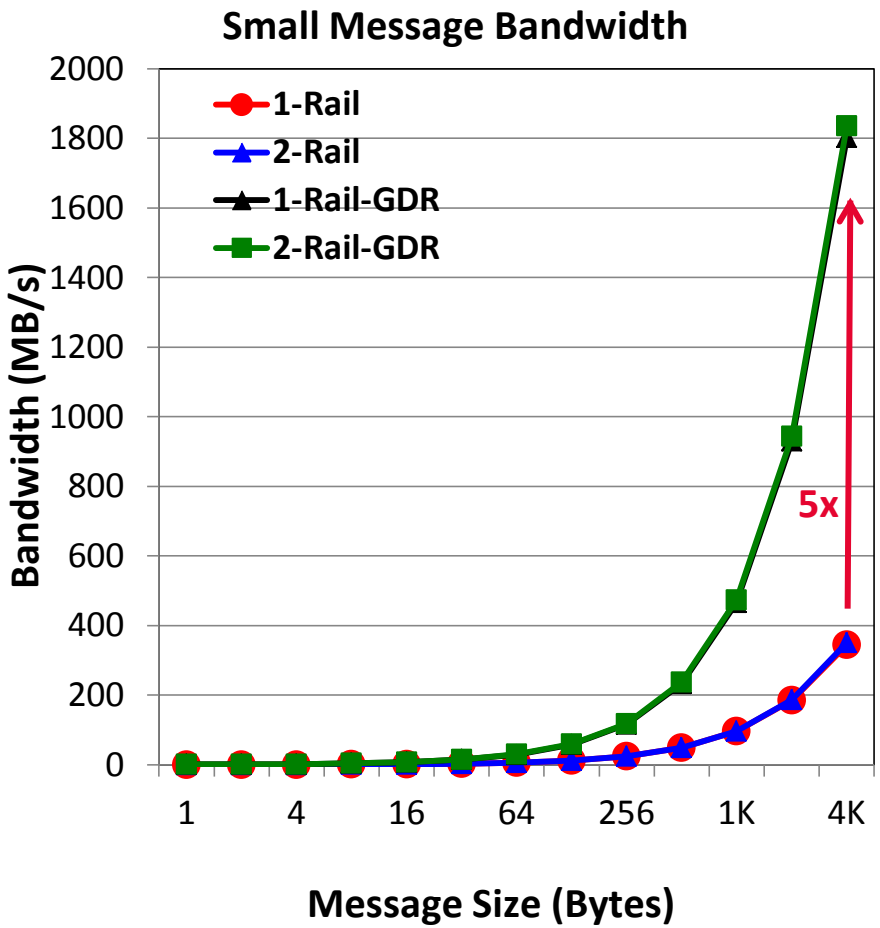


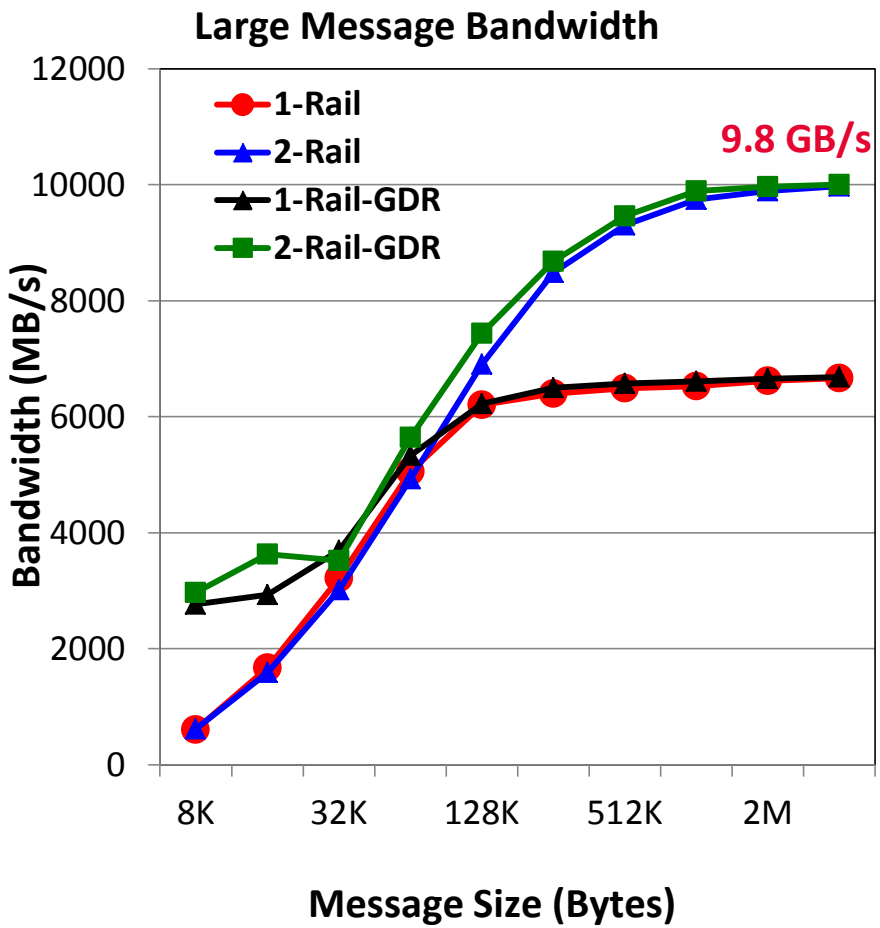


Based on MVAPICH2-2.0b
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Patch

Performance of MVAPICH2 with GPUDirect-RDMA: Bandwidth

GPU-GPU Internode MPI Uni-Directional Bandwidth

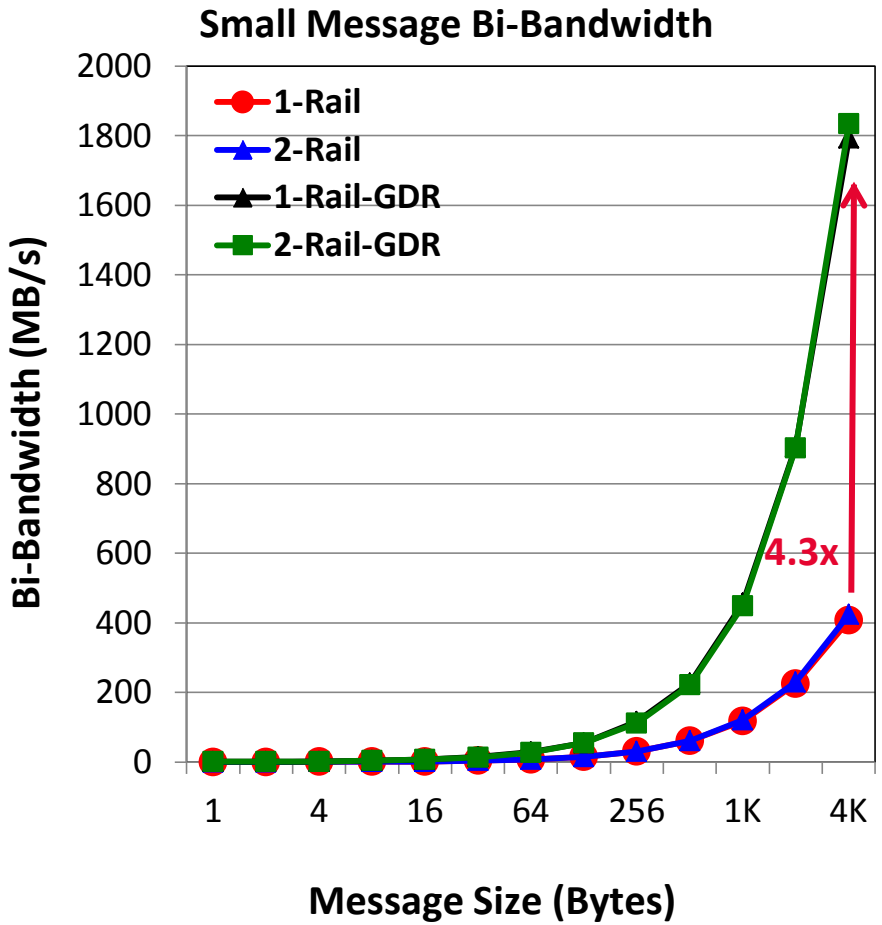


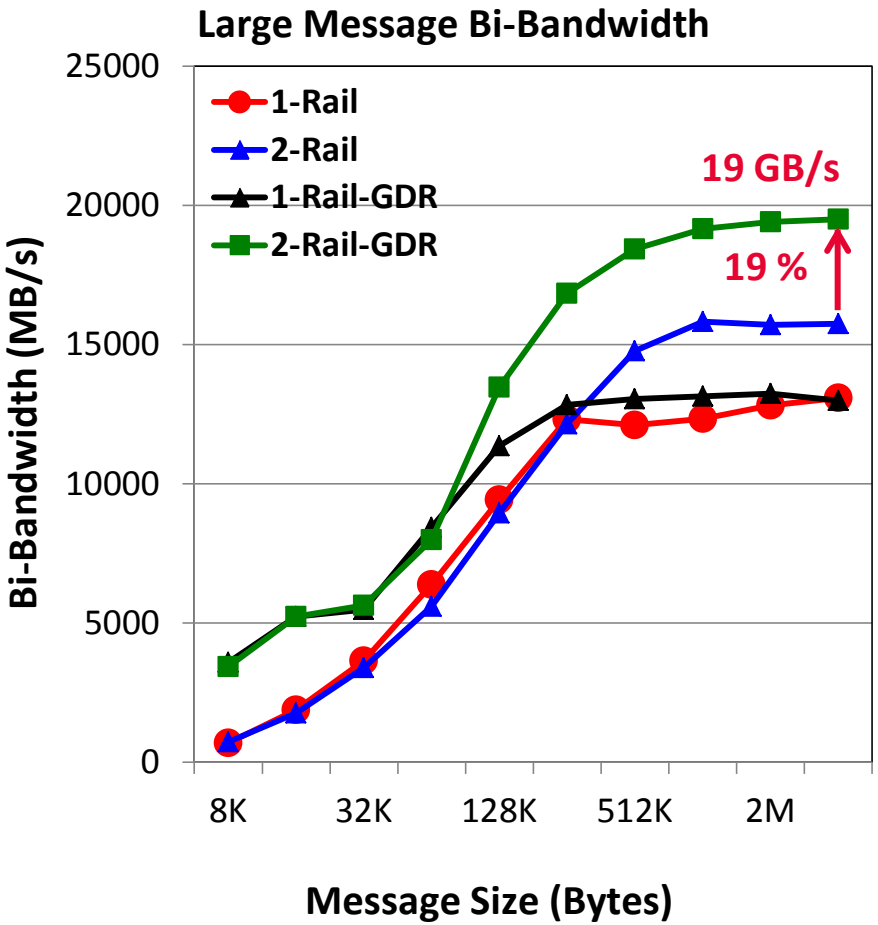


Based on MVAPICH2-2.0b
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Patch

Performance of MVAPICH2 with GPUDirect-RDMA: Bi-Bandwidth

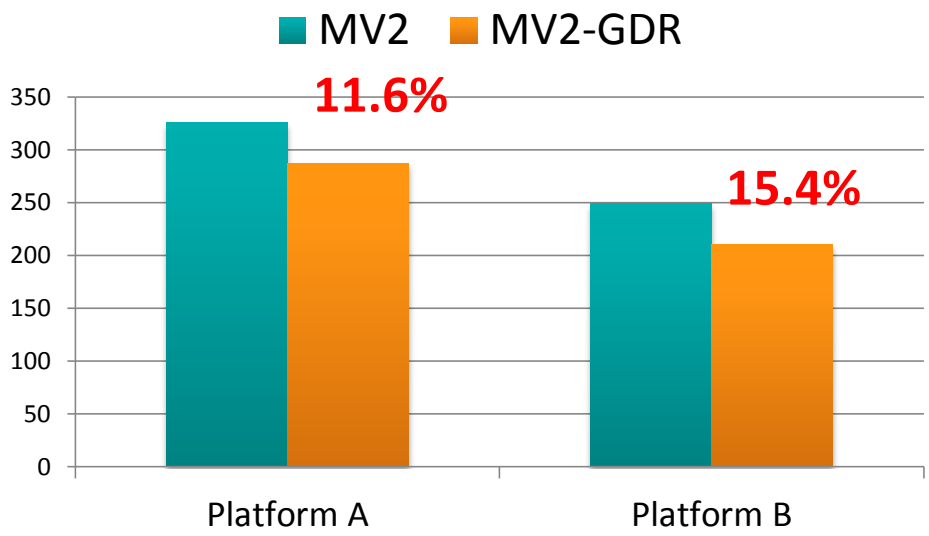
GPU-GPU Internode MPI Bi-directional Bandwidth

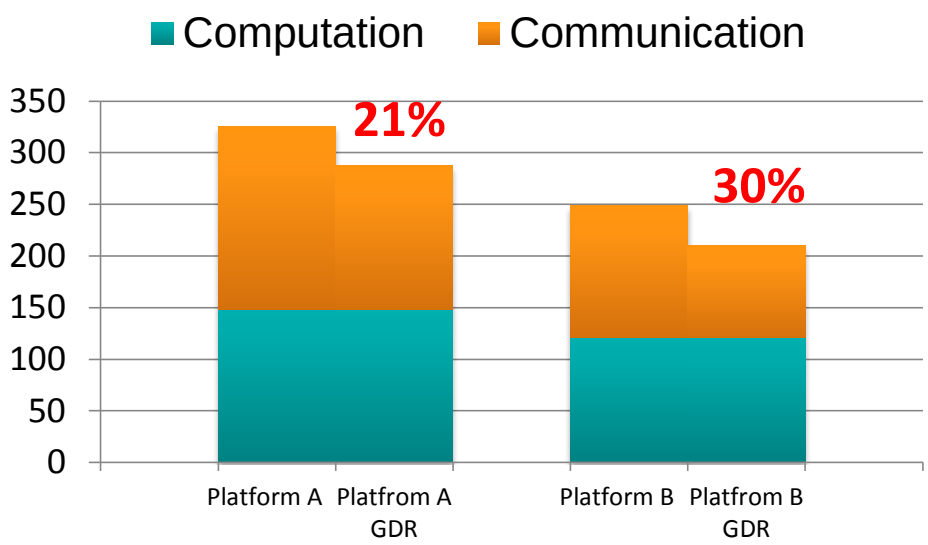




Based on MVAPICH2-2.0b
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Patch

Applications-level Benefits: AWP-ODC with MVAPICH2-GPU





Platform A: Intel Sandy Bridge + NVIDIA Tesla K20 + Mellanox ConnectX-3

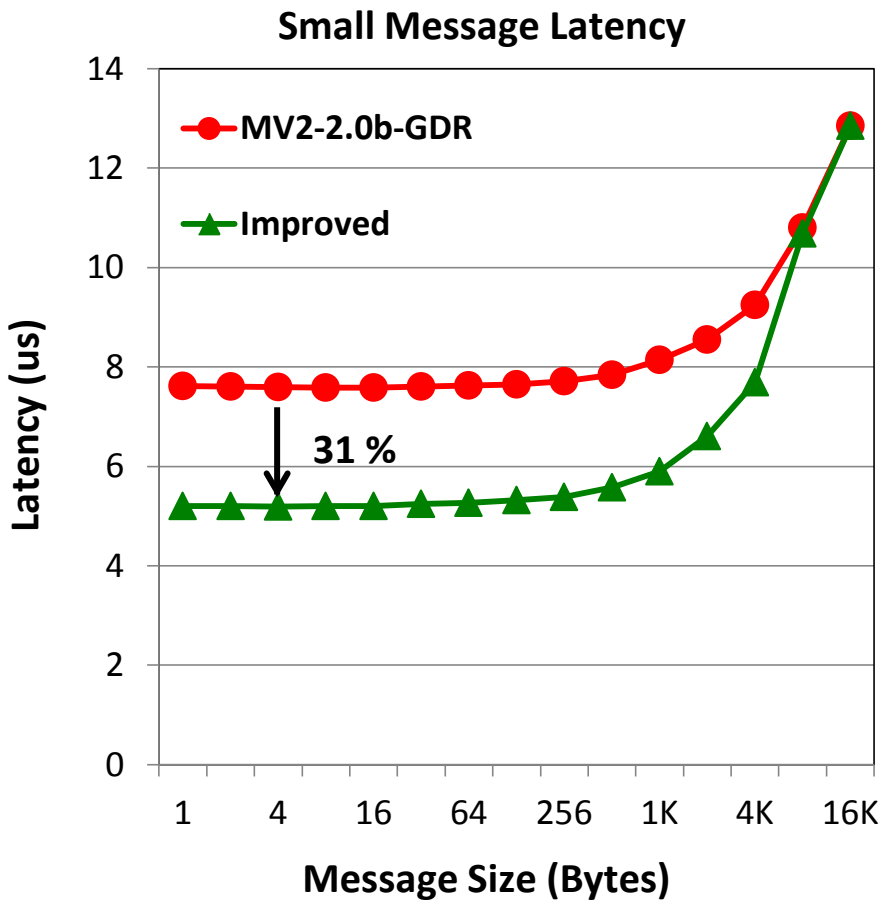
Platform B: Intel Ivy Bridge + NVIDIA Tesla K40 + Mellanox Connect-IB

- A widely-used seismic modeling application, Gordon Bell Finalist at SC 2010
- An initial version using MPI + CUDA for GPU clusters
- Takes advantage of CUDA-aware MPI, two nodes, 1 GPU/Node and 64x32x32 problem
- GPUDirect-RDMA delivers better performance with newer architecture

Based on MVAPICH2-2.0b, CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Patch
Two nodes, one GPU/node, one Process/GPU

Continuous Enhancements for Improved Point-to-point Performance

GPU-GPU Internode MPI Latency

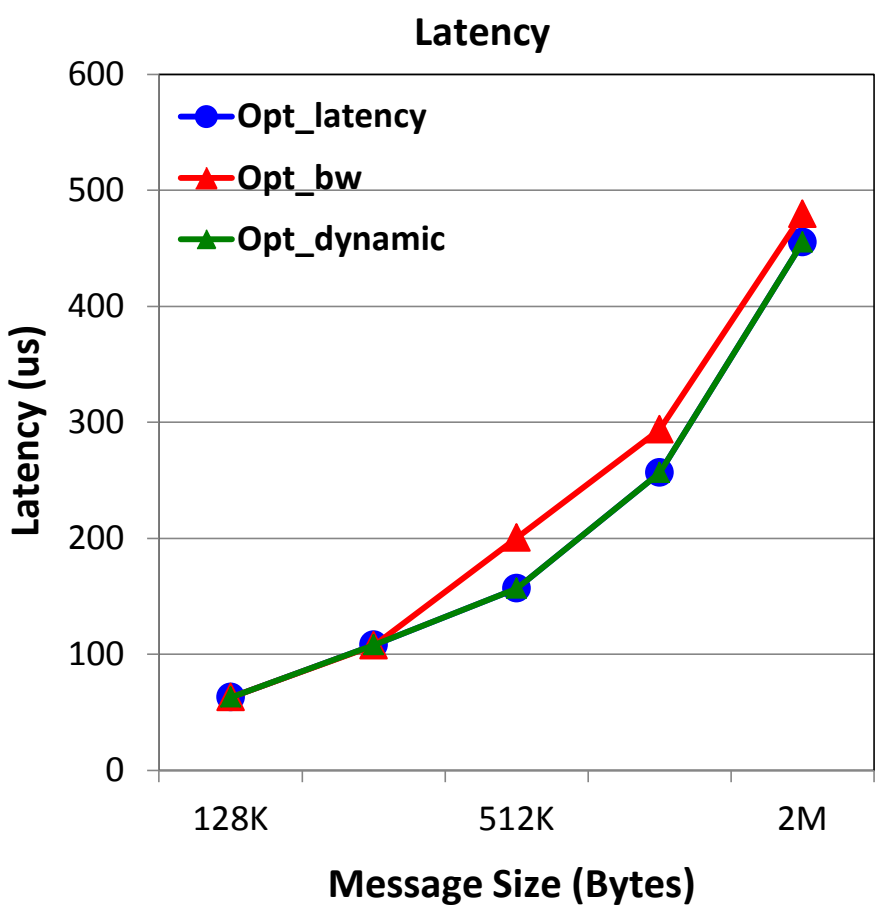


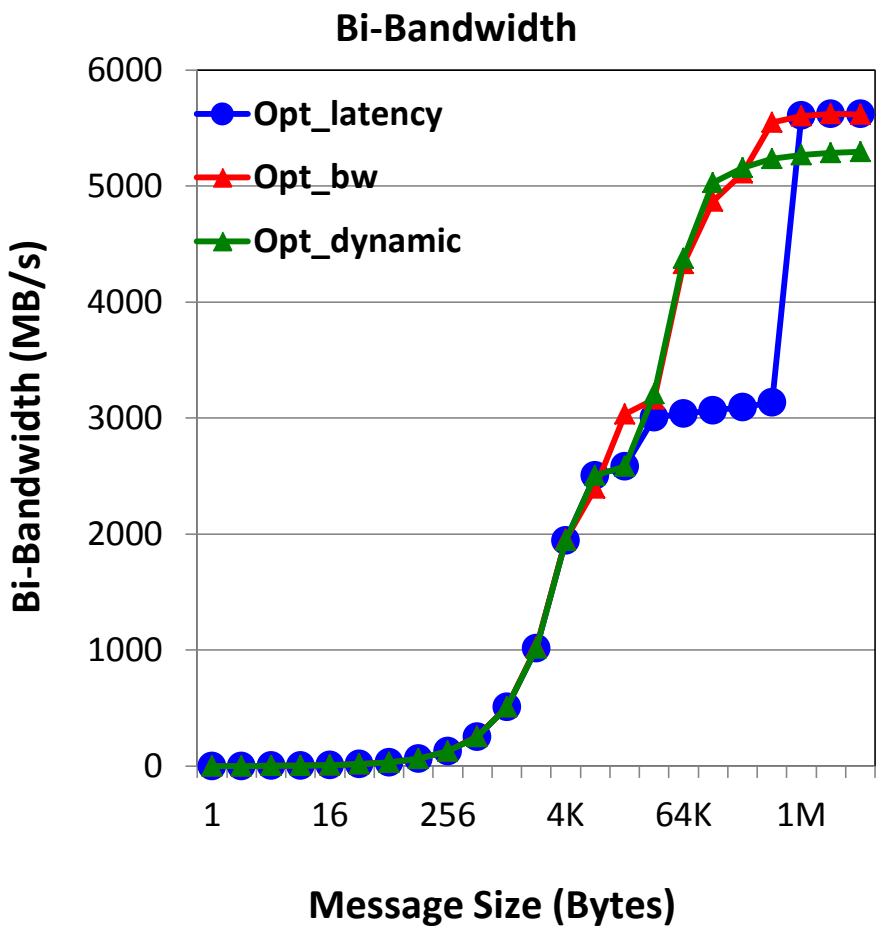
- Reduced synchronization and while avoiding expensive copies

Based on MVAPICH2-2.0b + enhancements
Intel Ivy Bridge (E5-2630 v2) node with 12 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Patch

Dynamic Tuning for Point-to-point Performance

GPU-GPU Internode MPI Performance





Based on MVAPICH2-2.0b + enhancements
Intel Ivy Bridge (E5-2630 v2) node with 12 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Patch

Presentation Overview

- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
- MVAPICH2 and OpenACC
- Programming Models for Intel MIC
- MVAPICH2-MIC Designs

One-sided communication

- Send/Recv semantics incur overheads
  - Distributed buffer information
  - Message matching
  - Additional copies or rendezvous exchange

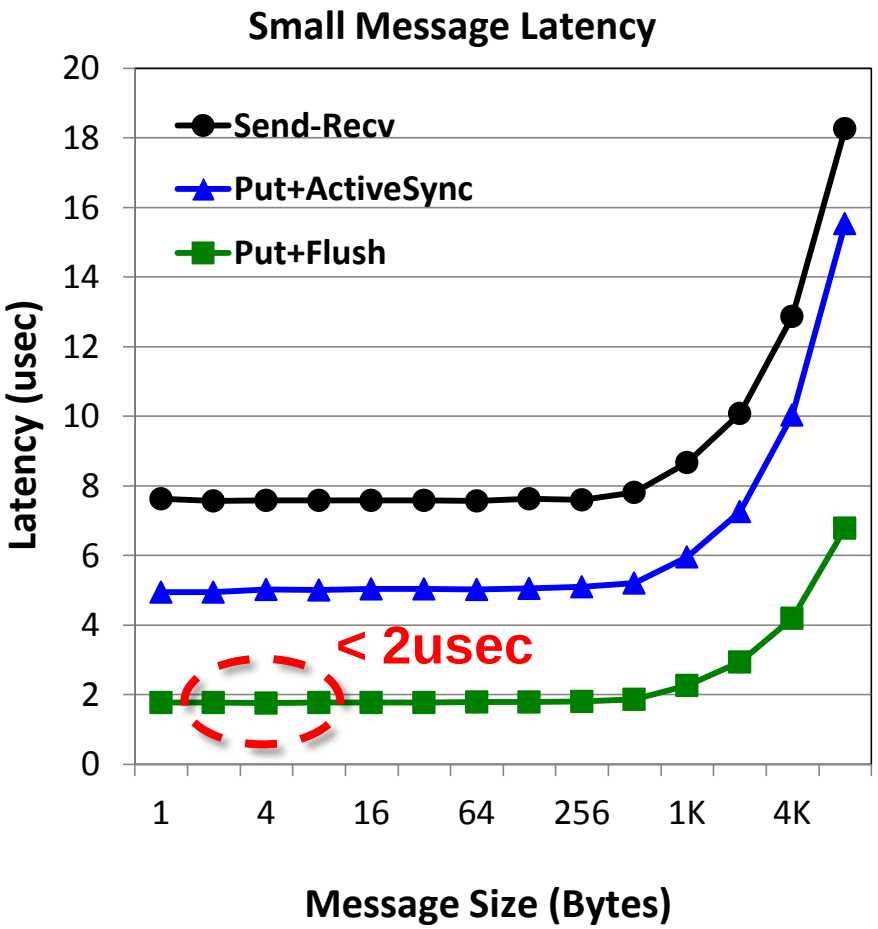
4 bytes	Host-Host	GPU-GPU
IB send/recv	0.98	1.84
MPI send/recv	1.25	6.95

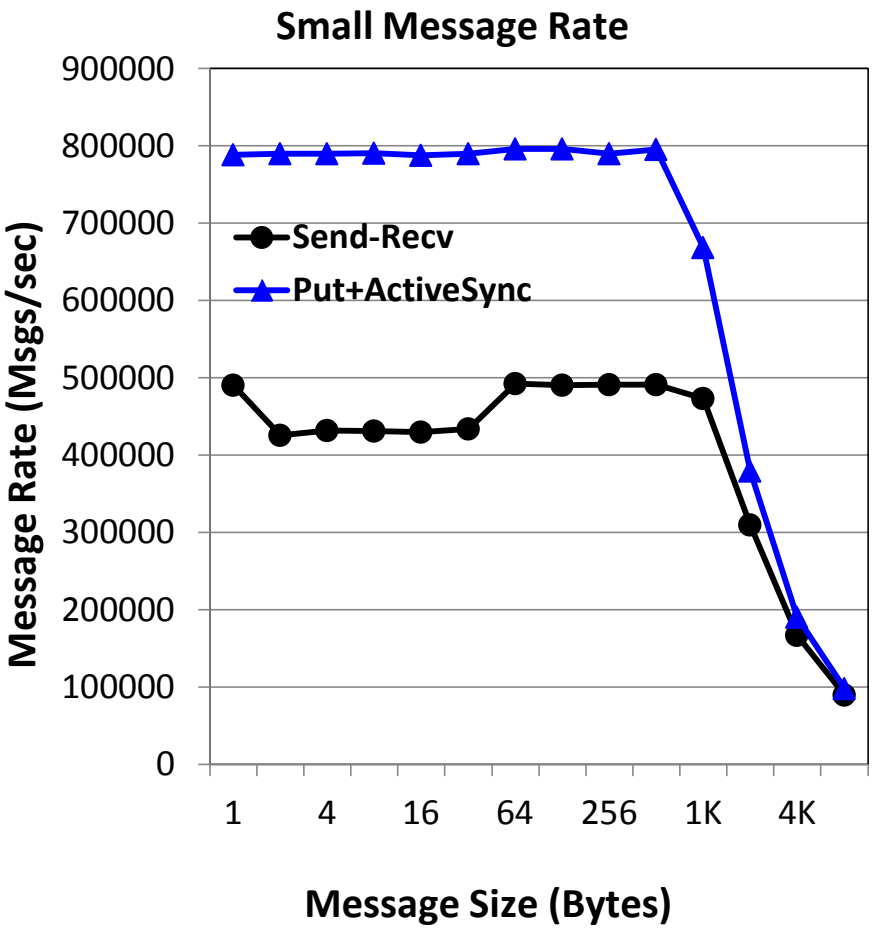
Table: Latency (half round trip) on SandyBridge nodes with FDR connect-IB

- One-sided communication
  - Separates synchronization from communication
  - Direct mapping over RDMA semantics
  - Lower overheads and better overlap

MPI-3 RMA Support with GPUDirect RDMA

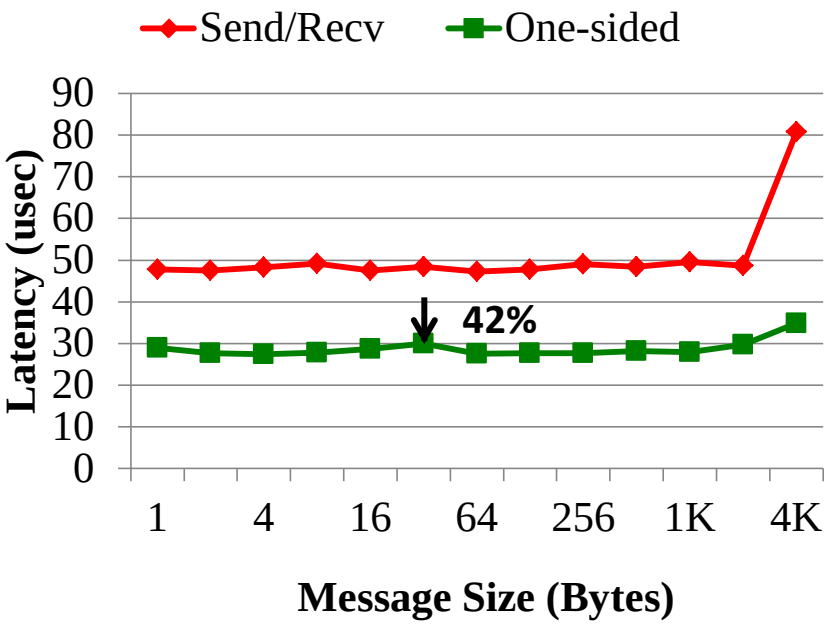
MPI-3 RMA provides flexible synchronization and completion primitives



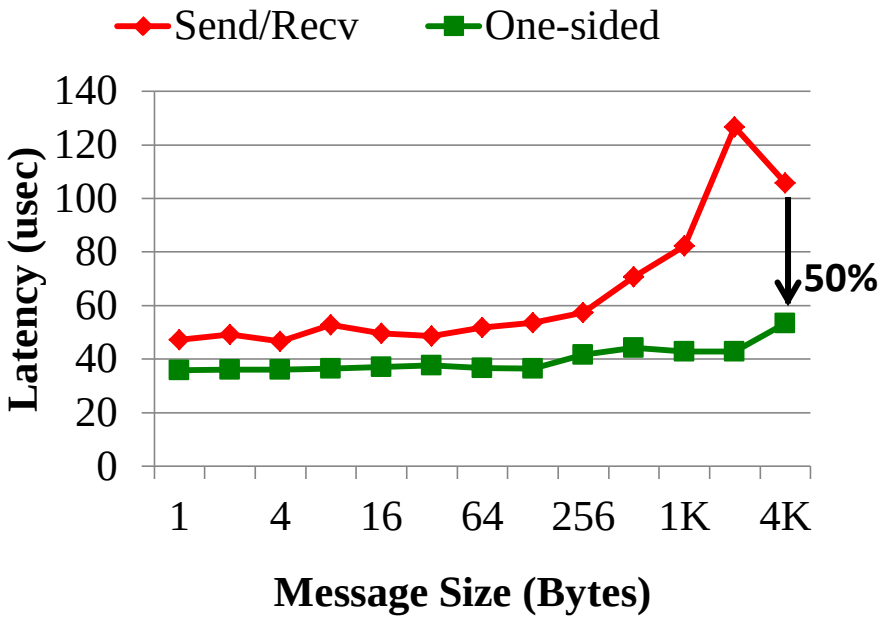


Based on MVAPICH2-2.0b + Extensions
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.1 with GPUDirect-RDMA Plugin

Communication Kernel Evaluation: 3Dstencil and Alltoall



3D Stencil with 16 GPU nodes



AlltoAll with 16 GPU nodes

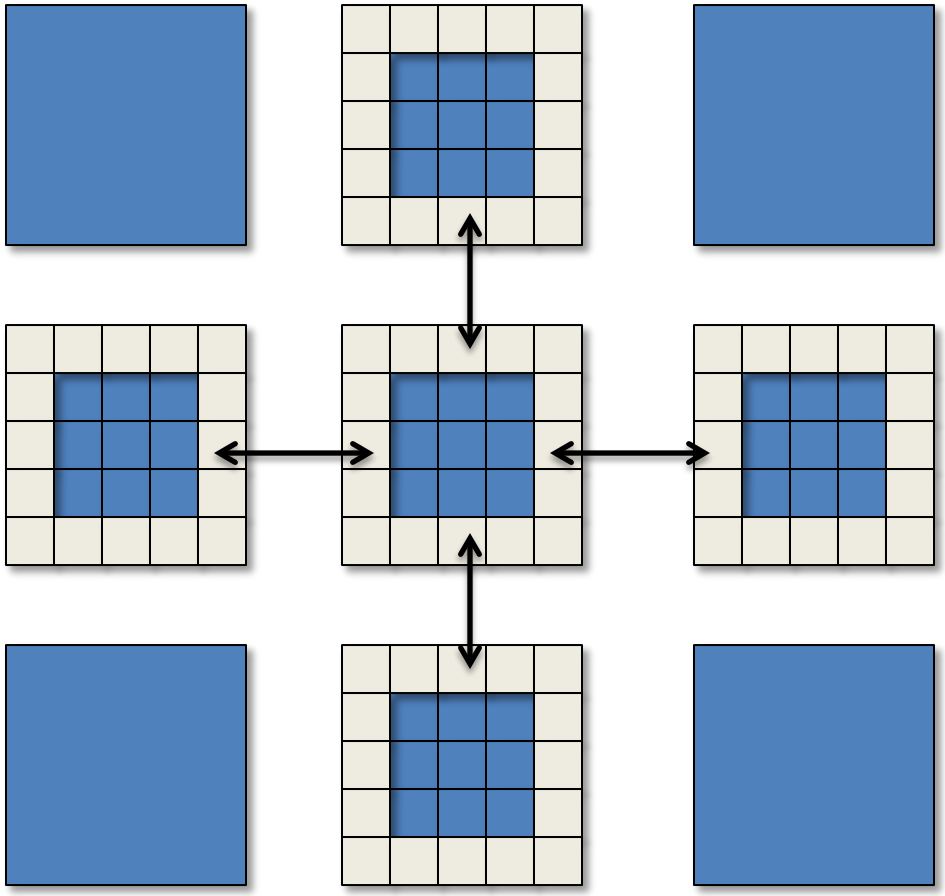
Based on MVAPICH2-2.0b + Extensions
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.1 with GPUDirect-RDMA Plugin

Presentation Overview

- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
- MVAPICH2 and OpenACC
- Programming Models for Intel MIC
- MVAPICH2-MIC Designs

Non-contiguous Data Exchange

Halo data exchange

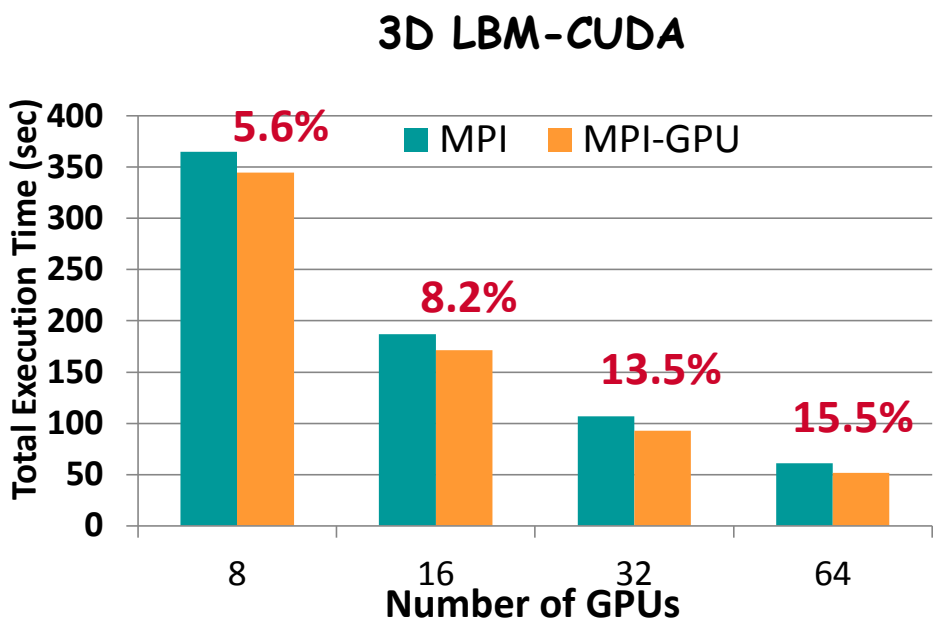


- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions
- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration

MPI Datatype Processing

- 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

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

How can I get Started with GDR Experimentation?

- MVAPICH2-2.0b with GDR support can be downloaded from

  https://mvapich.cse.ohio-state.edu/download/mvapich2gdr/
- System software requirements
  - Mellanox OFED 2.1
  - NVIDIA Driver 331.20 or later
  - NVIDIA CUDA Toolkit 5.5
  - Plugin for GPUDirect RDMA

    (http://www.mellanox.com/page/products_dyn?product_family=116)
- Has optimized designs for point-to-point communication using GDR
- Work under progress for optimizing collective and one-sided communication
- Contact MVAPICH help list with any questions related to the package mvapich-help@cse.ohio-state.edu
- MVAPICH2-GDR-RC1 with additional optimizations coming soon!!

Presentation Overview

- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
- MVAPICH2 and OpenACC
- Programming Models with Intel MIC
- MVAPICH2-MIC Designs

OpenACC

- OpenACC is gaining popularity
- Multiple sessions during GTC '14
- 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 MVPICH2 with the new OpenACC 2.0

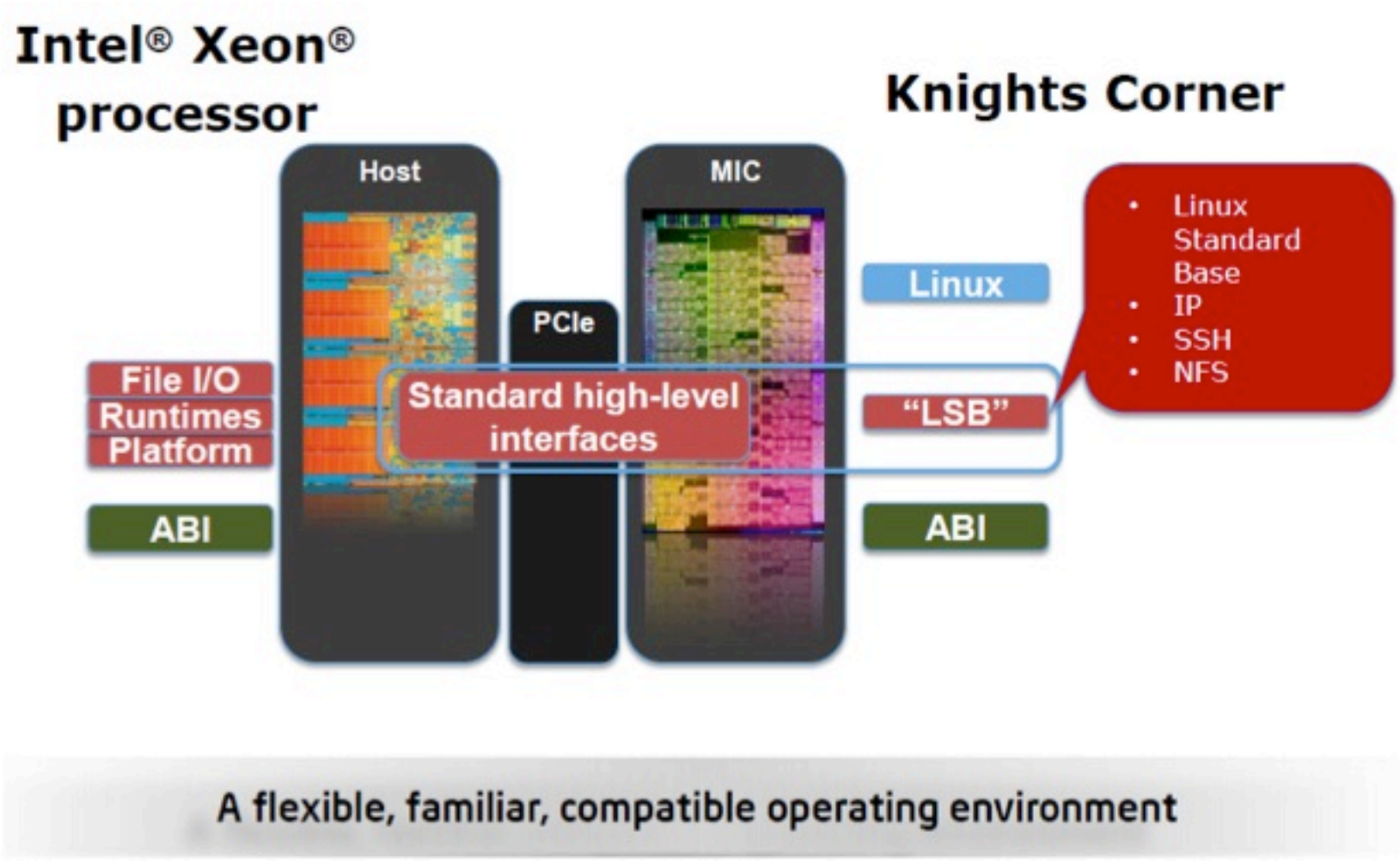
- acc_deviceptr to get device pointer (in OpenACC 2.0)
  - Enables MPI communication from memory allocated by compiler when it is available in OpenACC 2.0 implementations
  - MVAPICH2 will detect the device pointer and optimize communication
  - Delivers the same performance as with CUDA

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

......

#pragma acc data copyin(A)
{

#pragma acc parallel for
//compute for  loop

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

}
......
free(A);
```

Presentation Overview

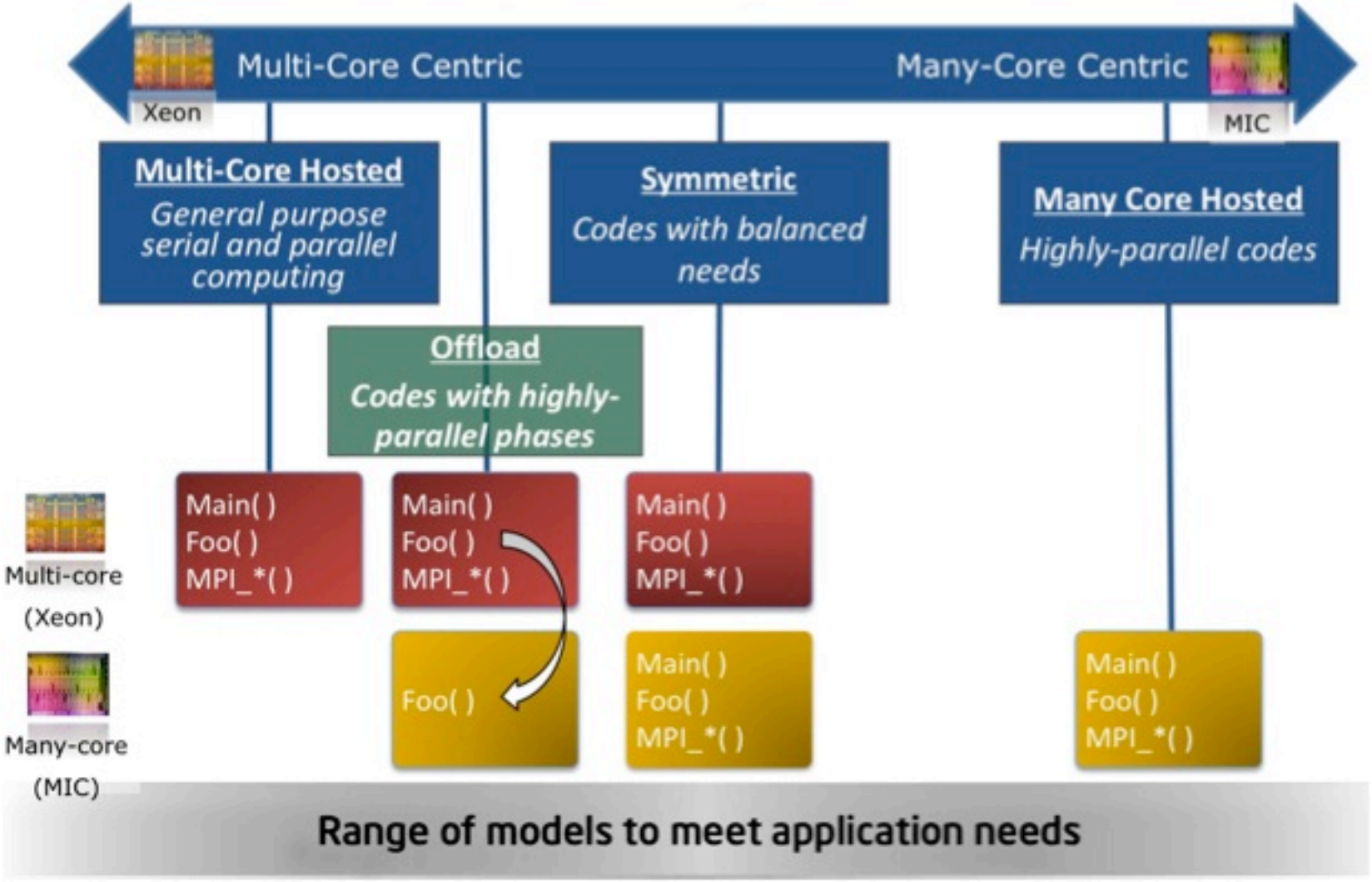
- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
  - Two-sided Communication
  - One-sided Communication
  - MPI Datatype Processing
- MVAPICH2 and OpenACC
- Programming Models for Intel MIC
- MVAPICH2-MIC Designs

InfiniBand + MIC systems



Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

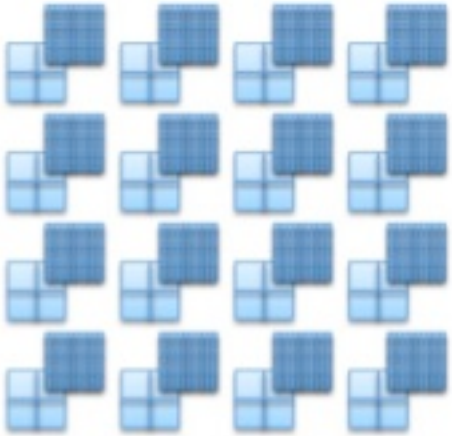
Programming Models for Intel MIC Architecture

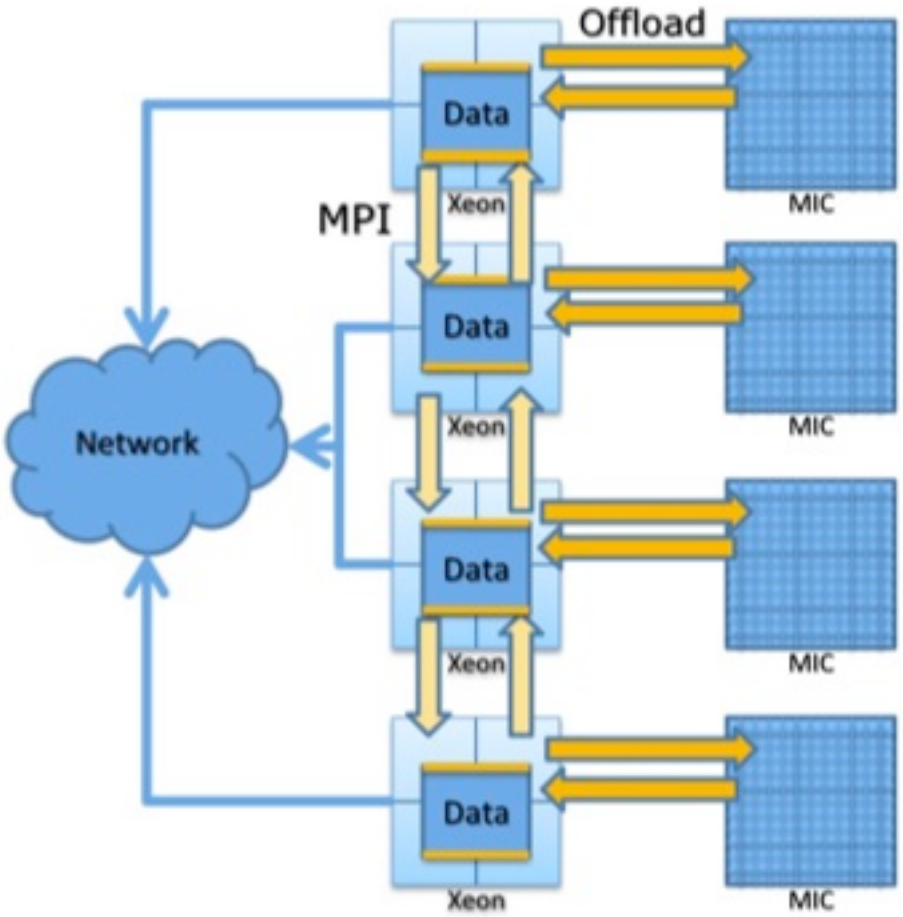


Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Offload Model for Intel MIC-based Systems

- MPI processes on the host
- Intel MIC used through offload – OpenMP, Intel Cilk Plus, Intel TBB and Pthreads
- MPI communication – Intranode and Internode

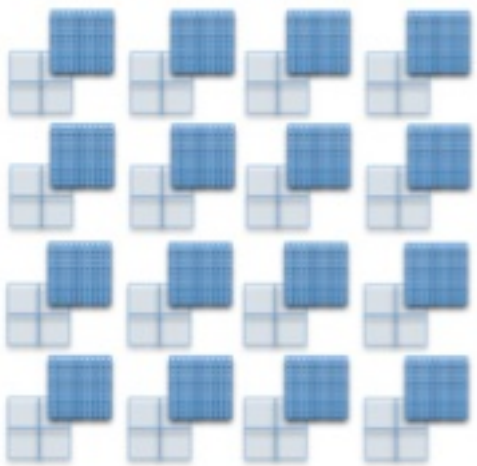


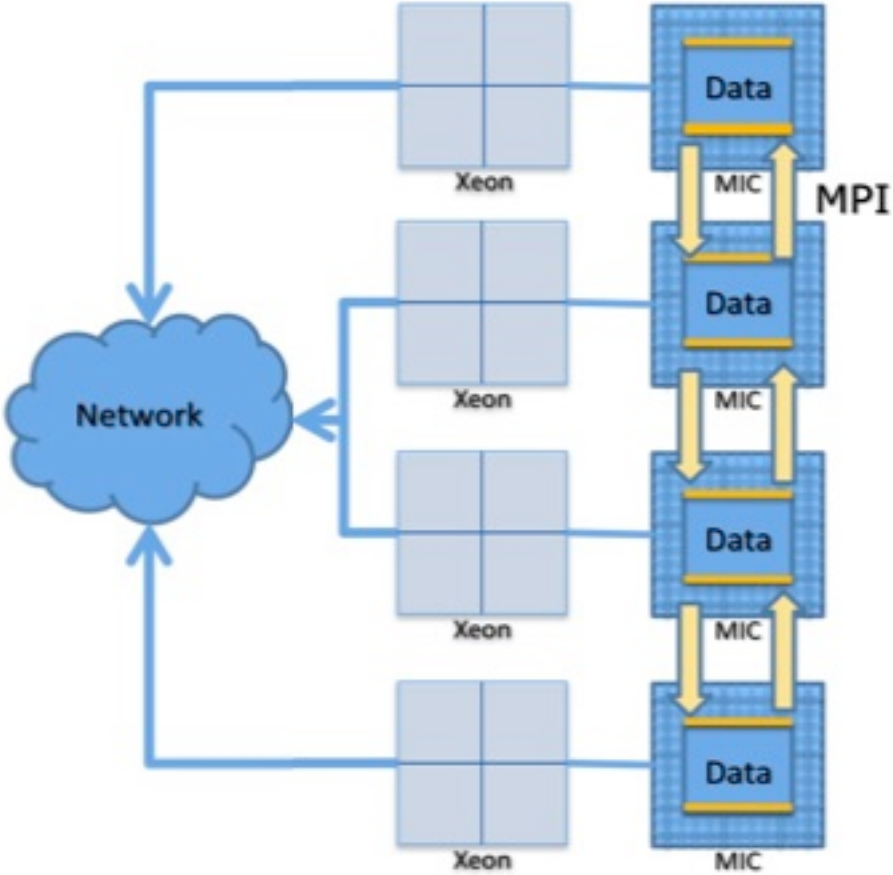


Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Many-core Hosted Model for Intel MIC-based Systems

- MPI processes on the Intel MIC
- MPI communication – IntraMIC, InterMIC (host-bypass/directly over the network)

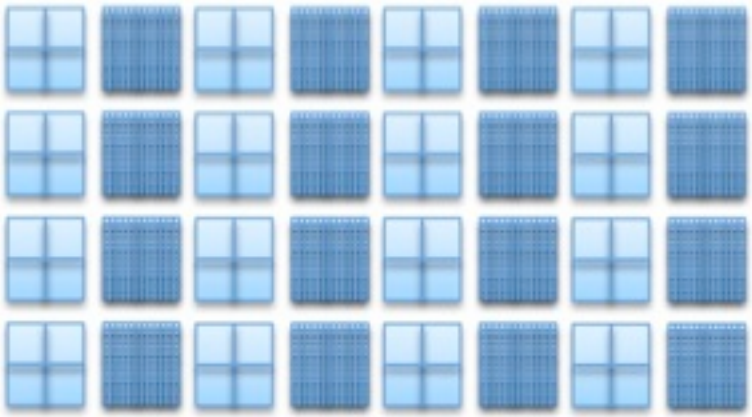


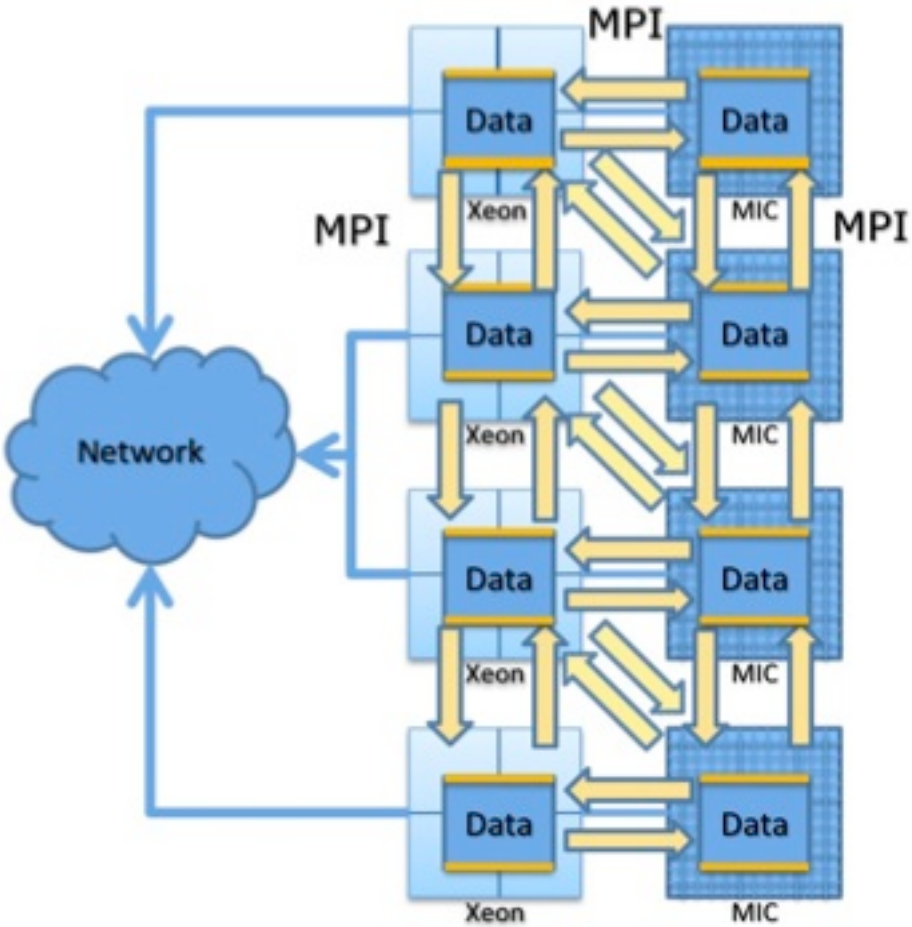


Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Symmetric Model for Intel MIC-based Systems

- MPI processes on the host and the Intel MIC
- MPI communication – Intranode, Internode, IntraMIC, InterMIC, Host-MIC





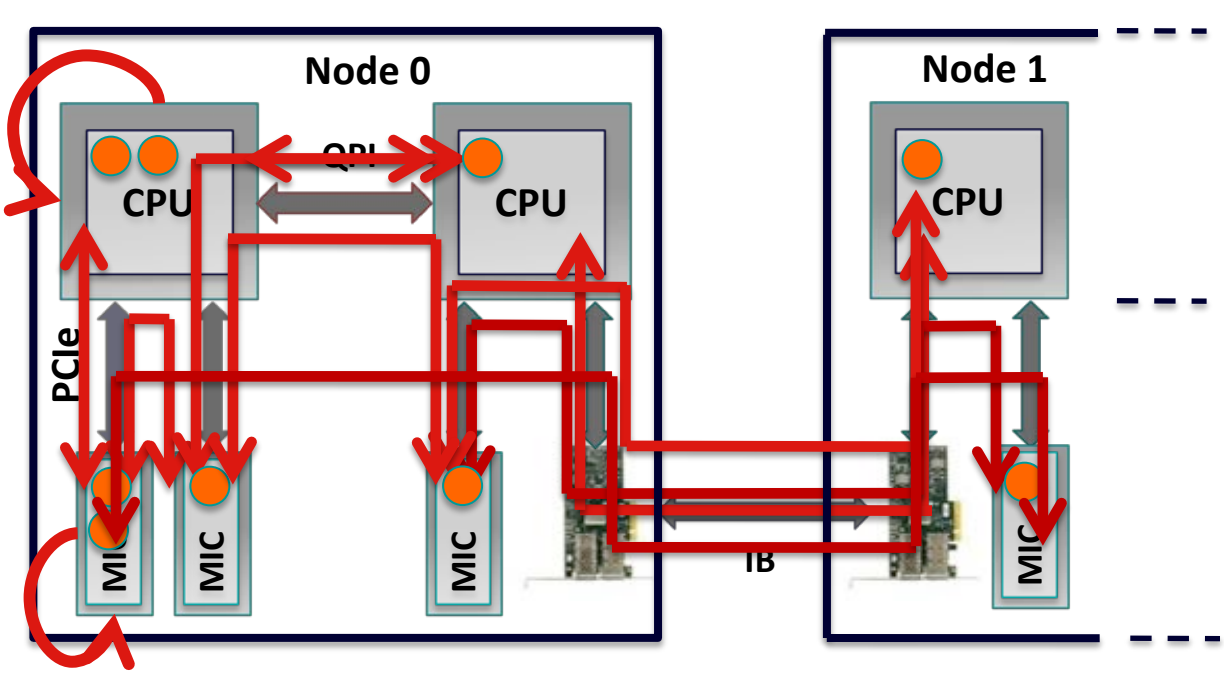
Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Presentation Overview

- CUDA-Aware MPI (MVAPICH2-GPU)
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- MVAPICH2 and OpenACC
- Programming Models for Intel MIC
- MVAPICH2-MIC Designs

Data Movement on Intel Xeon Phi Clusters

- Connected as PCIe devices – Flexibility but Complexity



MPI Process

1. Intra-Socket
2. Inter-Socket
3. Inter-Node
4. Intra-MIC
5. Intra-Socket MIC-MIC
6. Inter-Socket MIC-MIC
7. Inter-Node MIC-MIC
8. Intra-Socket MIC-Host
9. Inter-Socket MIC-Host
10. Inter-Node MIC-Host

11. Inter-Node MIC-MIC with IB adapter on remote socket and more . . .

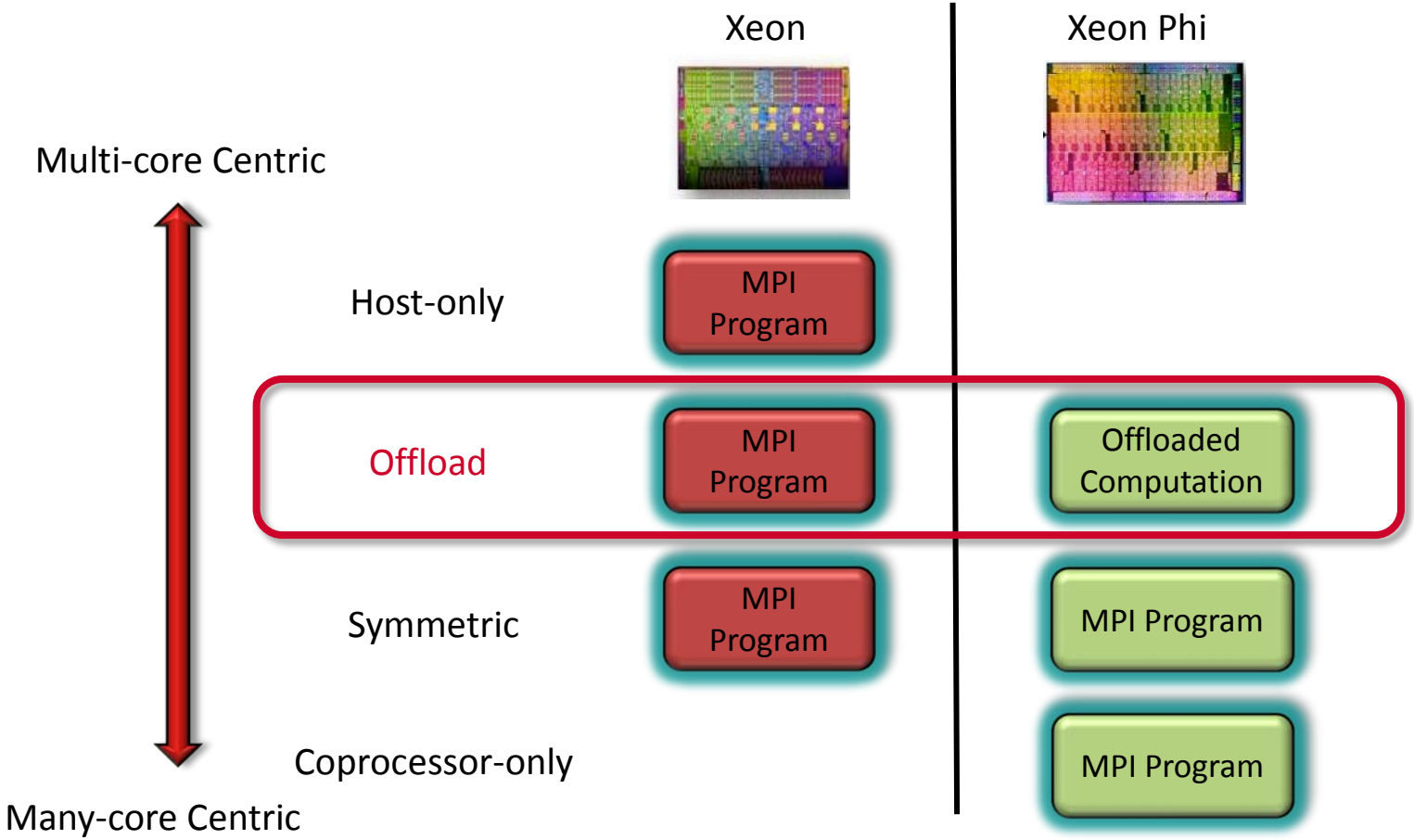
- Critical for runtimes to optimize data movement, hiding the complexity

Efficient MPI Communication on Xeon Phi Clusters

- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only Mode
  - Symmetric Mode
- Comparison with Intel's MPI Library

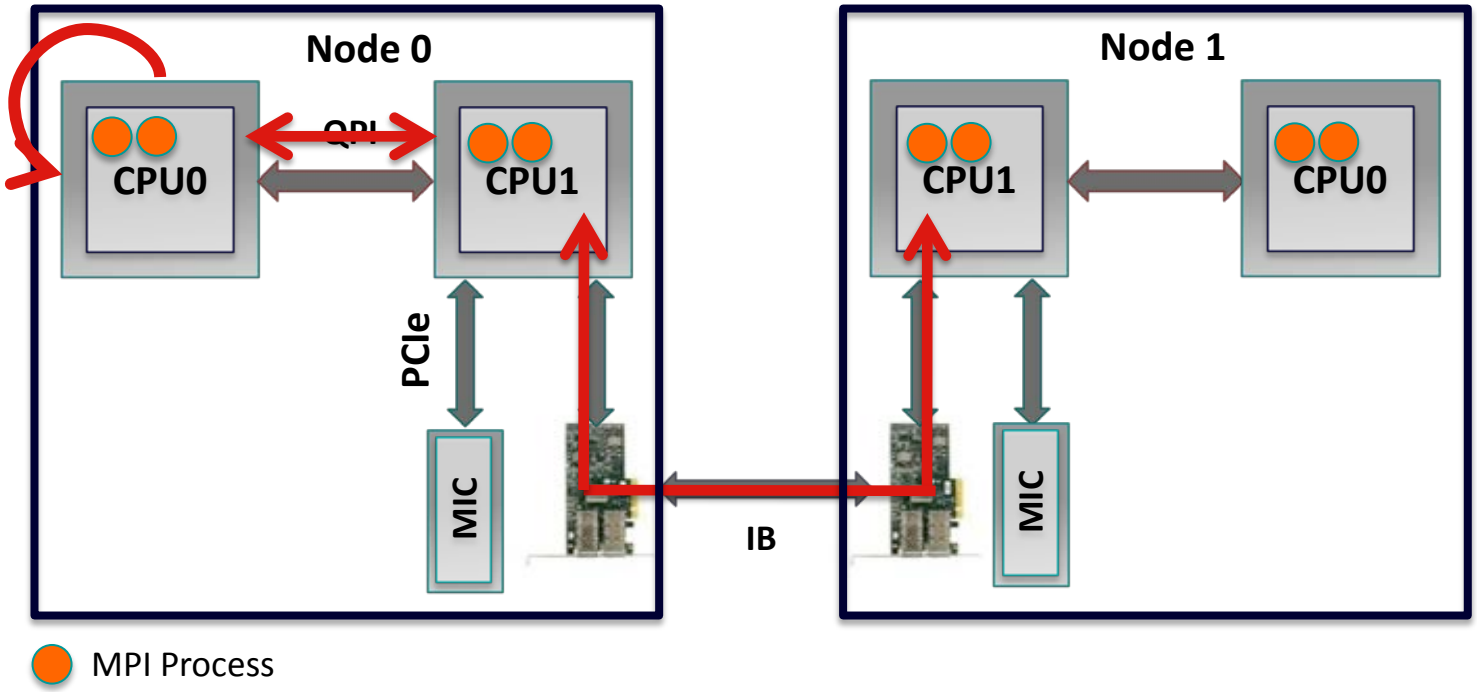
MPI Applications on MIC Clusters

- Offload mode: lesser overhead way to extract performance from Xeon Phi
- MPI processes run only on the host



MPI Data Movement in Offload Mode

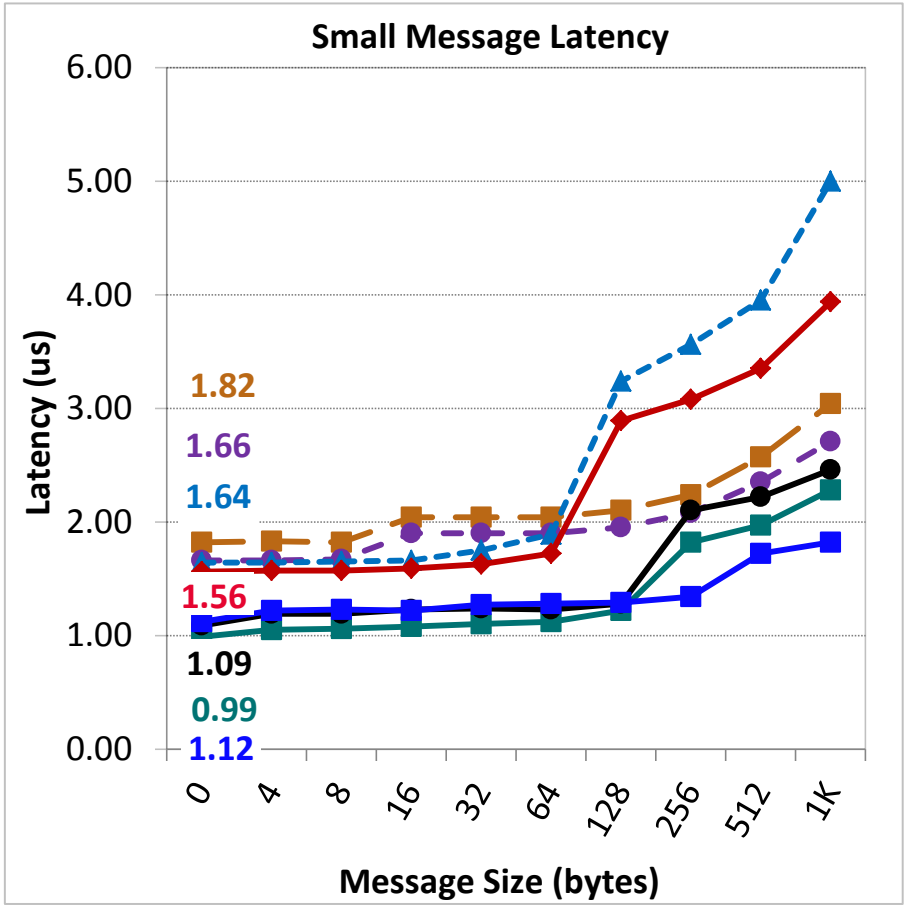
- MPI communication only from the host

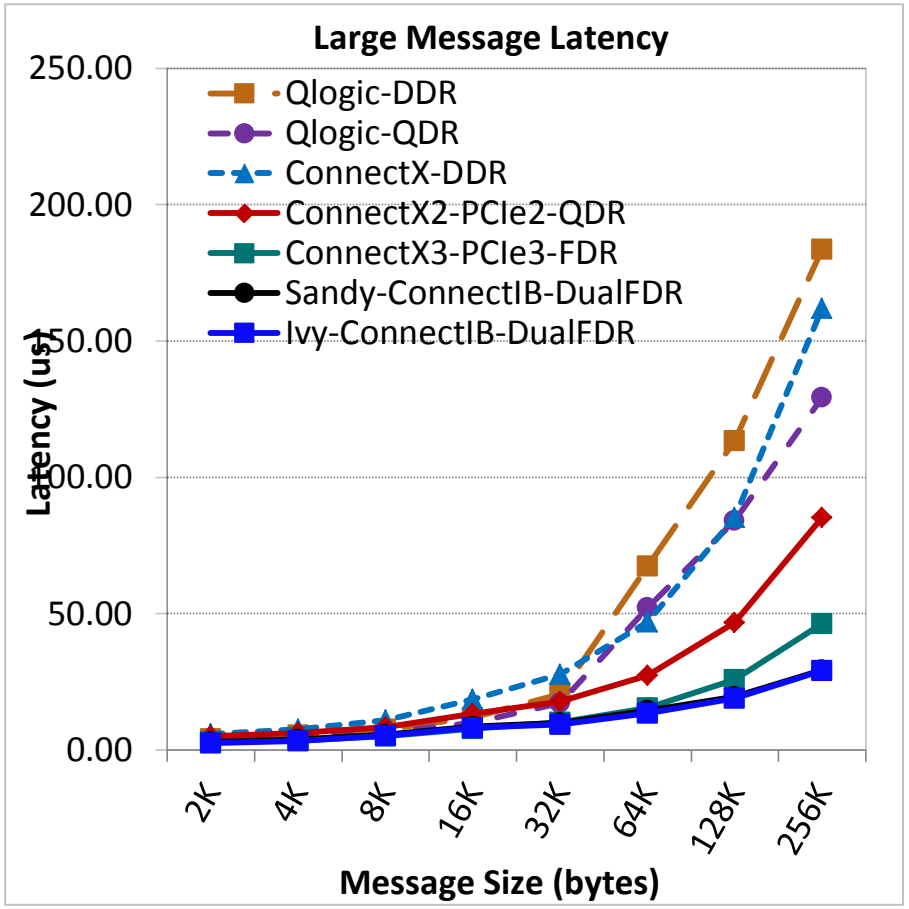


1. Intra-Socket 2. Inter-Socket 3. Inter-Node

- **Existing MPI library like MVAPICH2 1.9 or 2.0rc1 can be used**

One-way Latency: MPI over IB with MVAPICH2





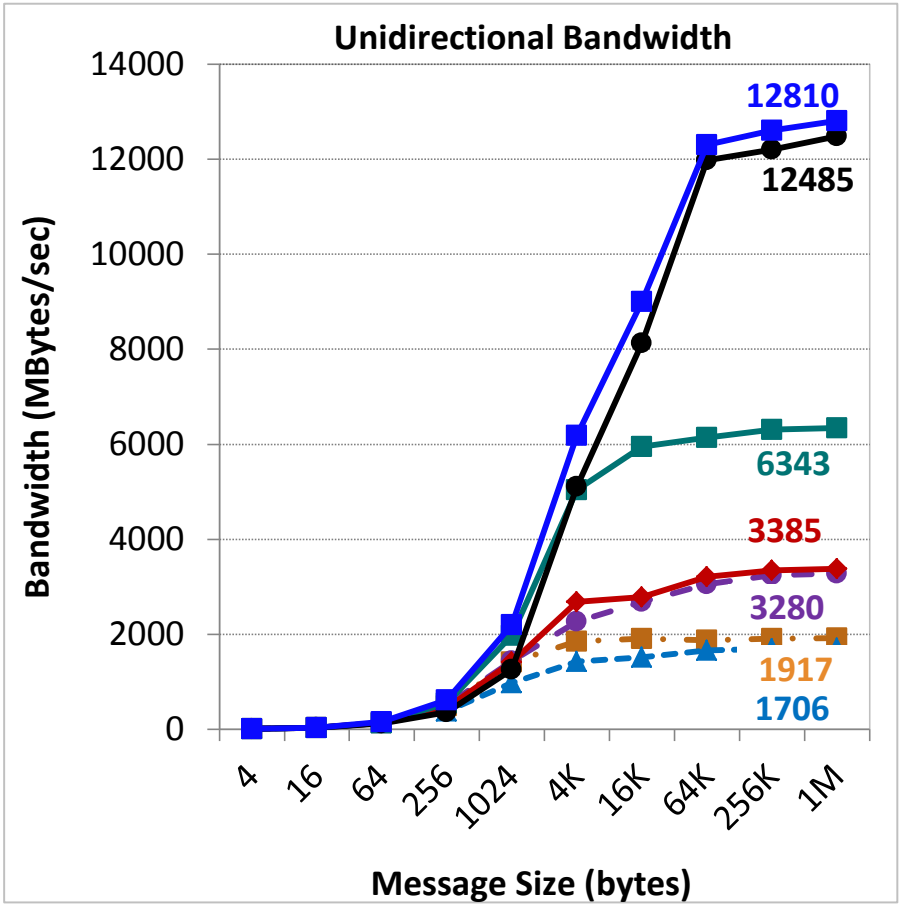
DDR, QDR - 2.4 GHz Quad-core (Westmere) Intel PCI Gen2 with IB switch

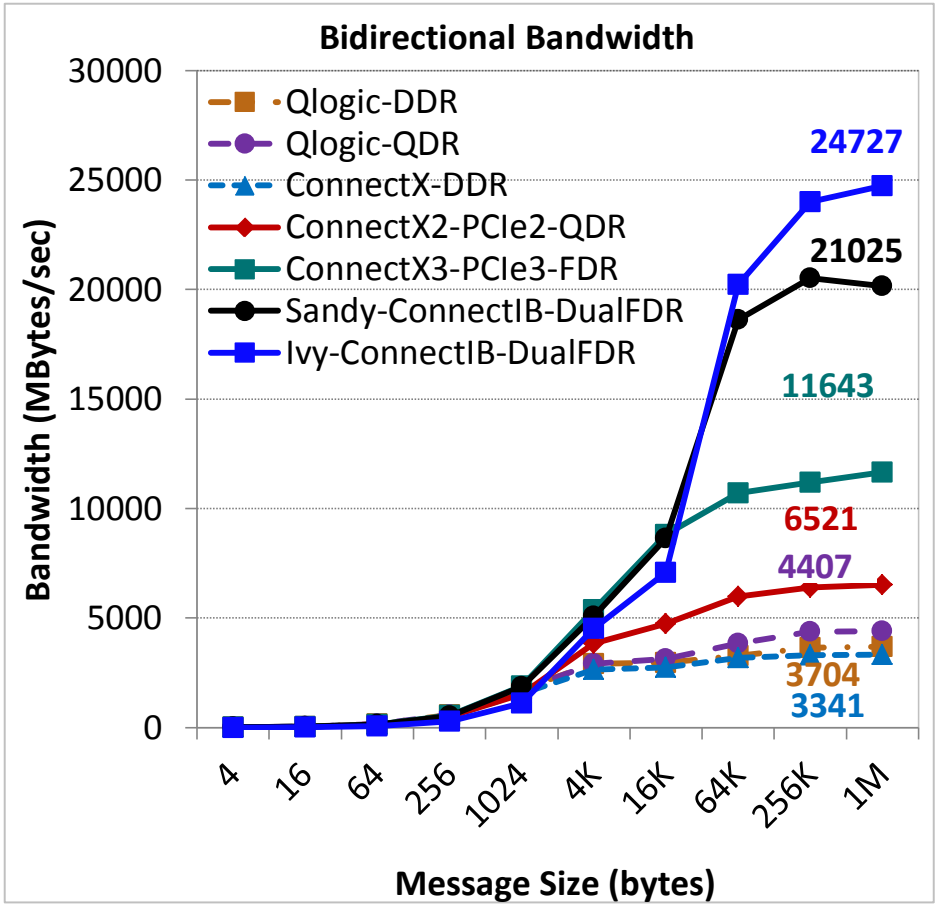
FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch

ConnectIB-Dual FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch

ConnectIB-Dual FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

Bandwidth: MPI over IB with MVAPICH2





DDR, QDR - 2.4 GHz Quad-core (Westmere) Intel PCI Gen2 with IB switch

FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch

ConnectIB-Dual FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch

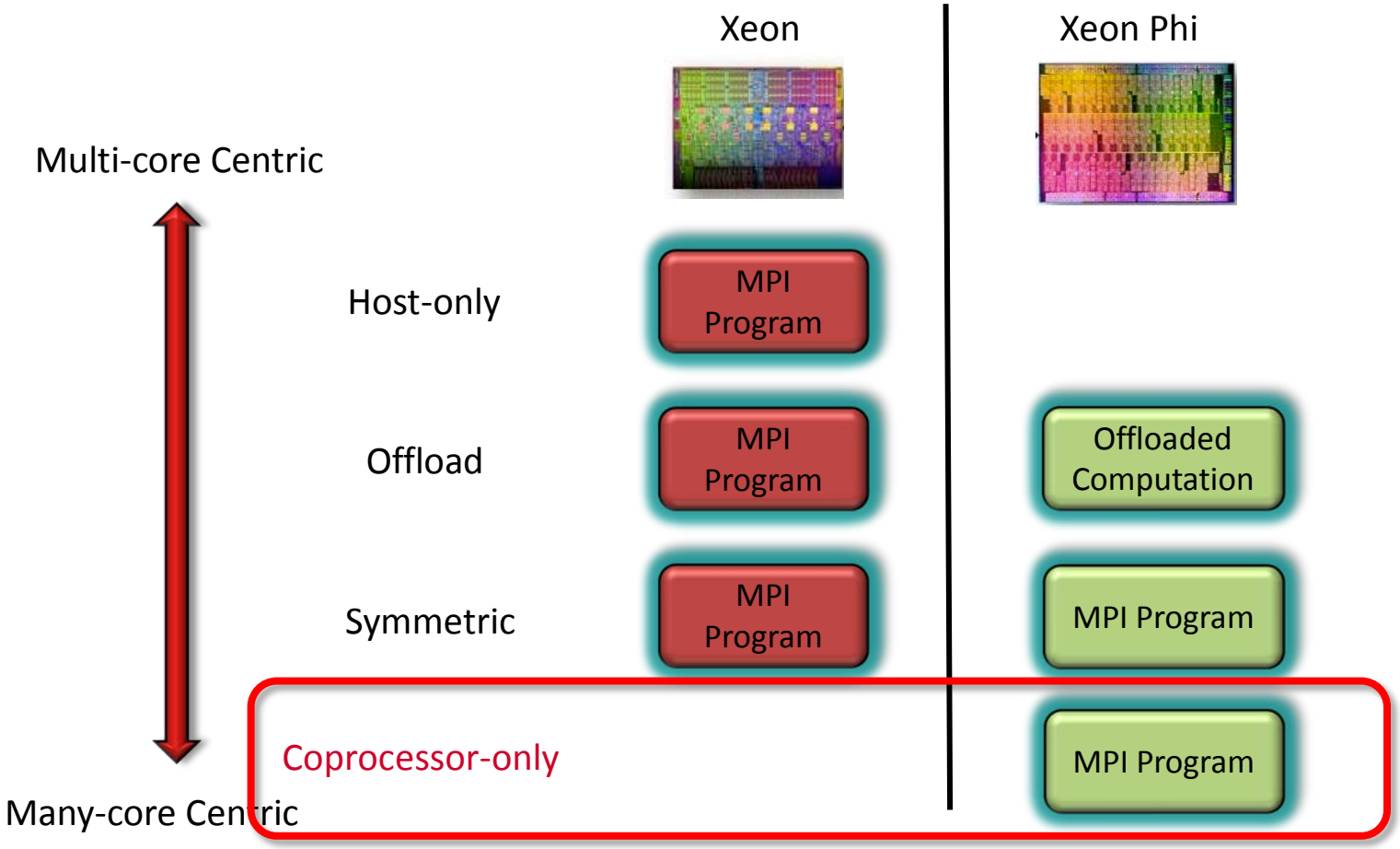
ConnectIB-Dual FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

Efficient MPI Communication on Xeon Phi Clusters

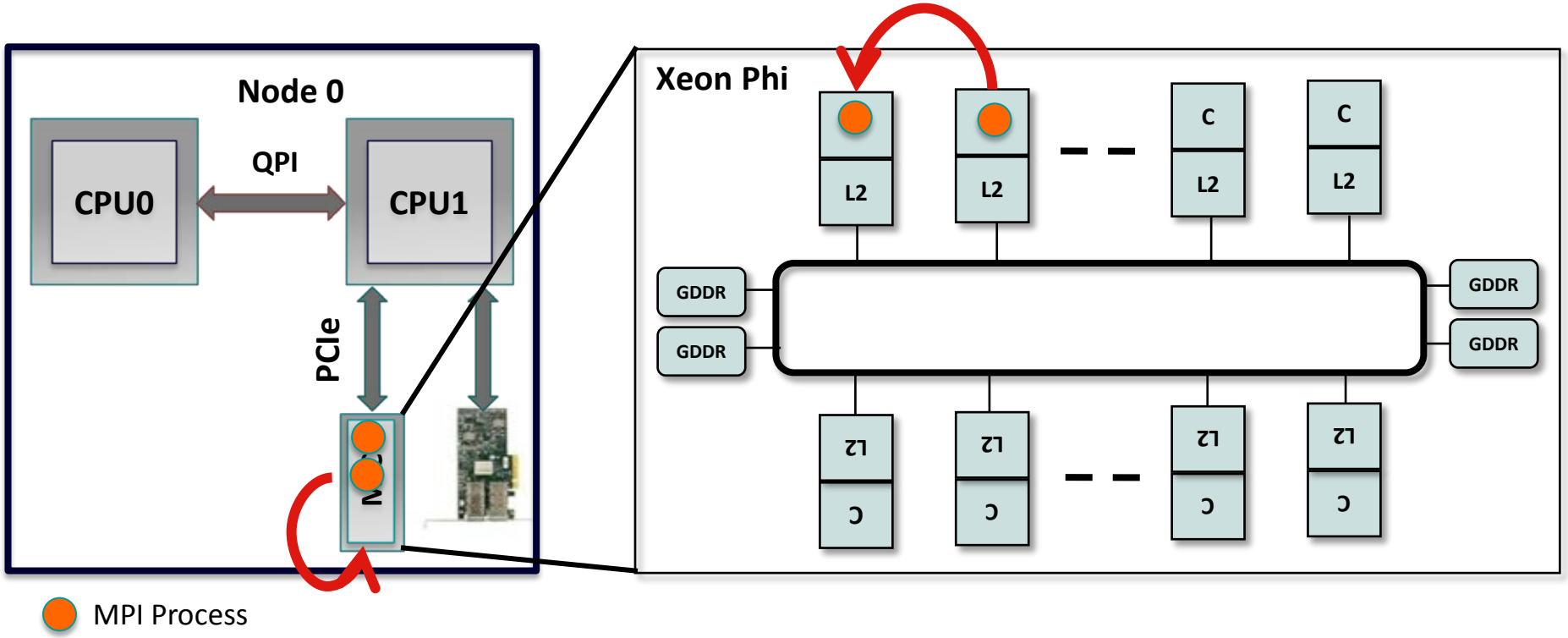
- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only Mode
  - Symmetric Mode
- Comparison with Intel's MPI Library

MPI Applications on Xeon Phi Clusters - IntraNode

- Xeon Phi as a many-core node



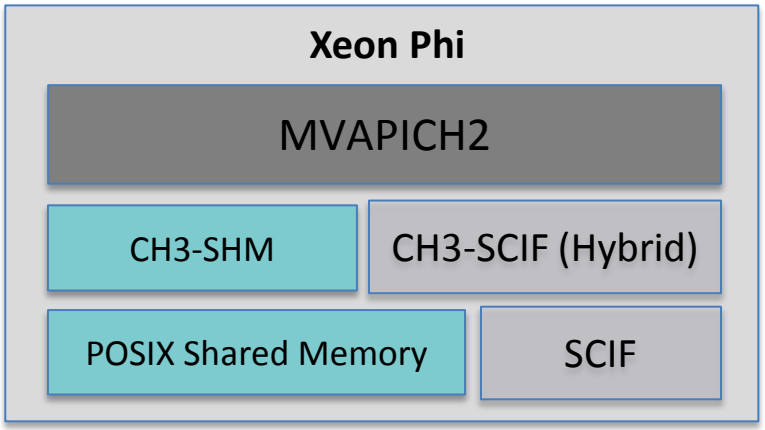
MPI Data Movement in Coprocessor-Only Mode



Intra-MIC Communication

MVAPICH2-MIC Channels for Intra-MIC Communication

- MVAPICH2 provides a hybrid of two channels
  - CH3-SHM
    - Derives the design for shared memory communication on host
    - Tuned for the Xeon Phi architecture
  - CH3-SCIF (Hybrid)
    - SCIF is a lower level API provided by MPSS
    - Provides user control of the DMA
    - Takes advantage of SCIF to improve performance



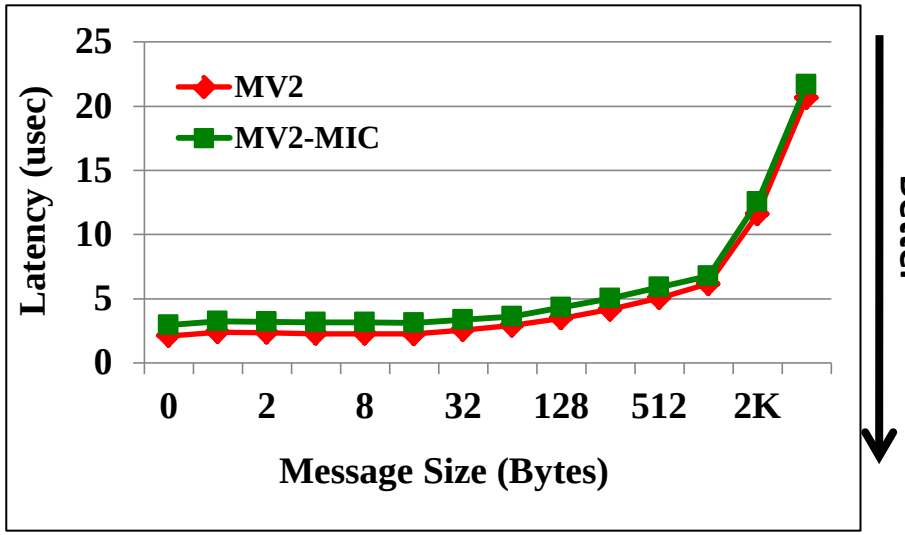
S. Potluri, A. Venkatesh, D. Bureddy, K. Kandalla, and D. K. Panda - Efficient Intra-node Communication on Intel-MIC Clusters - International Symposium on Cluster, Cloud and Grid Computing, May 2013 (CCGrid' 13).

S. Potluri, K. Tomko, D. Bureddy and D. K. Panda - Intra-MIC MPI Communication using MVAPICH2: Early Experience - TACC-Intel Highly-Parallel Computing Symposium (TI-HPCS), April 2012 - Best Student Paper

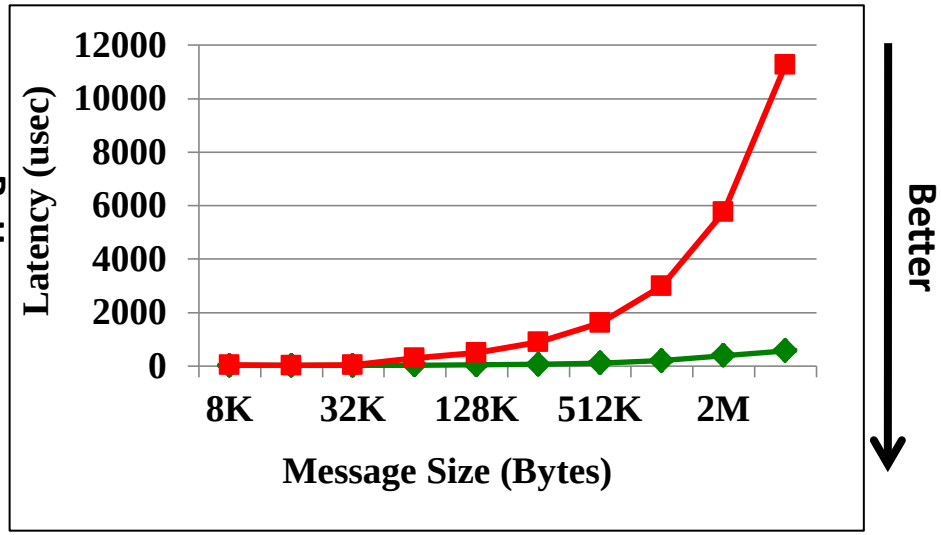
Experimental Setup

- Stampede@TACC
  - Dual socket oct-core Intel Sandy Bridge (E5-2680, 2.70 GHz)
  - 32 GB Memory
  - SE10P (B0-KNC) MIC Coprocessor
  - Mellanox IB FDR MT4099 HCA
  - CentOS release 6.3 (Final), MPSS 2.1.6720-13, Intel Composer_xe_2013.2.146
- Multi-MIC configuration
  - Dual socket oct-core Intel Sandy Bridge (E5-2670, 2.70 GHz)
  - 32 GB Memory
  - 5110P MIC Coprocessor
  - Mellanox IB FDR MT4099 HCA
  - RHEL 6.3 (Santiago), MPSS 2.1.6720-13, Intel Compiler 13.4.183
- MVAPICH2 1.9, MVAPICH2-MIC based on 1.9, Intel MPI 4.1.1.036

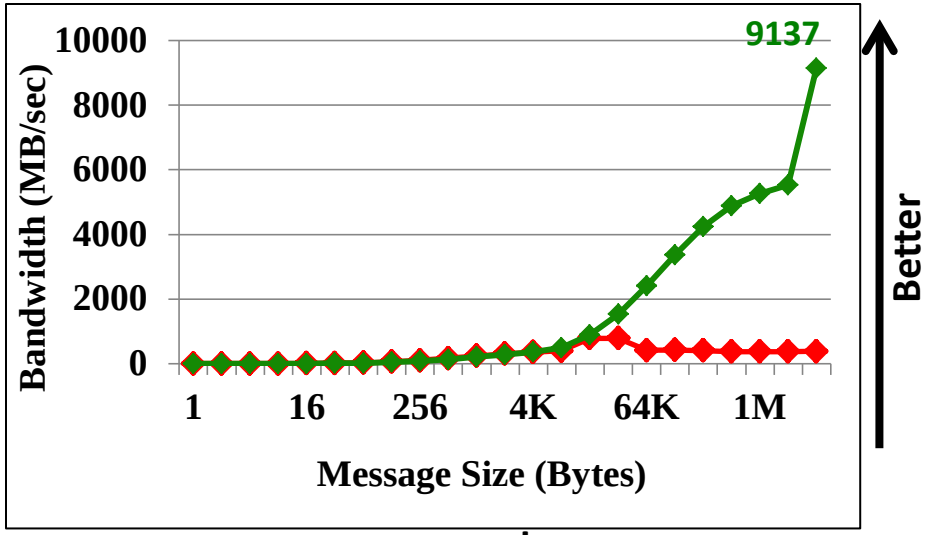
Intra-MIC - Point-to-Point Communication



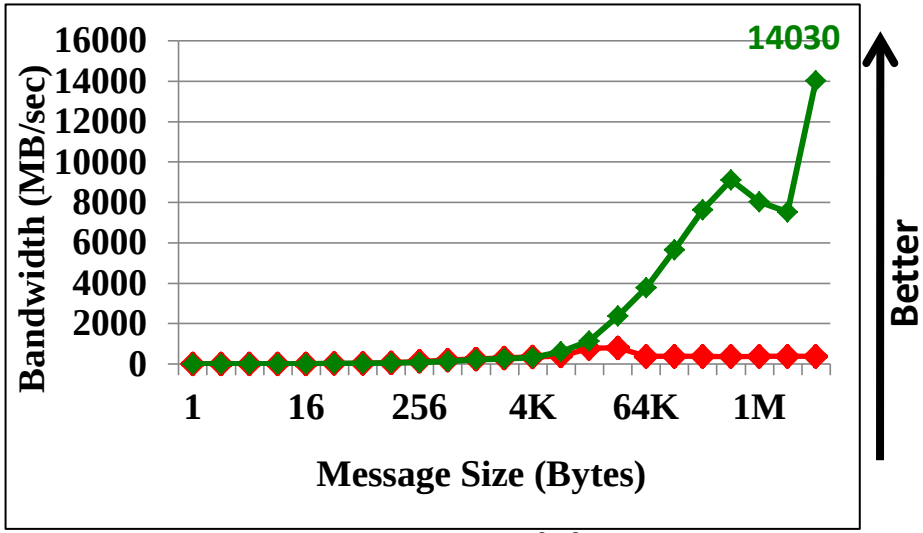
osu_latency (small)



osu_latency (large)

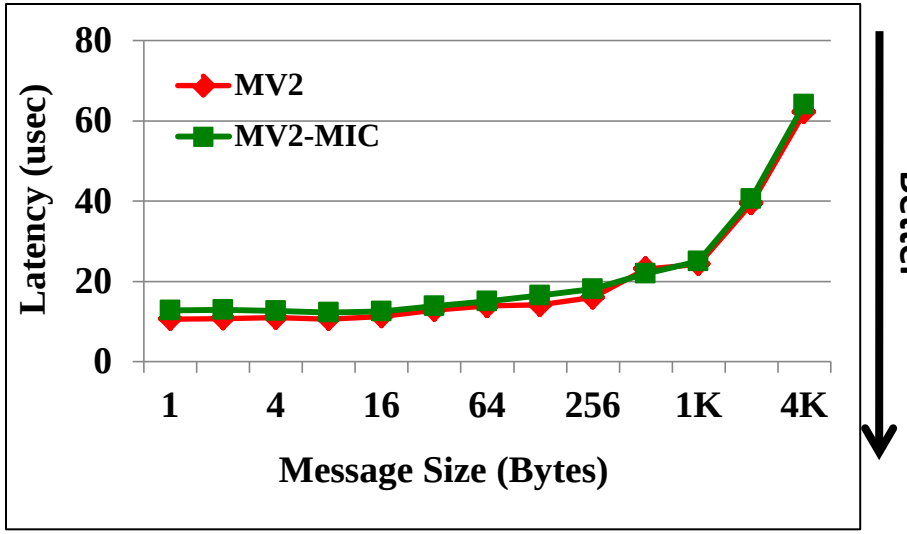


osu_bw

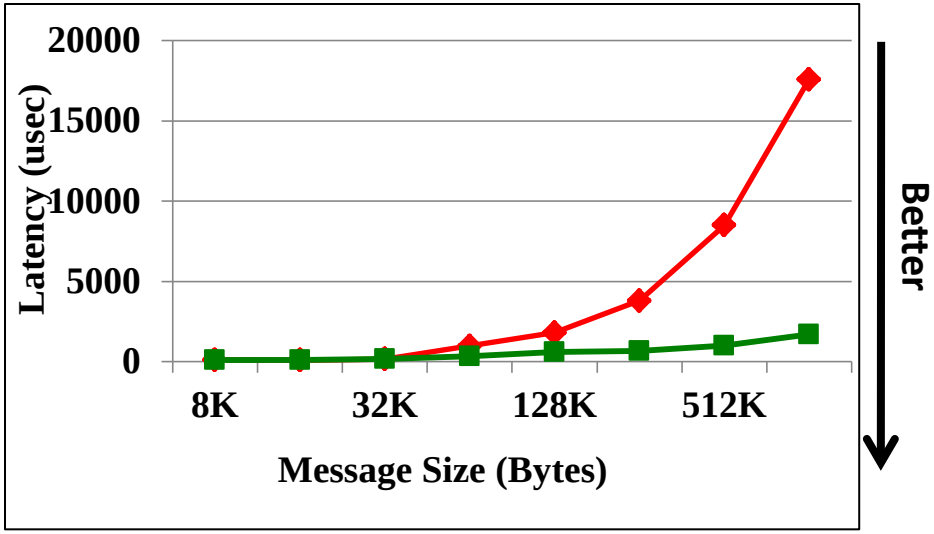


osu_bibw

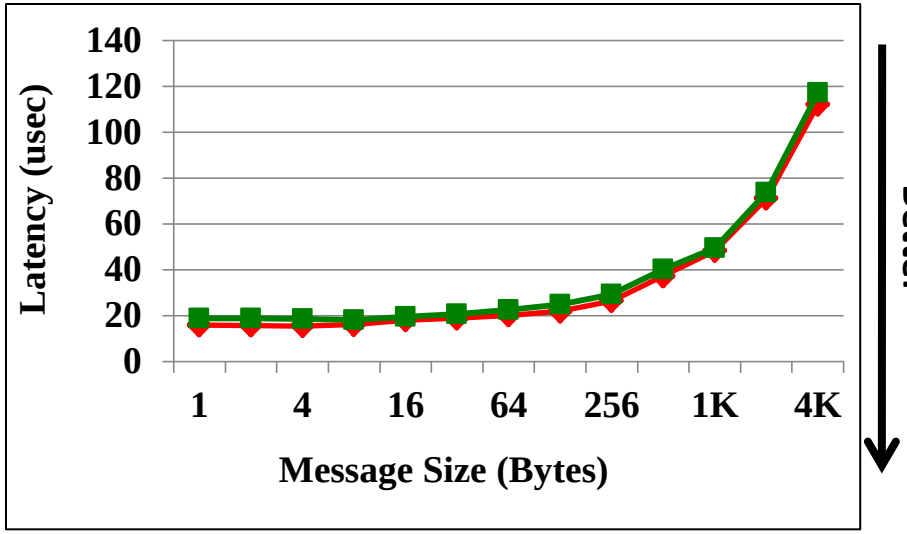
Intra-MIC - Collective Communication – Scatter



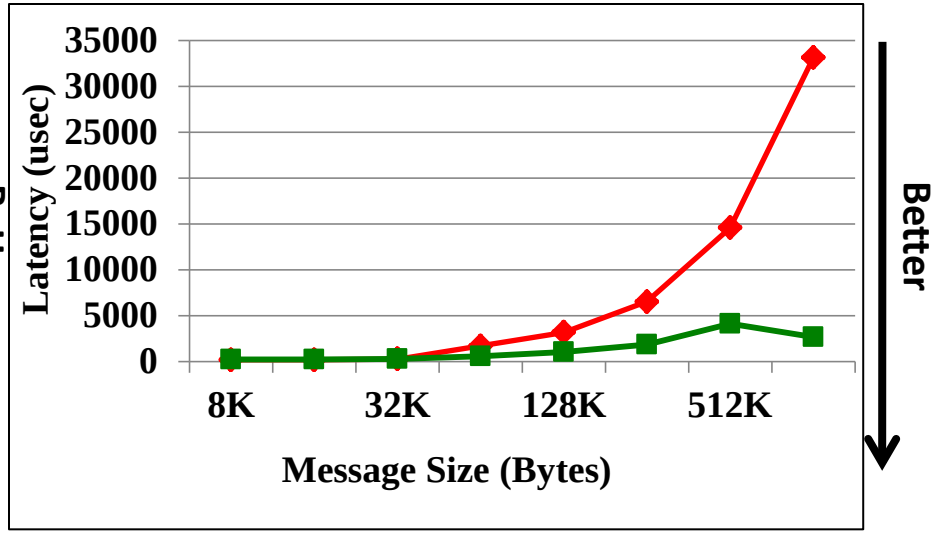
8 procs - osu_scatter (small)



8 procs - osu_scatter (large)

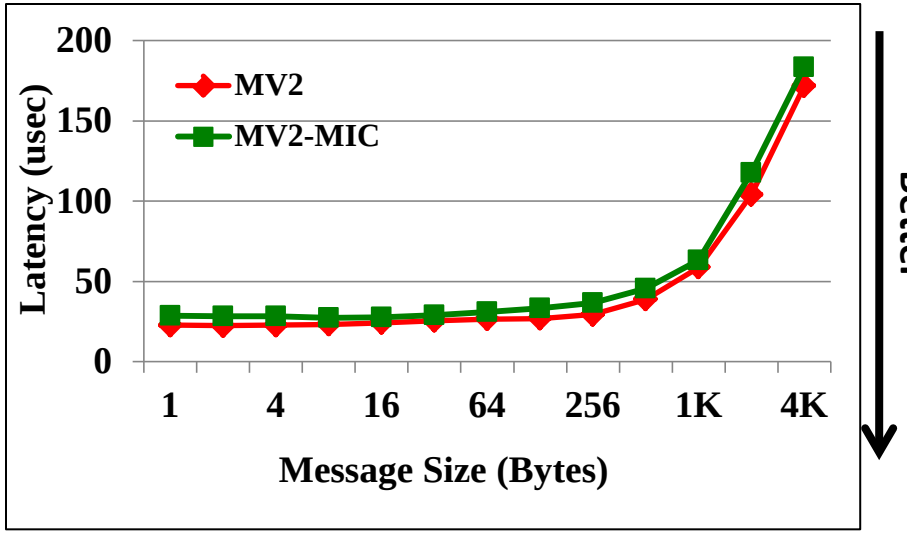


16 procs - osu_scatter (small)

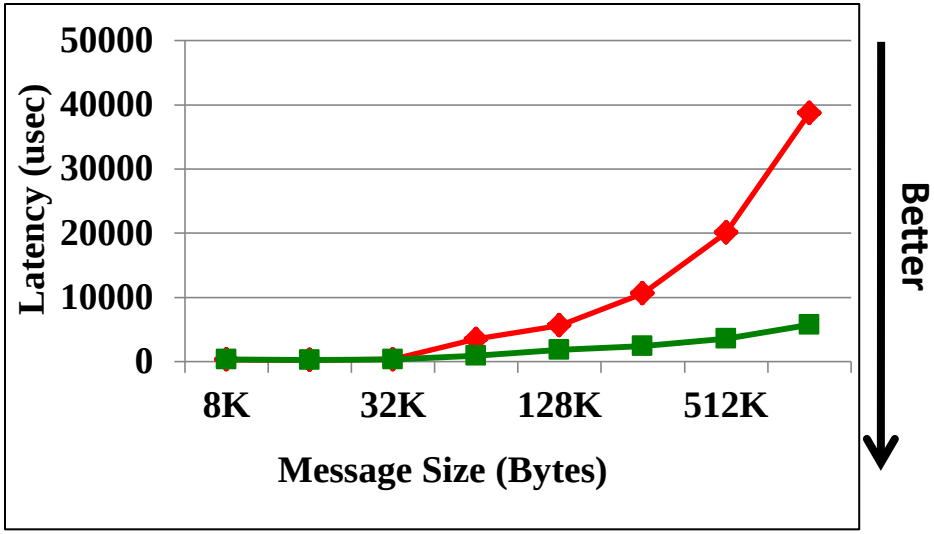


16 procs - osu_scatter (large)

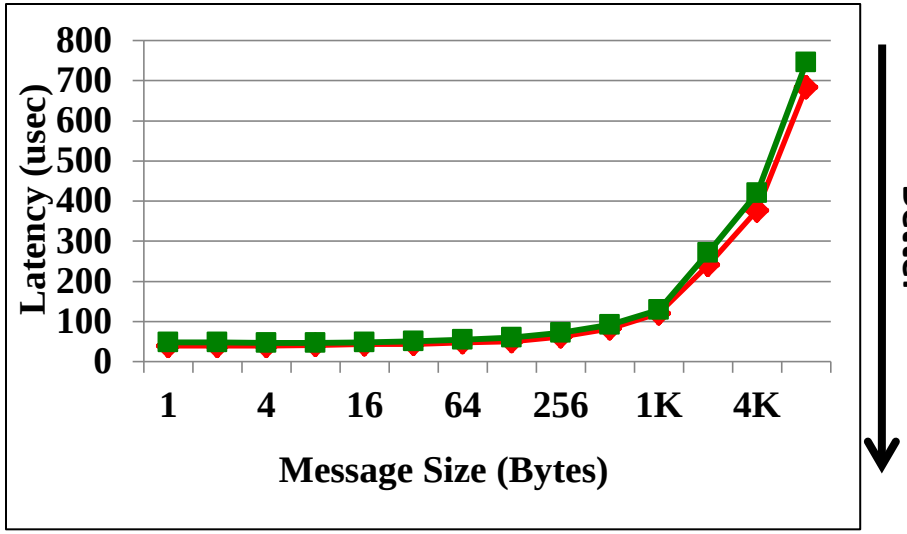
Intra-MIC - Collective Communication – Allgather



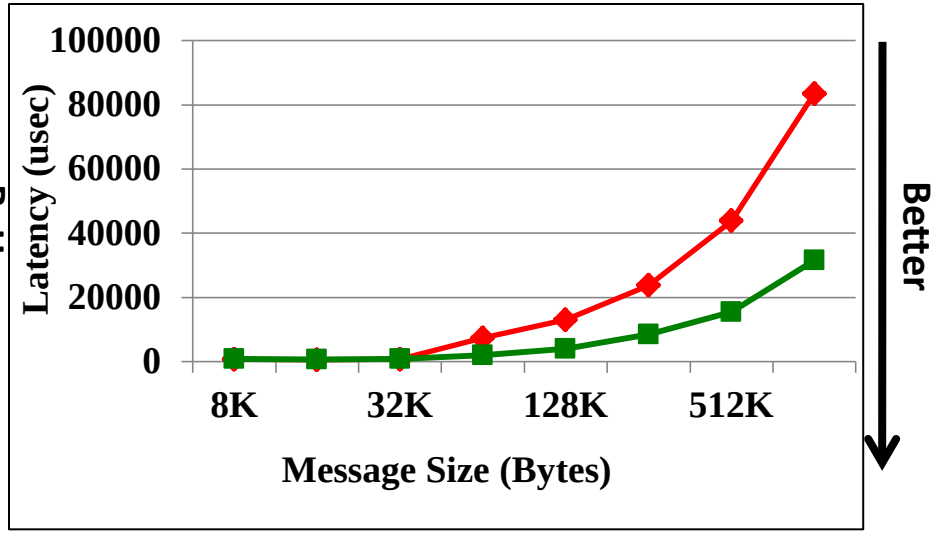
8 procs – osu_allgather (small)



8 procs – osu_allgather (large)

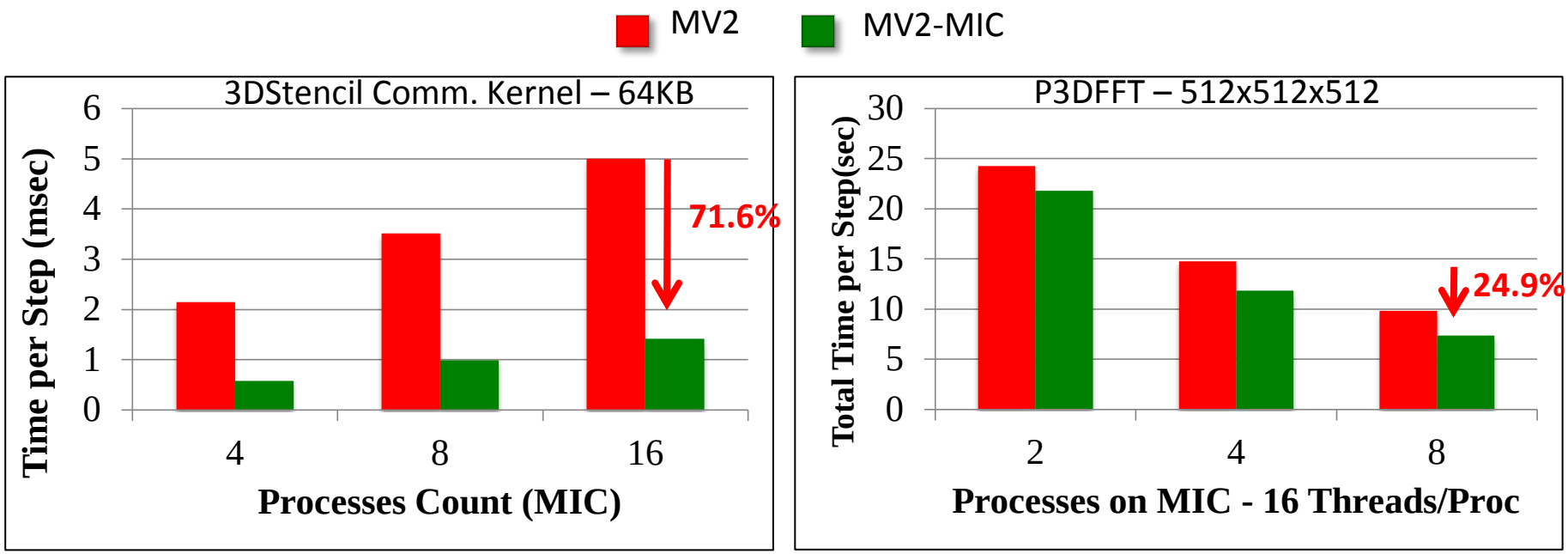


16 procs – osu_allgather (small)



16 procs – osu_allgather (large)

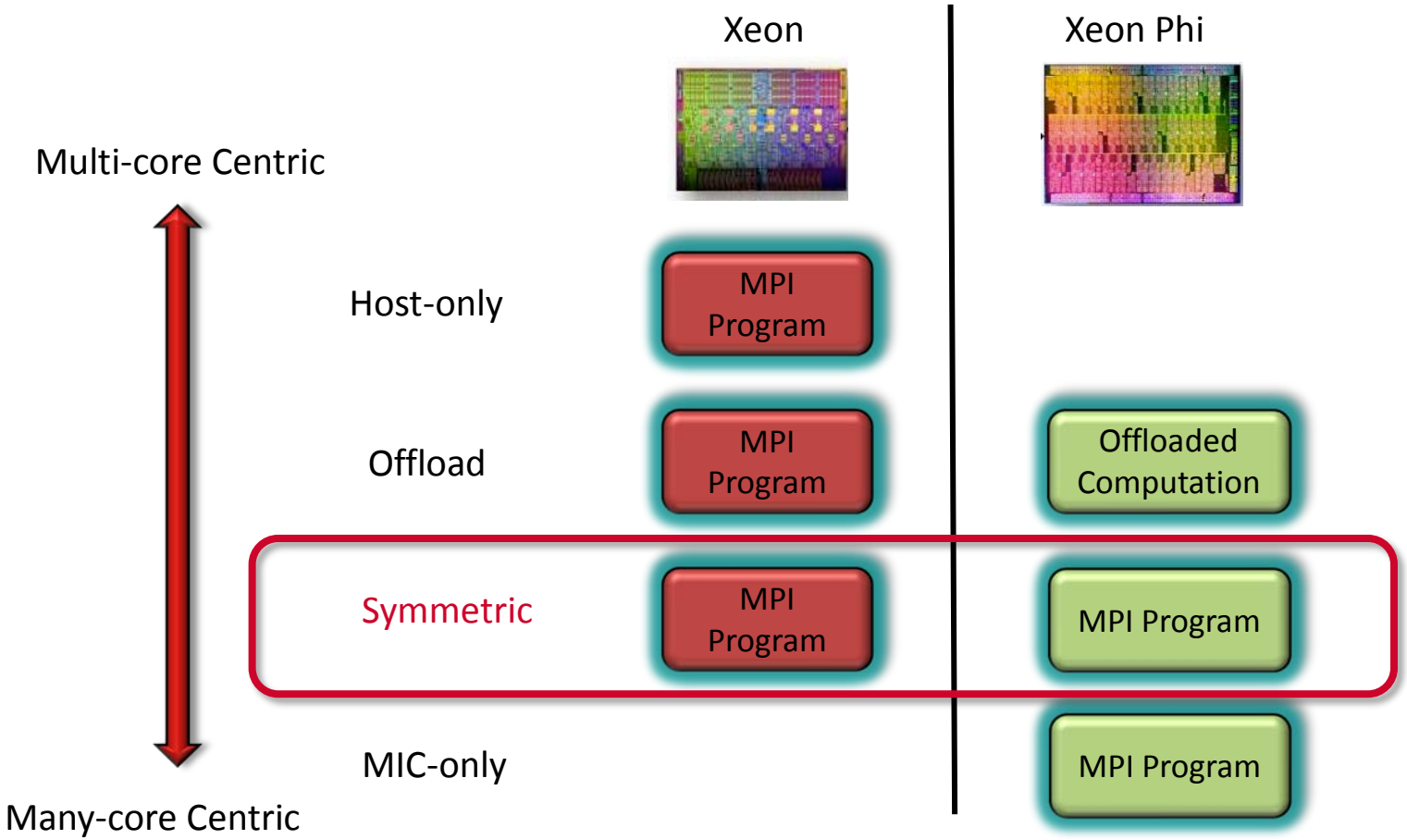
Intra-MIC – 3DStencil Comm. Kernel and P3DFFT



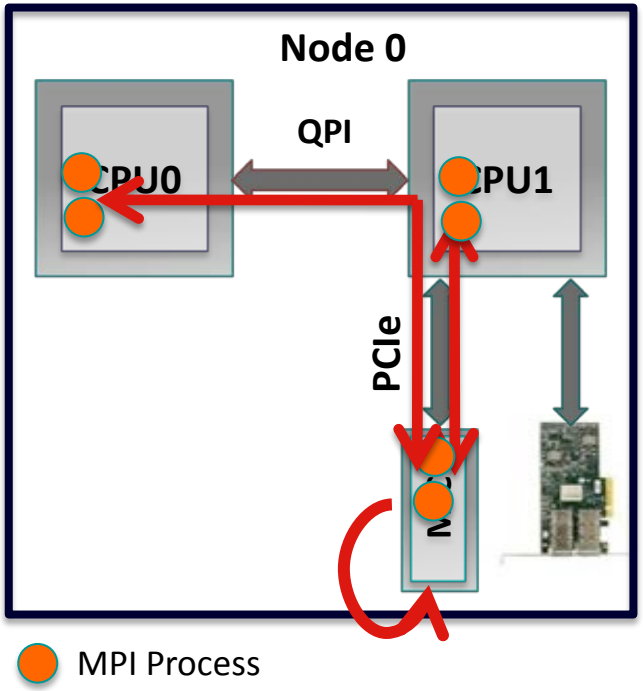
- **3DStencil - near-neighbor communication – up to 6 neighbors – 64KB message size**
- **P3DFFT, a popular library for 3D Fast Fourier Transforms**
  - **(MPI + OpenMP) version**
  - **Alltoall communication**
  - **Test performs forward transform and a backward transform in each iteration**

MPI Applications on MIC Clusters - IntraNode

- MPI processes running on the MIC and the **Host**

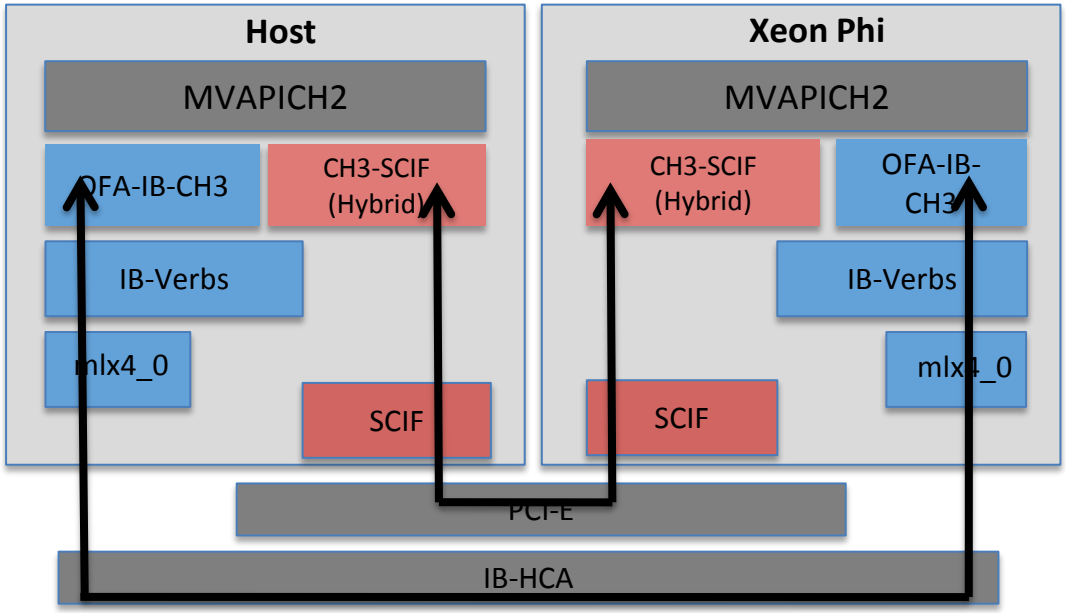


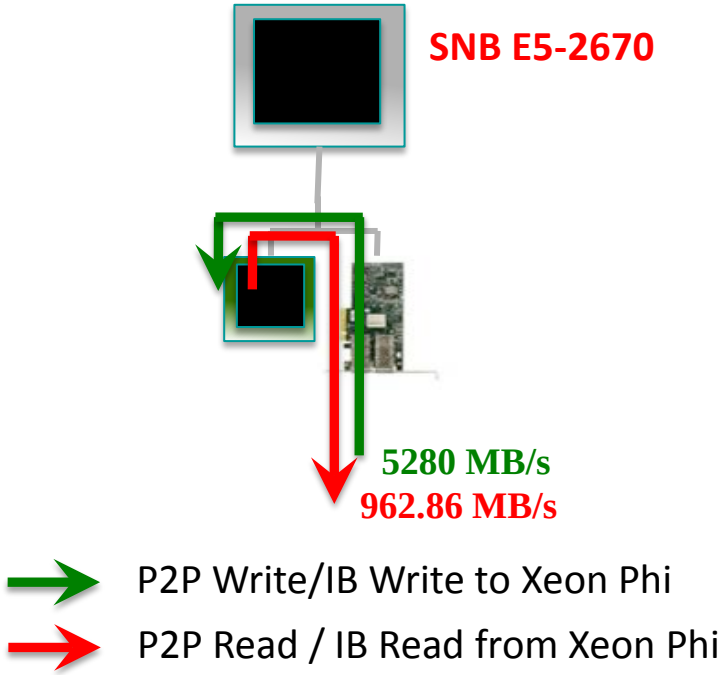
MPI Data Movement in Symmetric Mode – IntraNode



1. Intra-MIC 2. Intra-Socket MIC-Host 3. Inter-Socket MIC-Host

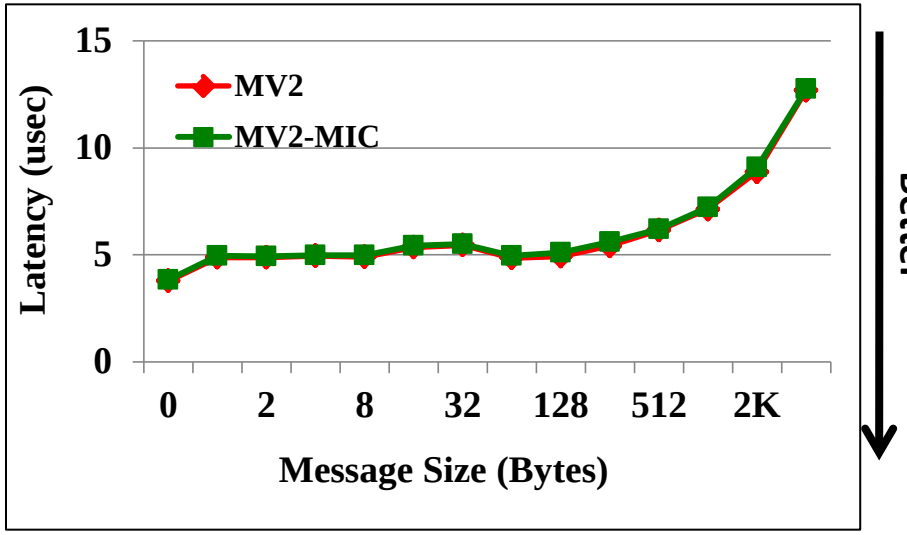
MVAPICH2-MIC Channels for MIC-Host Communication



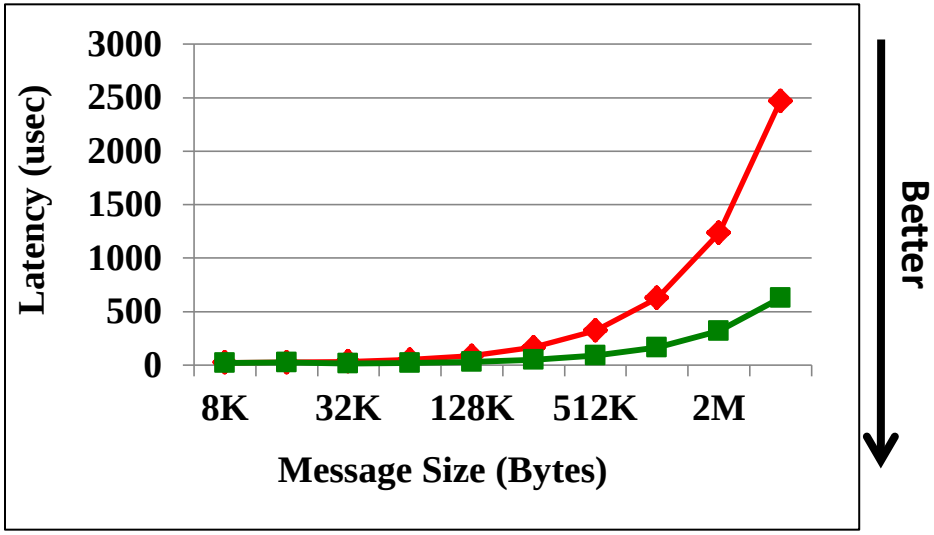


- IB Channel (OFA-IB-CH3)
  - High-performance InfiniBand channel in MVAPICH2
  - Uses IB verbs – can use both DirectIB and IB-SCIF by interface selection – MV2_IBA_HCA
  - DirectIB provides low latency path
  - Limited by P2P bandwidth on Sandy Bridge nodes
- CH3-SCIF (Hybrid)
  - SCIF can be used for communication between Xeon Phi and Host
  - DirectIB is used for low latency

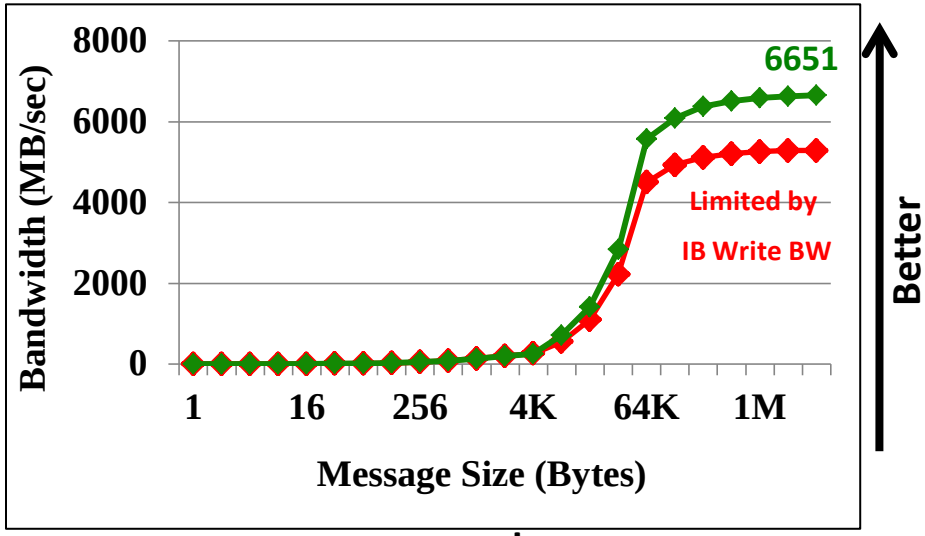
Intra-Socket Host-MIC - Point-to-Point Communication



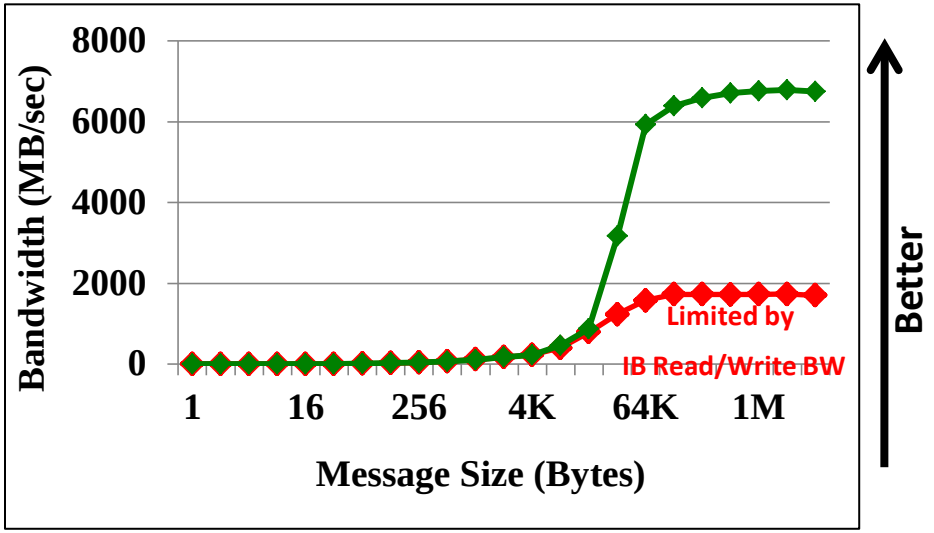
osu_latency (small)



osu_latency (large)



osu_bw

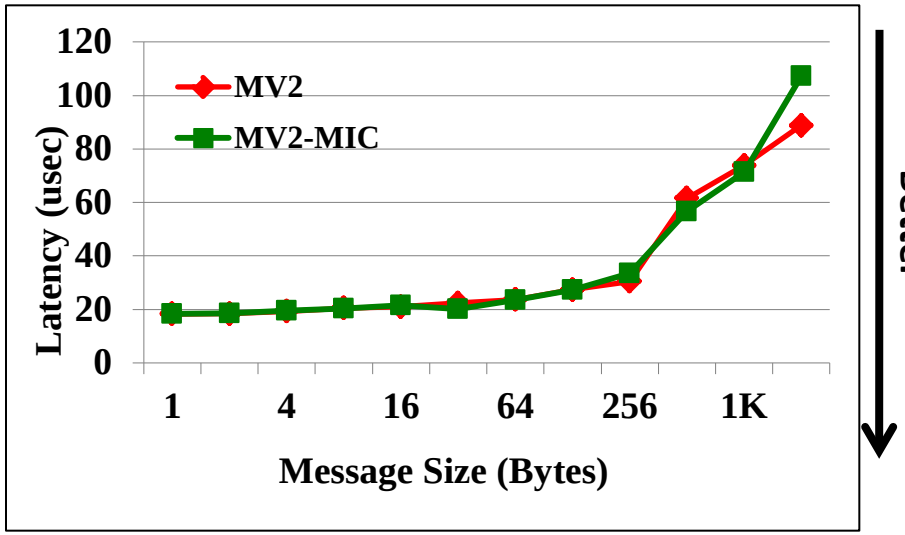


osu_bibw

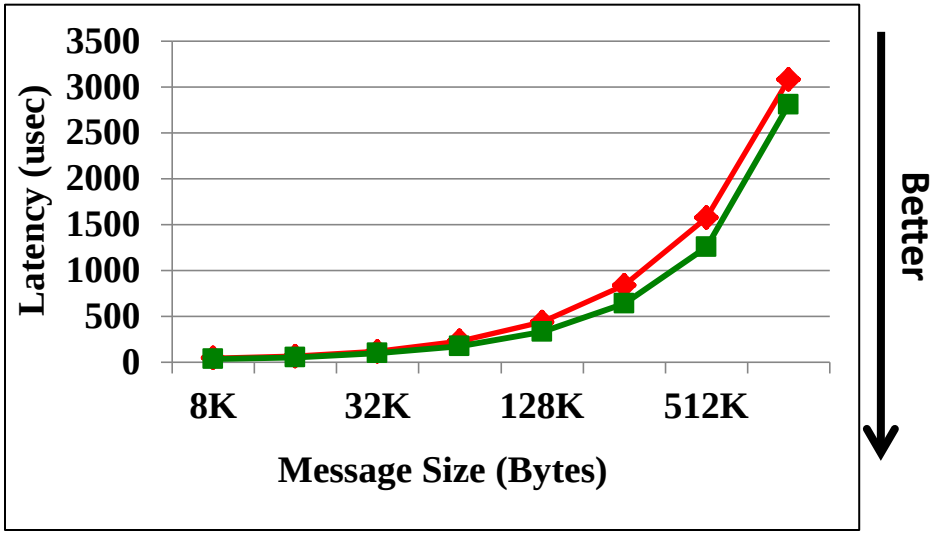
Similar for Inter-Socket Host-MIC

Intra-Node Symmetric Mode - Collective Performance

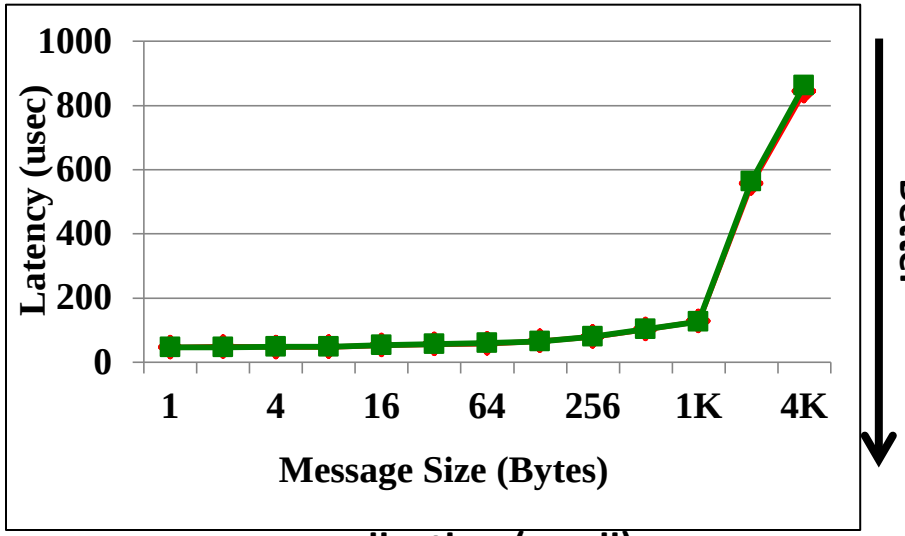
16 Procs on Host and 16 Procs on MIC



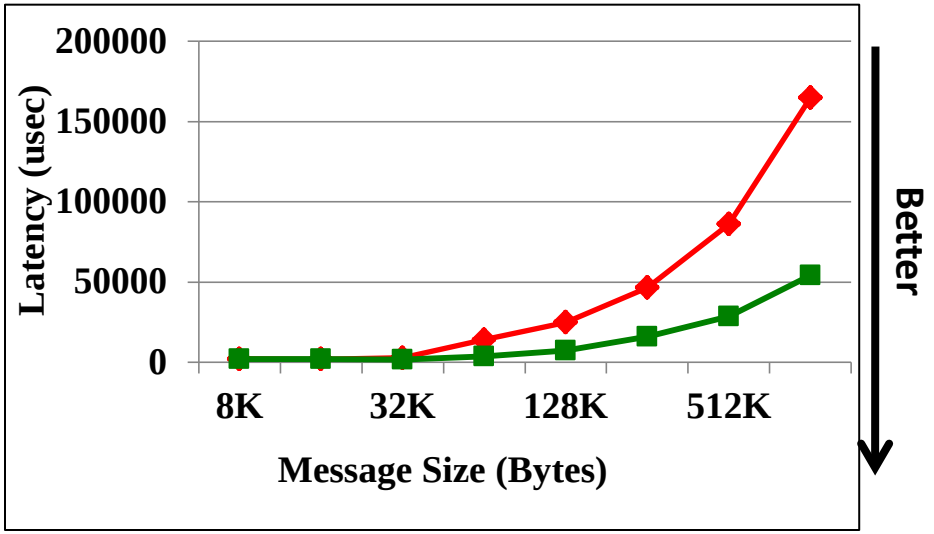
osu_scatter (small)



osu_scatter (large)

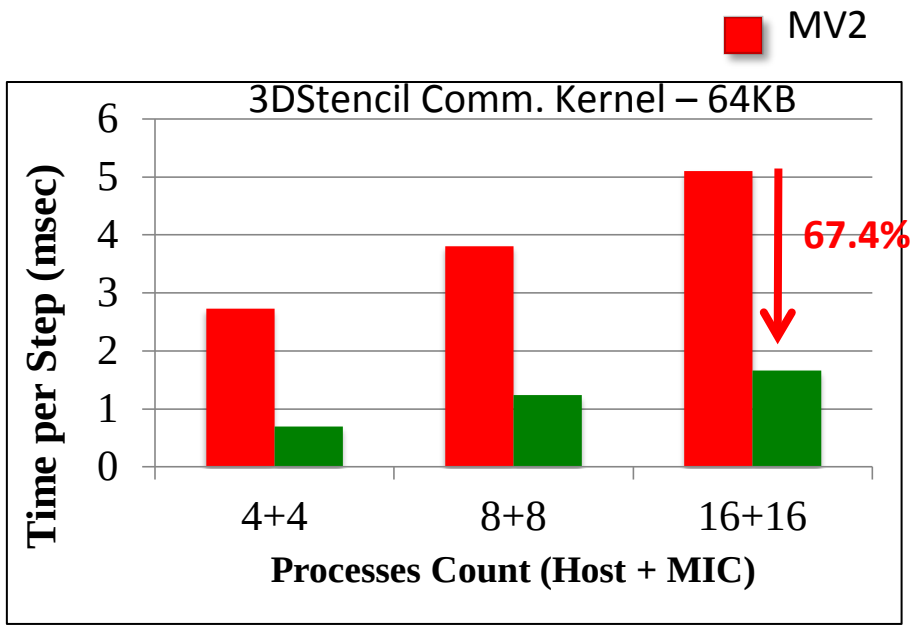


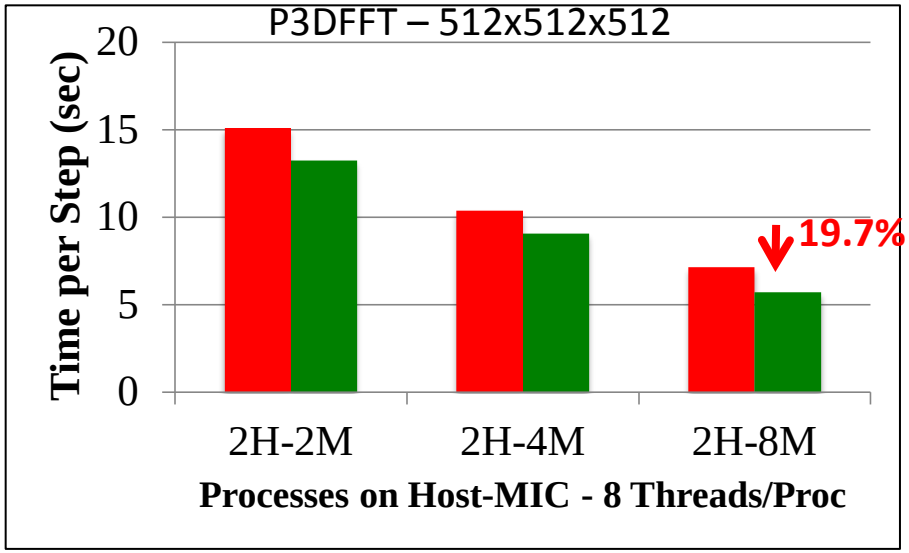
osu_allgather (small)



osu_allgather (large)

IntraNode Symmetric Mode – 3DStencil Comm. Kernel and P3DFFT



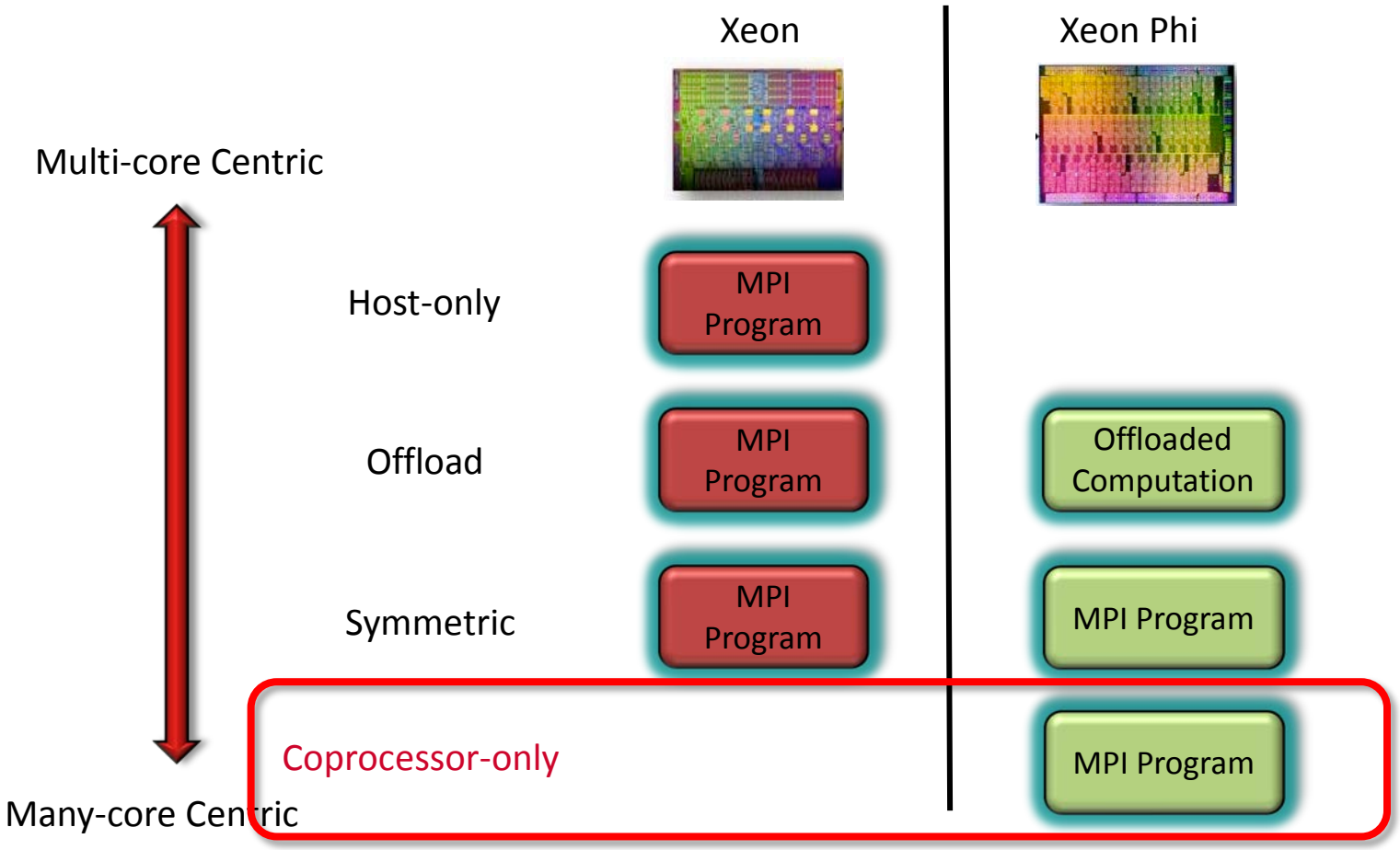


- **Running across MIC and Host**
- **Benefits from shared memory tuning and hybrid SCIF+IB design**
- **To explore the best combination of MPI processes and threads**

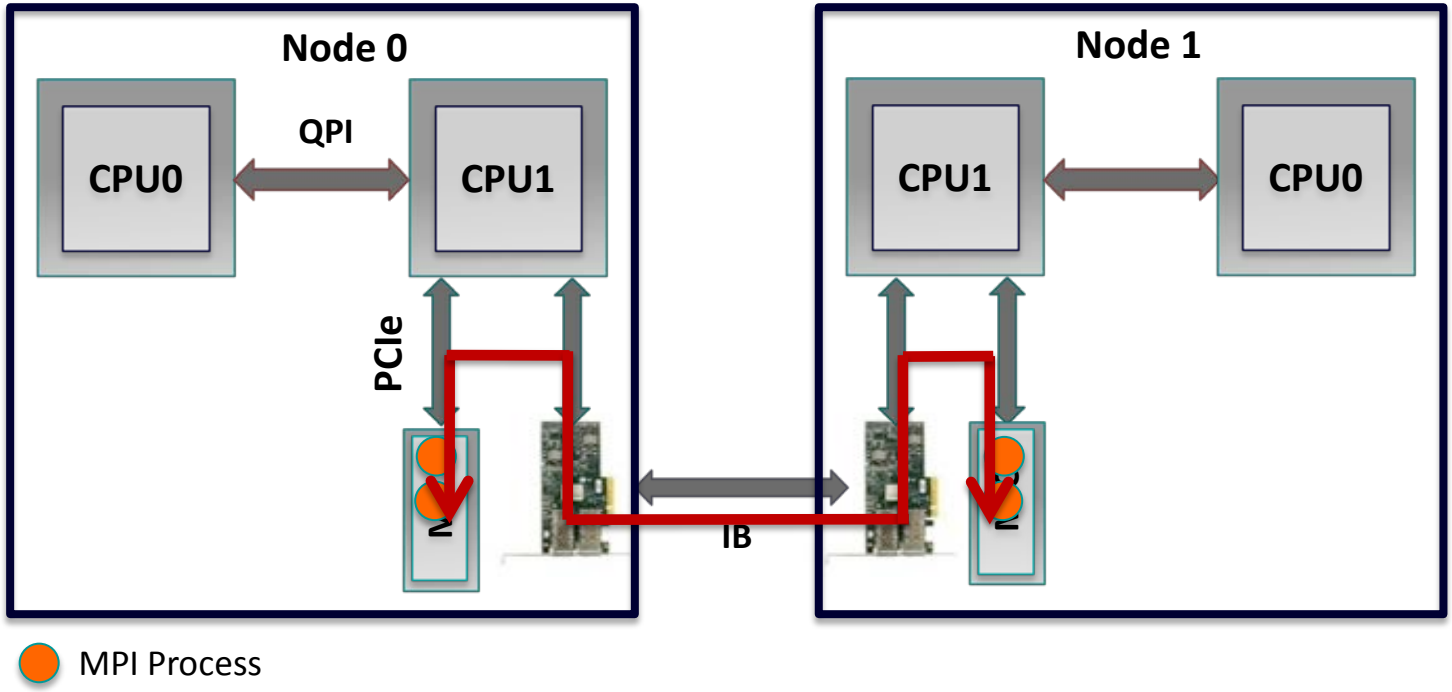
Efficient MPI Communication on Xeon Phi Clusters

- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only Mode
  - Symmetric Mode
- Comparison with Intel's MPI Library

MPI Applications on MIC Clusters - InterNode

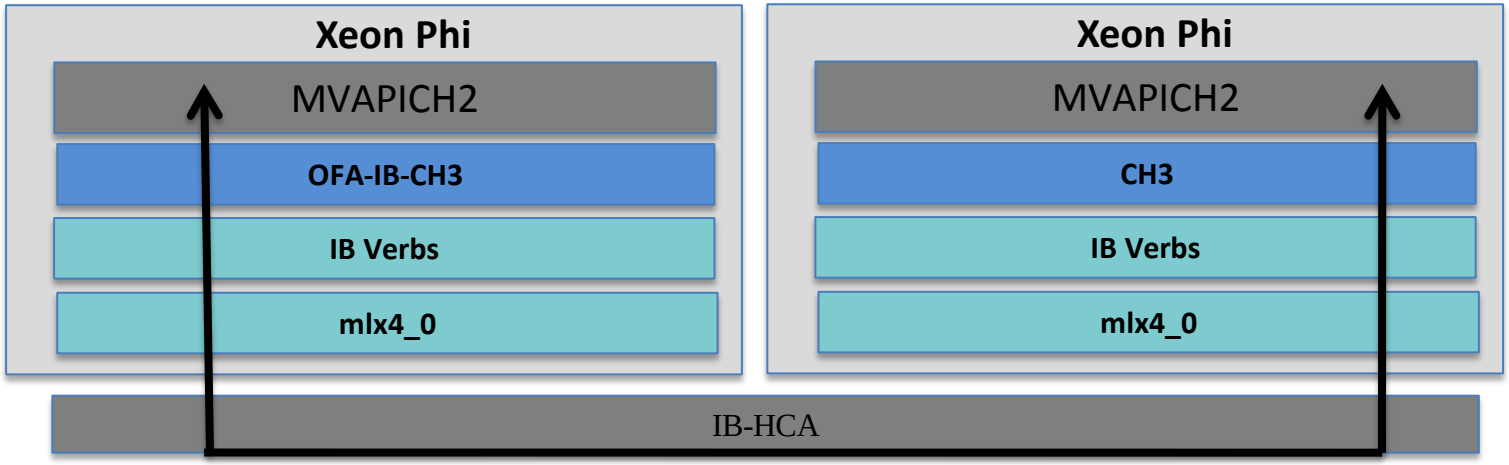


MPI Data Movement in Coprocessor-Only Mode



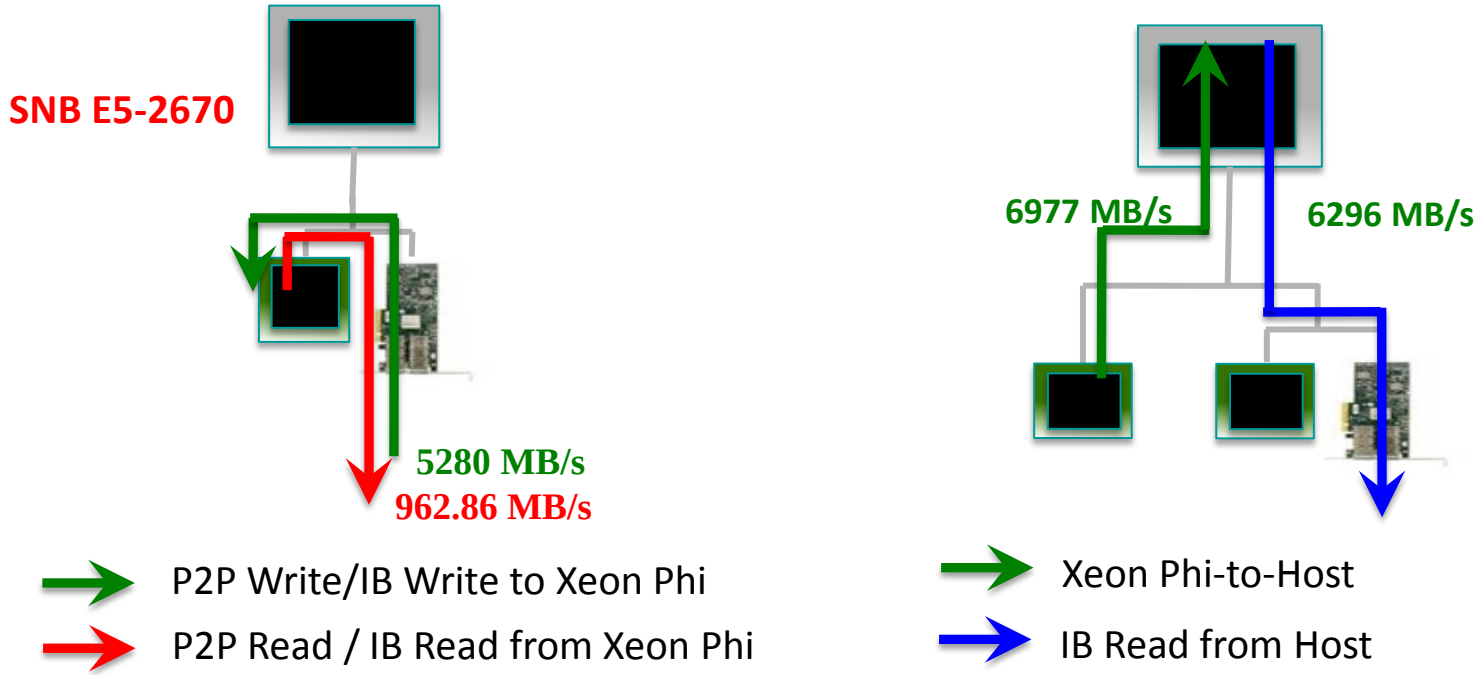
1. MIC-RemoteMIC

MVAPICH2 Channels for MIC-RemoteMIC Communication



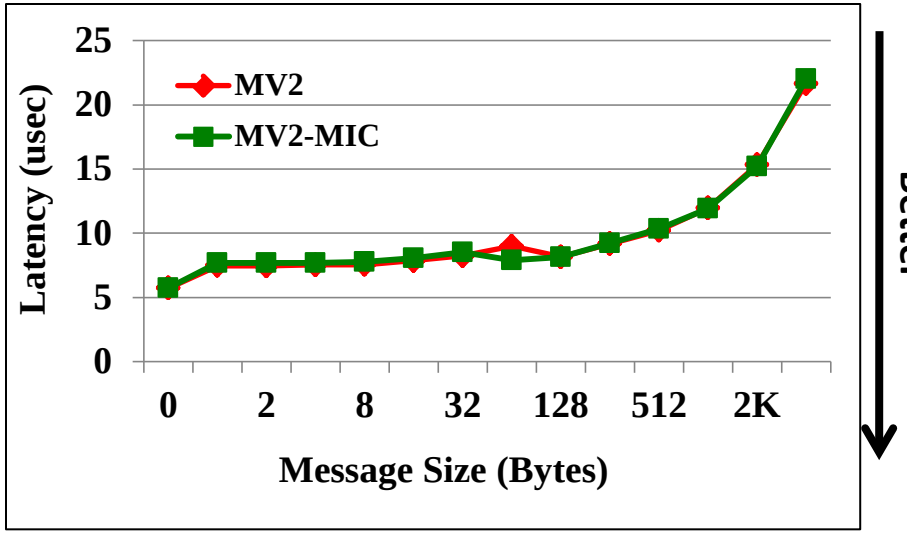
- CH3-IB channel
  - High-performance InfiniBand channel in MVAPICH2
  - Implemented using IB Verbs - DirectIB
  - Currently does not support advanced features like hardware multicast,etc.
  - Limited by P2P Bandwidth offered on Sandy Bridge platform

Host Proxy-based Designs in MVAPICH2-MIC

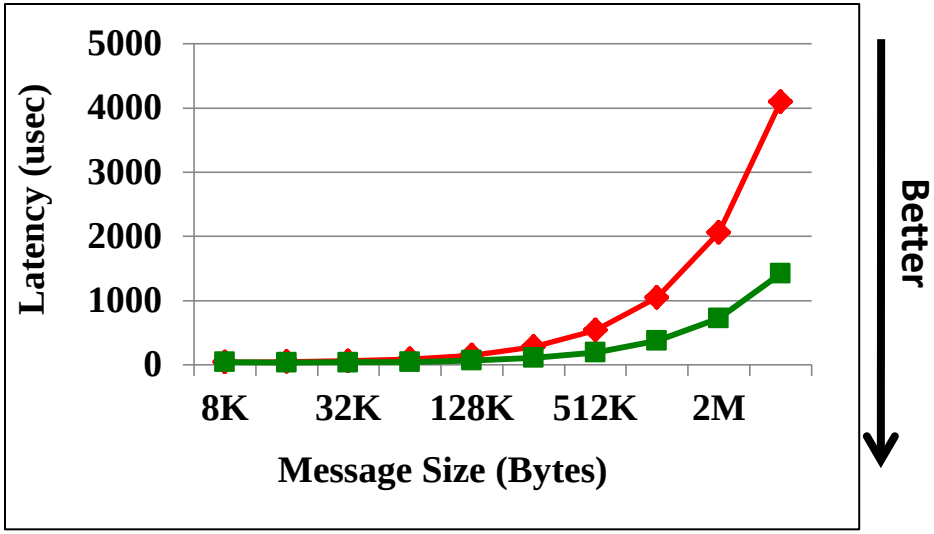


- Direct IB channels is limited by P2P read bandwidth
- MVAPICH2-MIC uses a hybrid DirectIB + host proxy-based approach to work around this

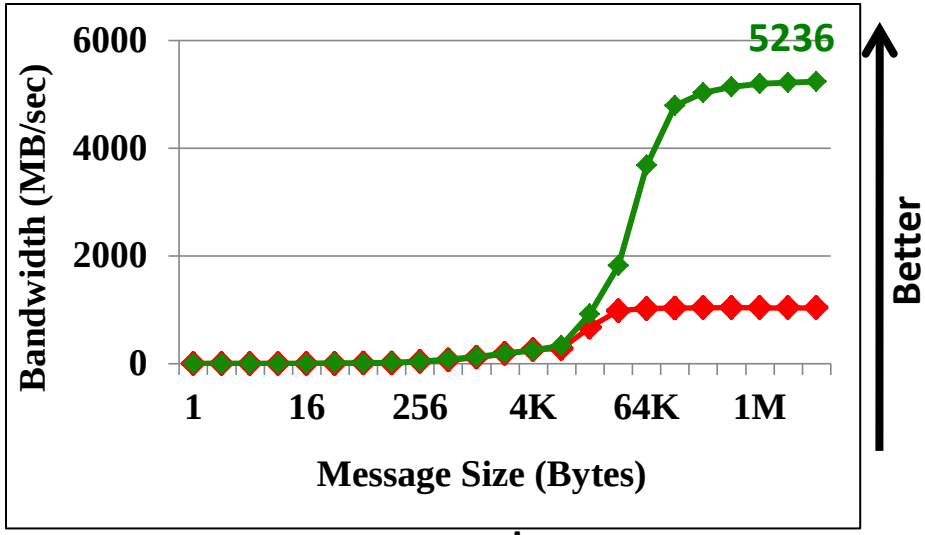
MIC-RemoteMIC Point-to-Point Communication



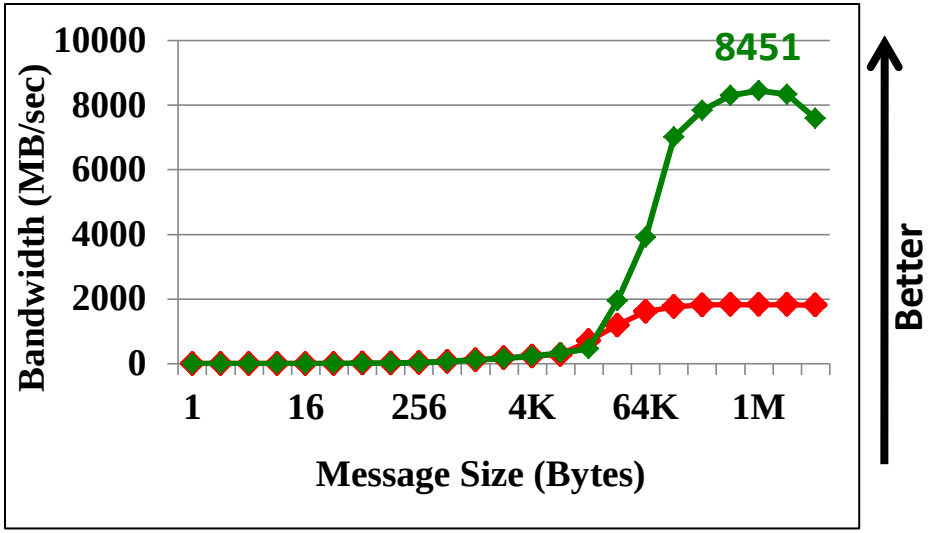
osu_latency (small)



osu_latency (large)



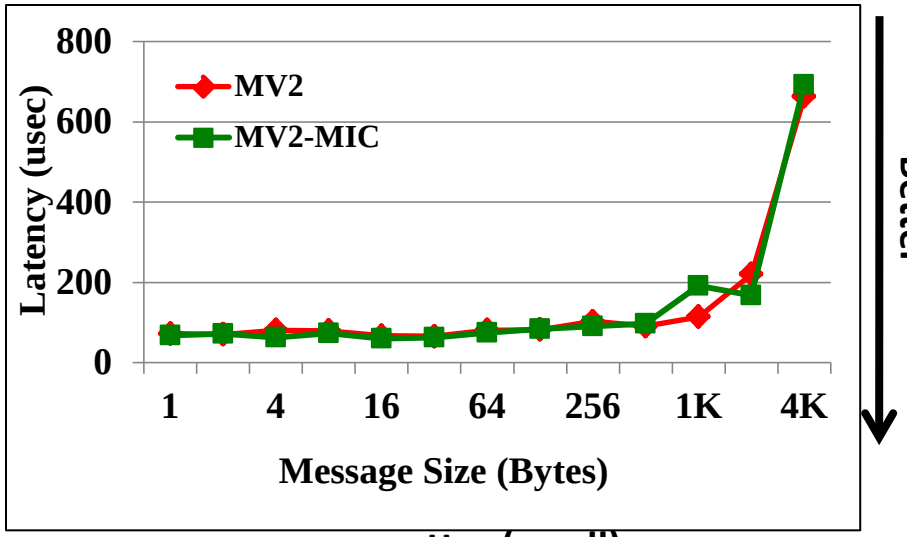
osu_bw



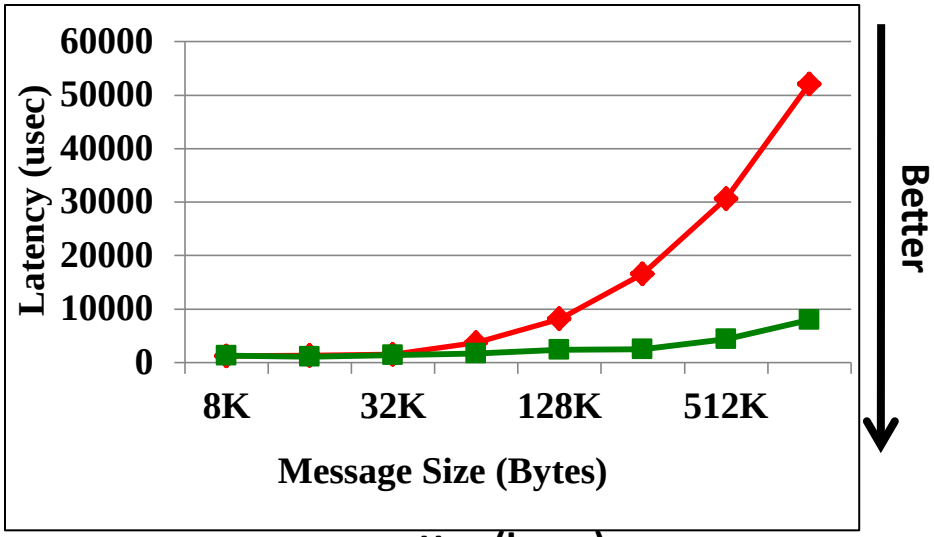
osu_bibw

InterNode Coprocessor-only Mode - Collective Communication

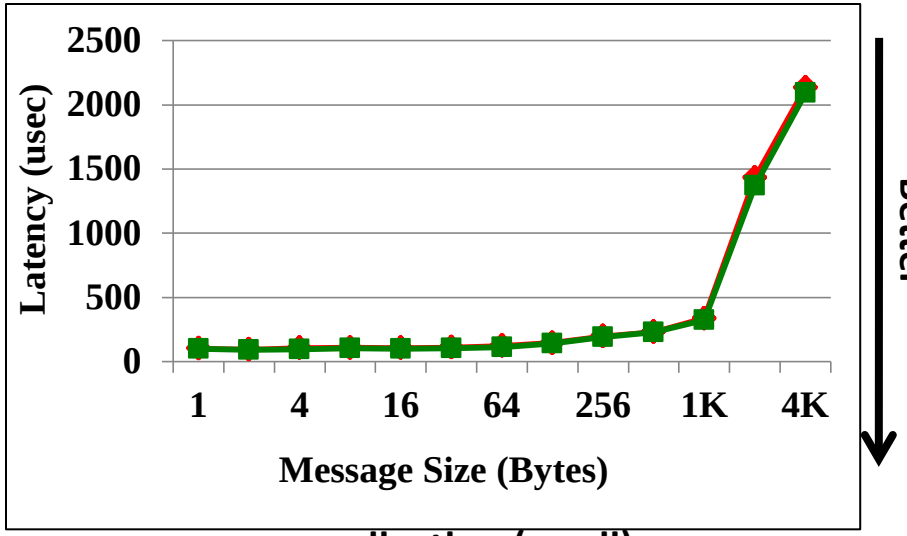
8 MICs with 8 Procs/MIC



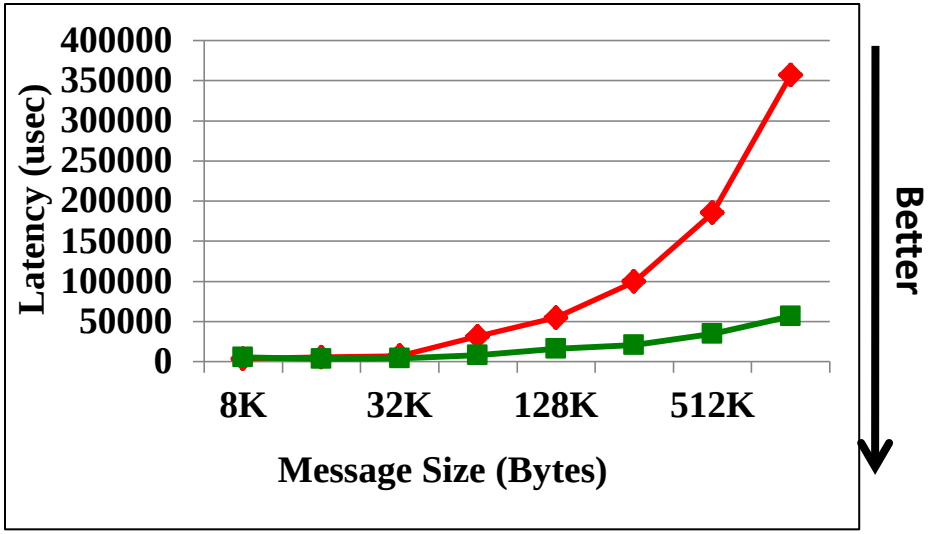
osu_scatter (small)



osu_scatter (large)

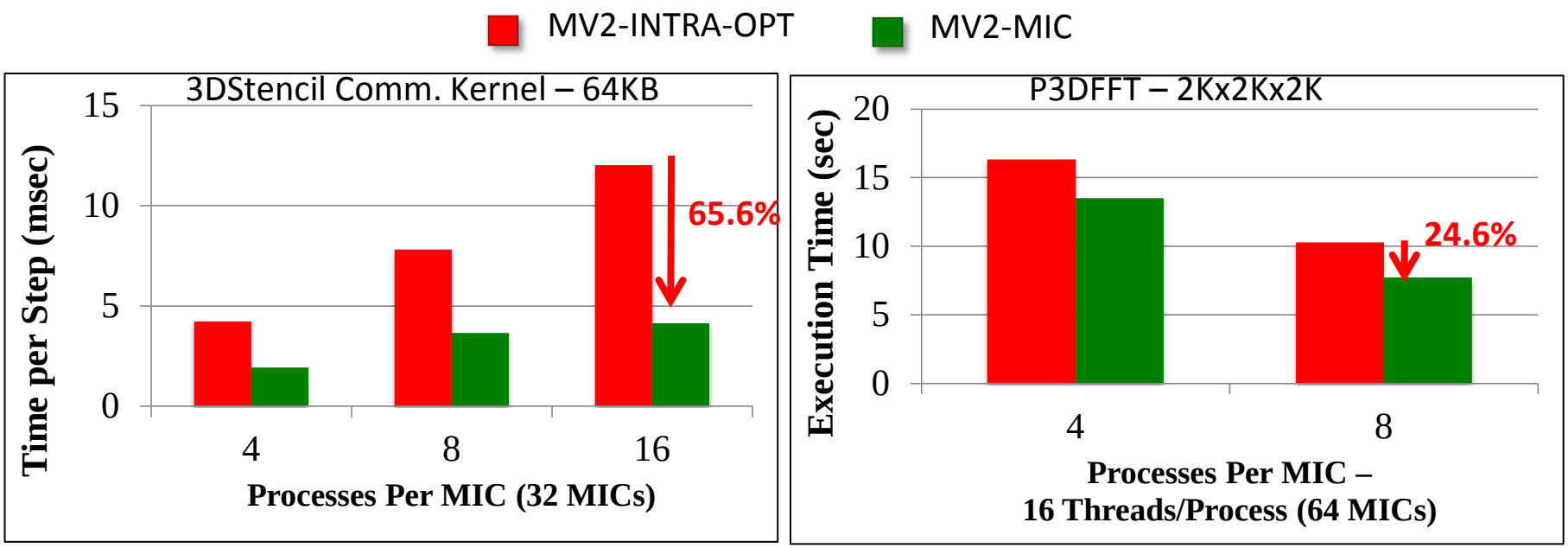


osu_allgather (small)



osu_allgather (large)

InterNode - Coprocessor-only Mode - P3DFFT Performance

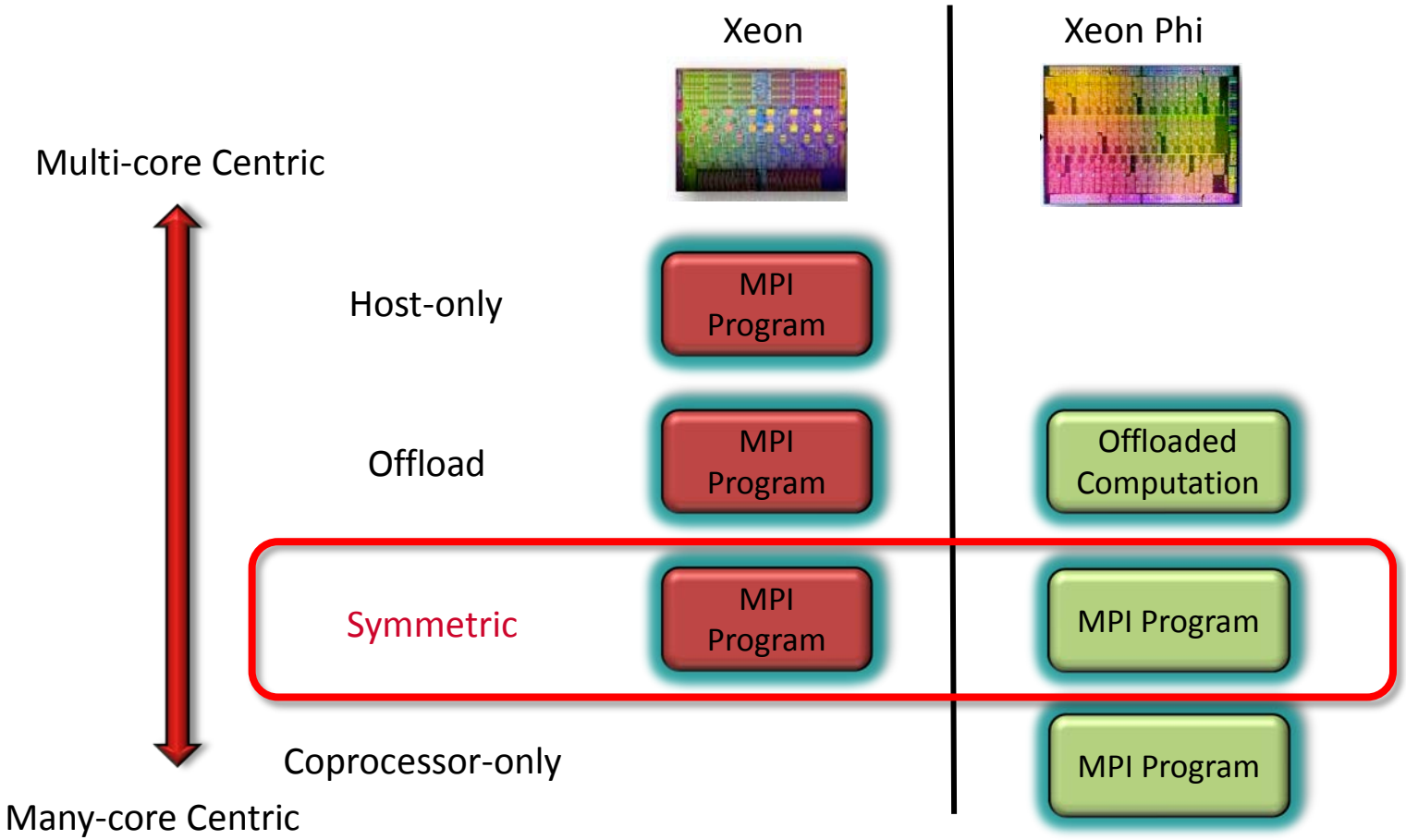


- **MV2-INTRA-OPT includes the intra-node tuning and optimizations**
- **Comparison shows benefit of proxy**

MVAPICH2-MIC Design for Clusters with IB and MIC

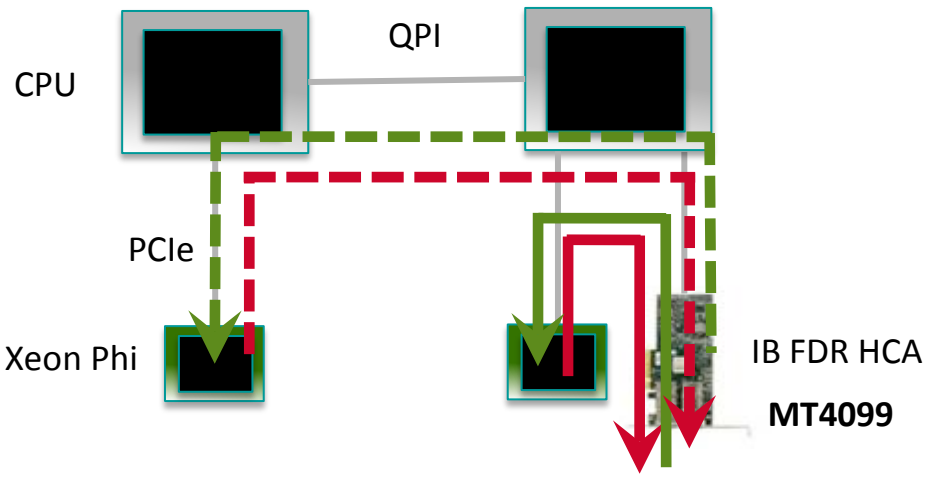
- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only Mode
  - Symmetric Mode
- Comparison with Intel's MPI Library

MPI Applications on MIC Clusters - InterNode



Direct-IB Communication

- **IB-verbs based communication from Xeon Phi**
- **No porting effort for MPI libraries with IB support**
- **Data movement between IB HCA and Xeon Phi: implemented as peer-to-peer transfers**

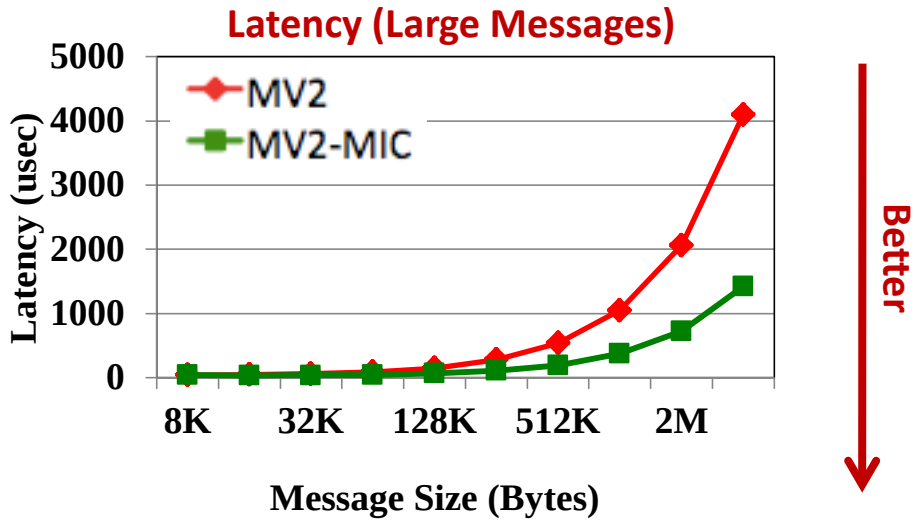


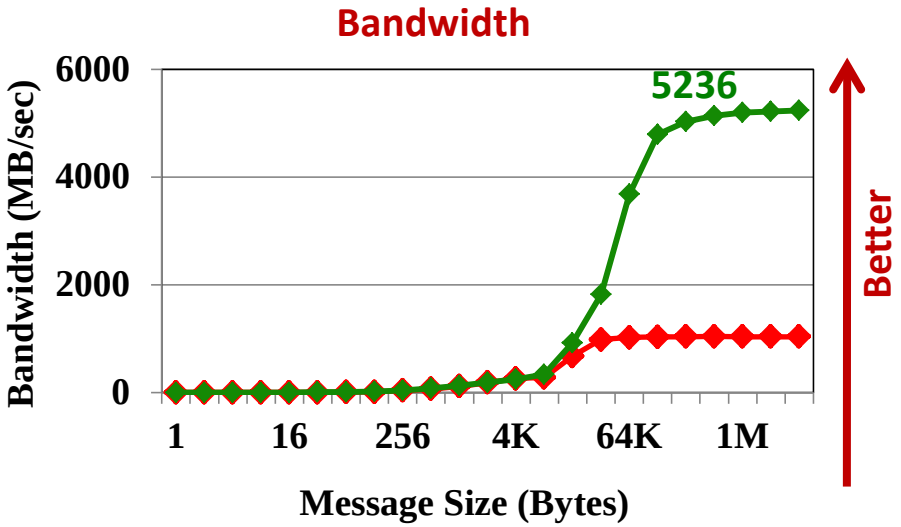
Peak IB FDR Bandwidth: 6397 MB/s

		E5-2670 (SandyBridge)	E5-2680 v2 (IvyBridge)
Intra-socket	IB Read from Xeon Phi (P2P Read)	962 MB/s (15%)	3421 MB/s (54%)
	IB Write to Xeon Phi (P2P Write)	5280 MB/s (83%)	6396 MB/s (100%)
Inter-socket	IB Read from Xeon Phi (P2P Read)	370 MB/s (6%)	247 MB/s (4%)
	IB Write to Xeon Phi (P2P Write)	1075 MB/s (17%)	1179 MB/s (19%)

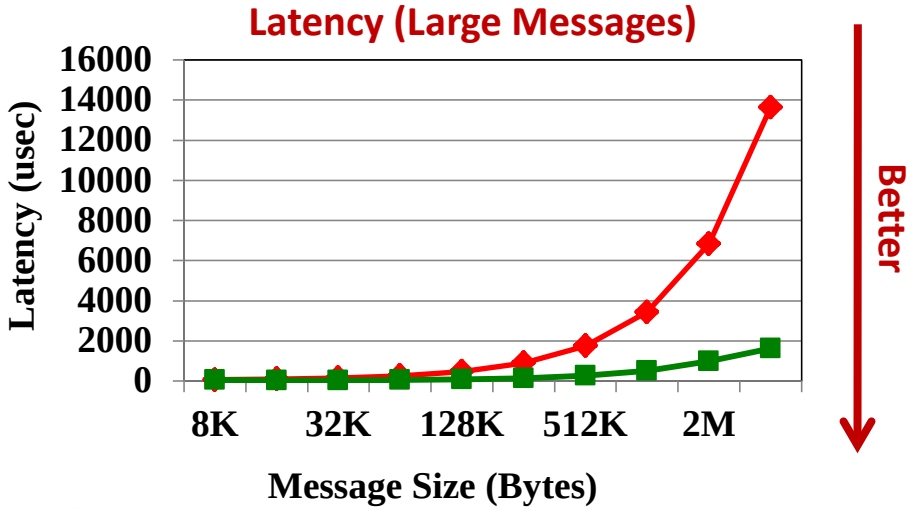
MIC-Remote-MIC P2P Communication

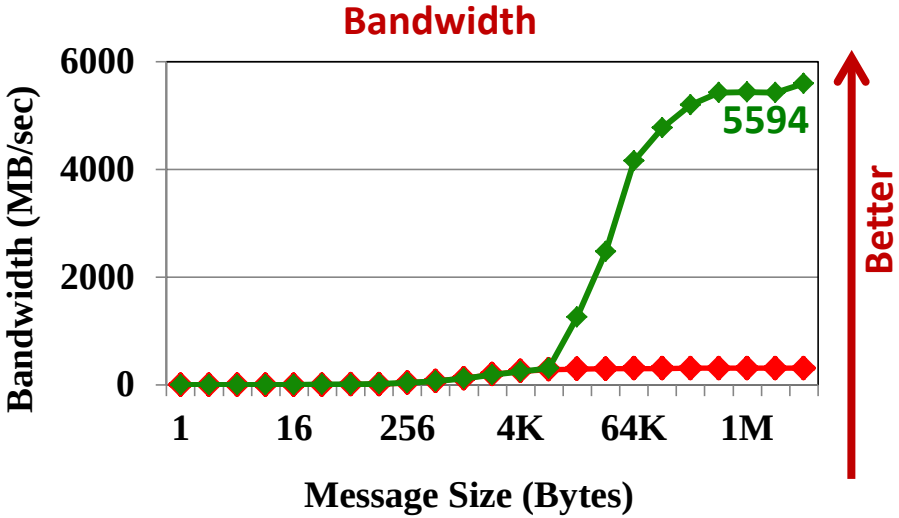
Intra-socket P2P





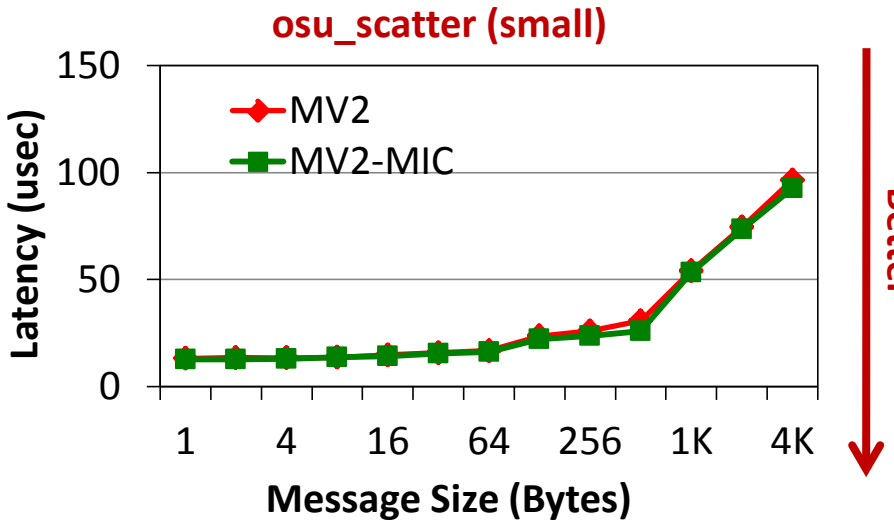
Inter-socket P2P

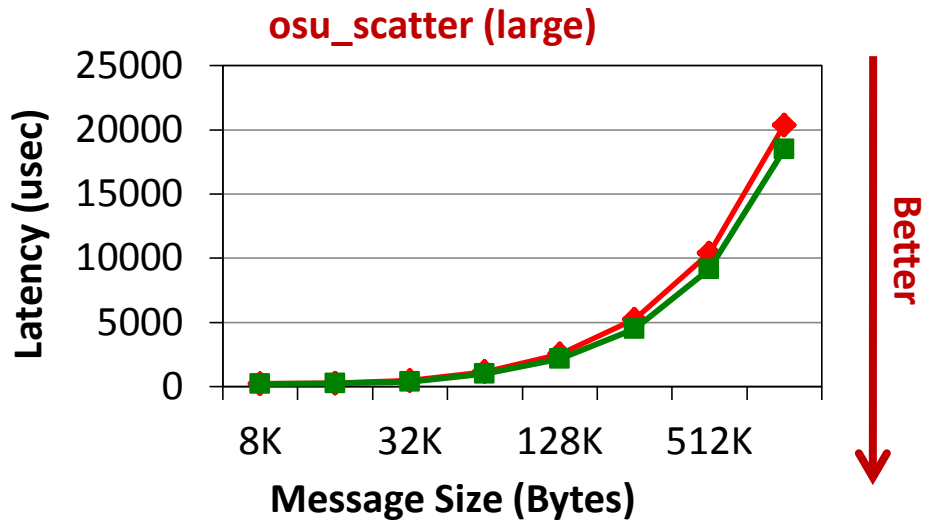


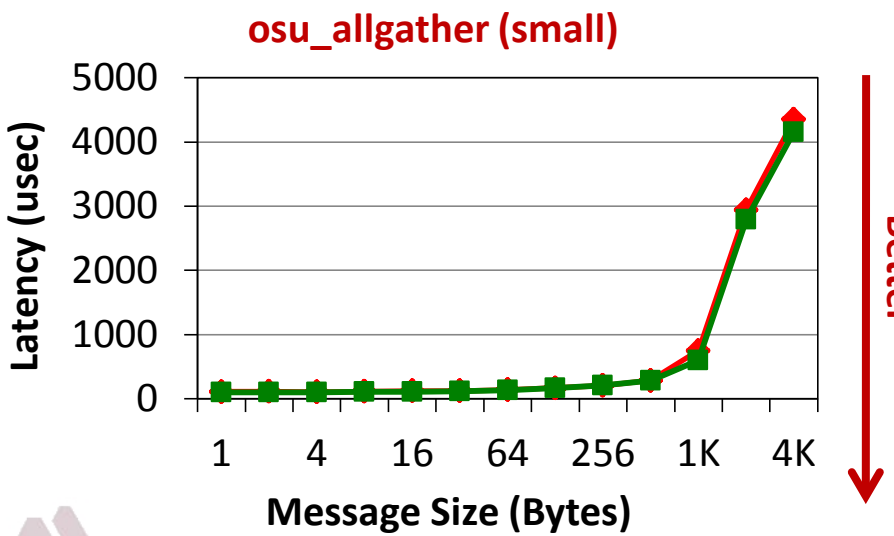


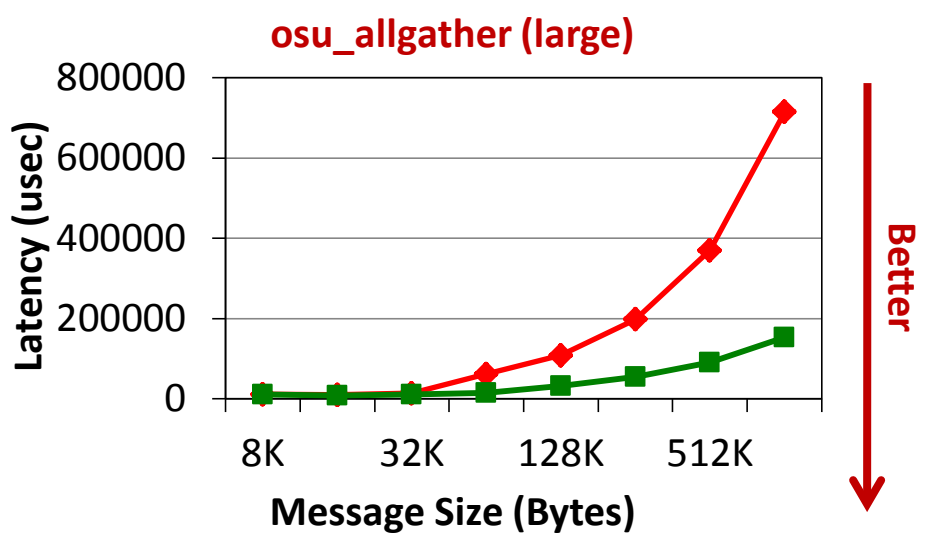
Inter-Node Symmetric Mode Collective Communication

8 Nodes with 8 Procs/Host and 8 Procs/MIC

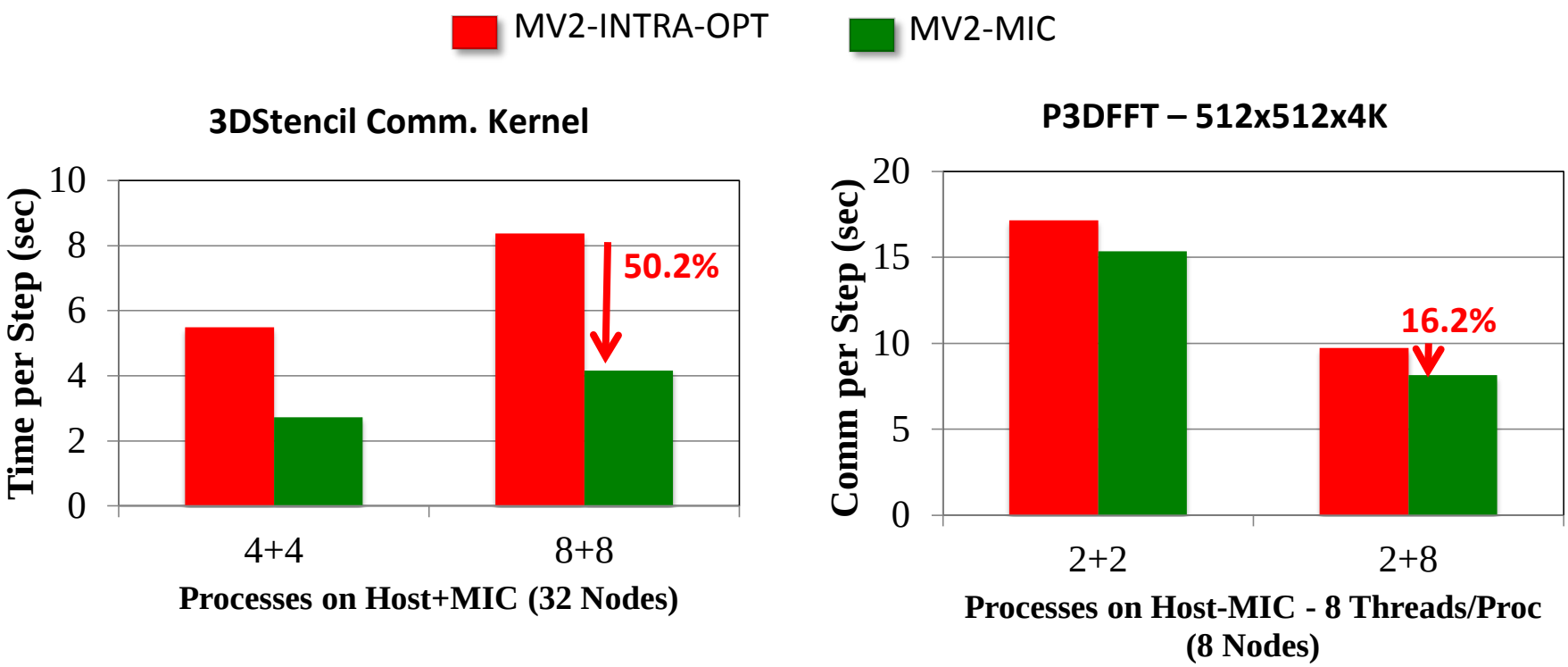






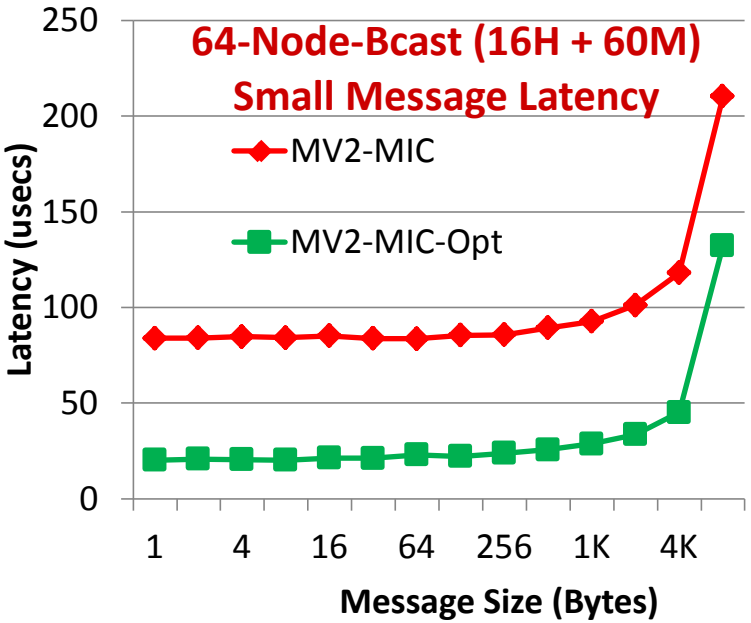


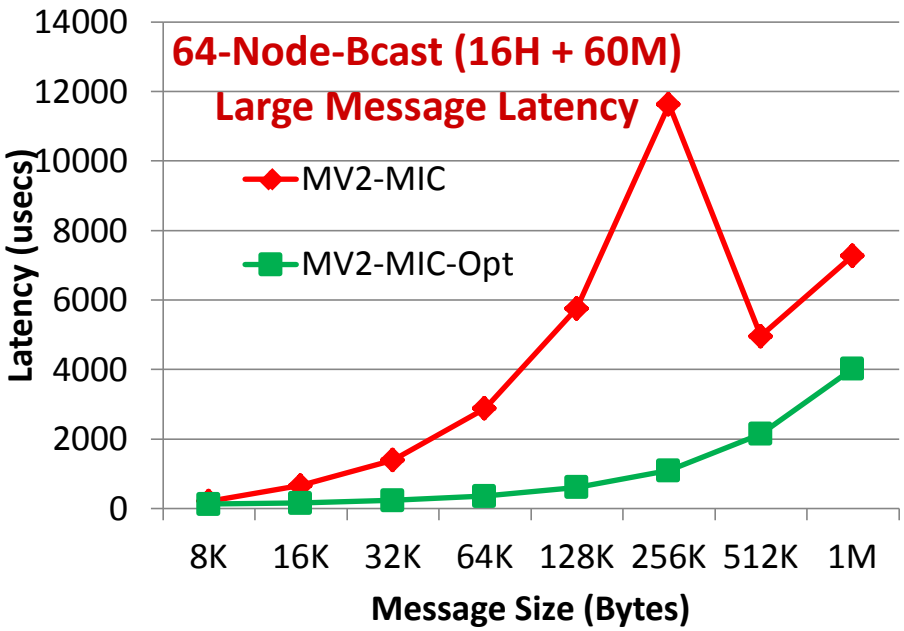
InterNode – Symmetric Mode - P3DFFT Performance

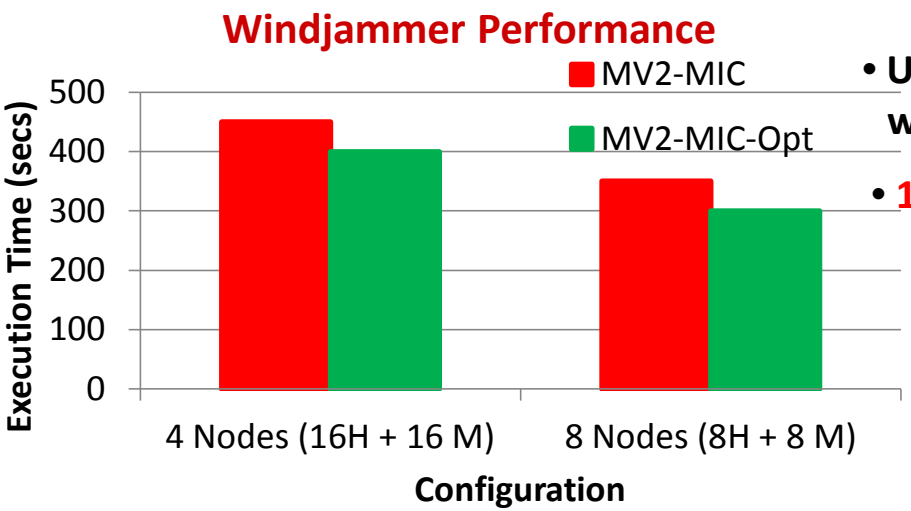


- To explore different threading levels for host and Xeon Phi

Optimized MPI Collectives for MIC Clusters (Broadcast)



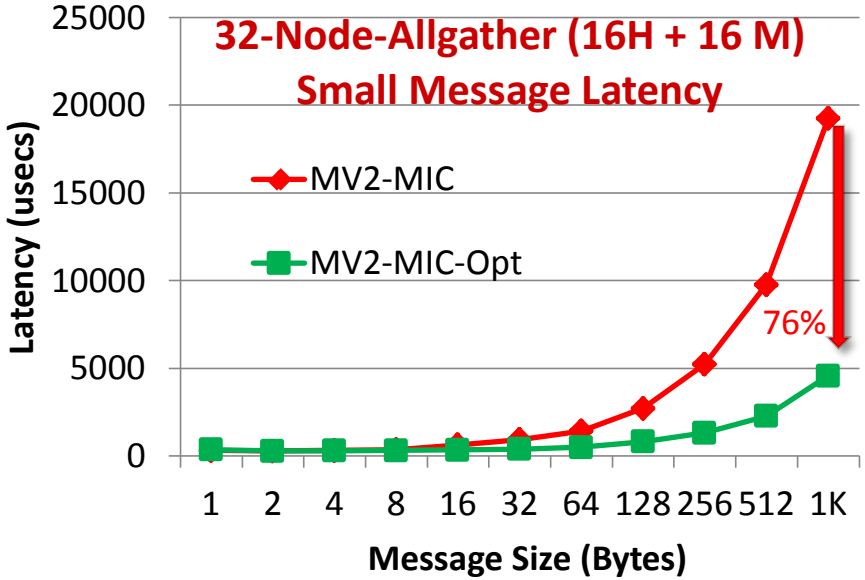


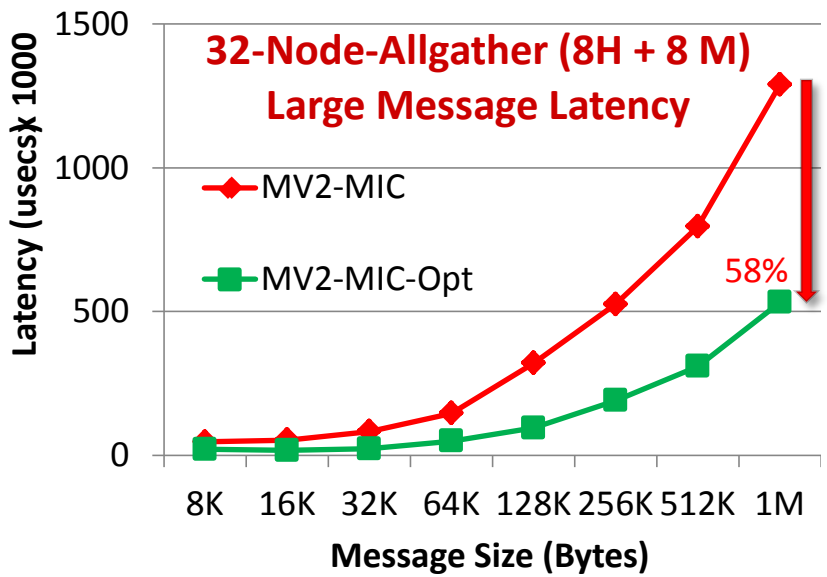


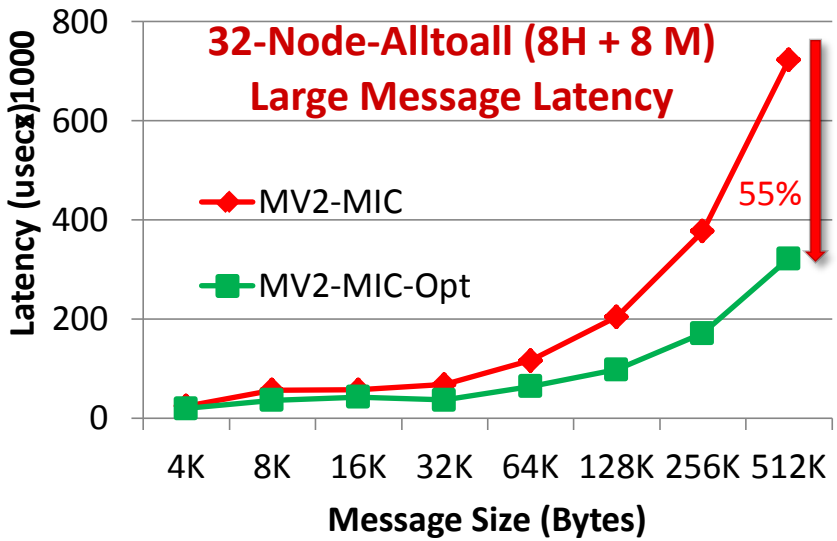
- Up to 50 - 80% improvement in MPI_Bcast latency with as many as 4864 MPI processes (Stampede)
- 12% and 16% improvement in the performance of Windjammer with 128 processes

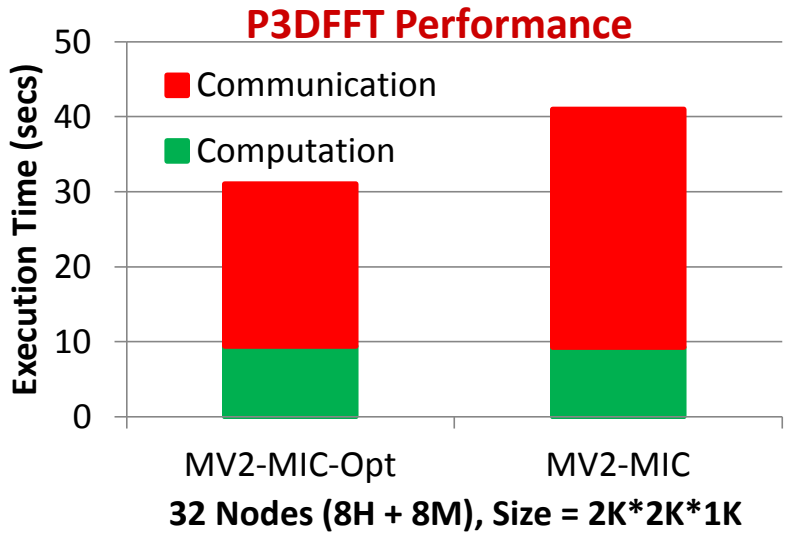
K. Kandalla , A. Venkatesh, K. Hamidouche, S. Potluri, D. Bureddy and D. K. Panda, "Designing Optimized MPI Broadcast and Allreduce for Many Integrated Core (MIC) InfiniBand Clusters; HotI'13

Optimized MPI Collectives for MIC Clusters (Allgather & Alltoall)







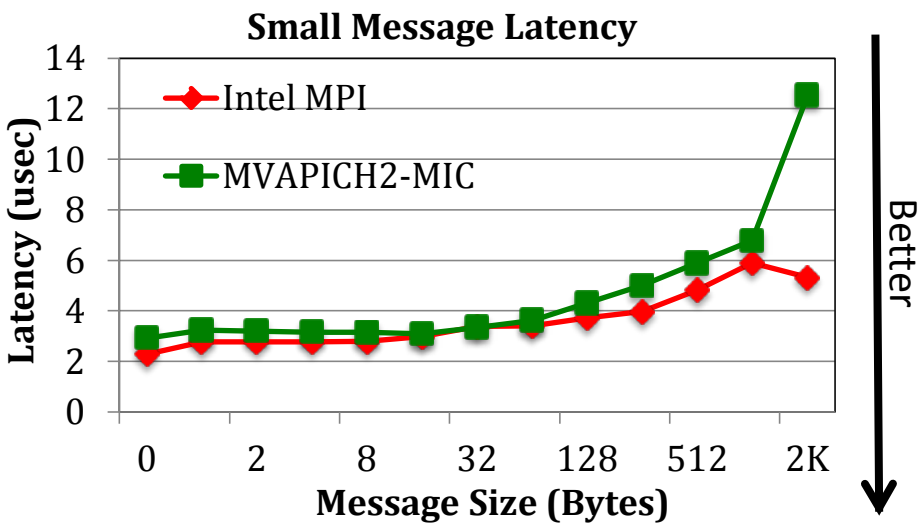


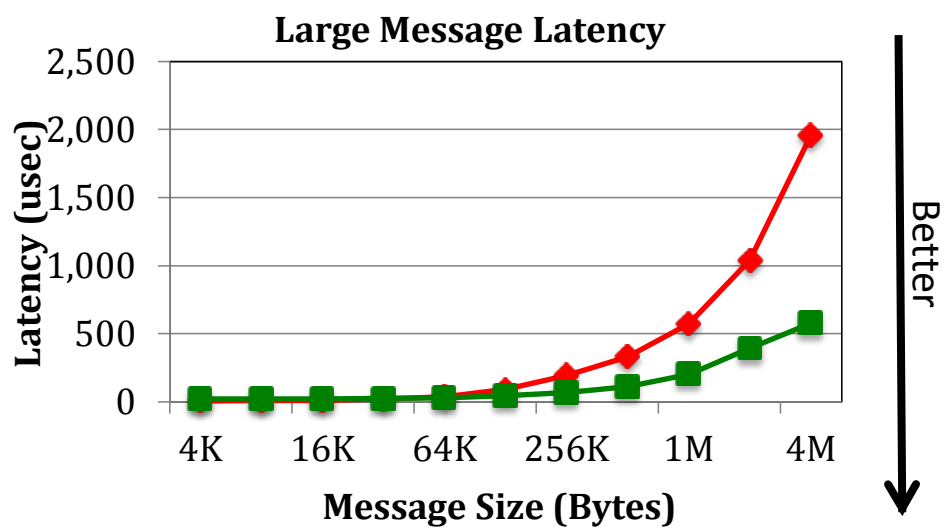
A. Venkatesh, S. Potluri, R. Rajachandrasekar, M. Luo, K. Hamidouche and D. K. Panda - High Performance Alltoall and Allgather designs for InfiniBand MIC Clusters; IPDPS'14 (accepted)

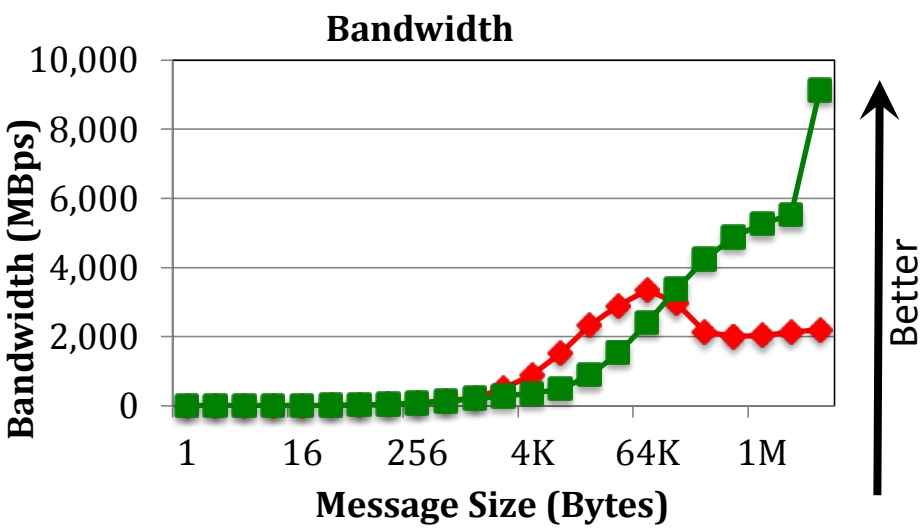
MVAPICH2-MIC Design for Clusters with IB and MIC

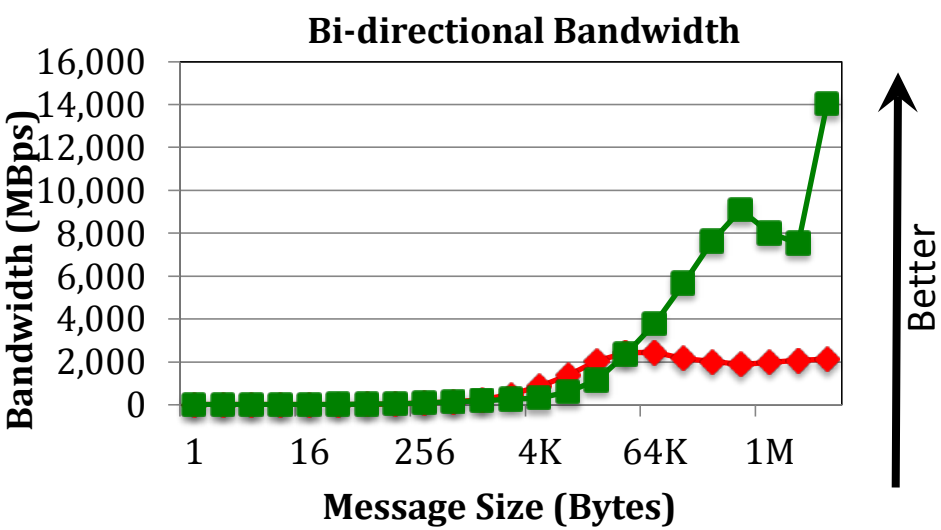
- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only Mode
  - Symmetric Mode
- Comparison with Intel's MPI Library

MVAPICH2 and Intel MPI – Intra-MIC

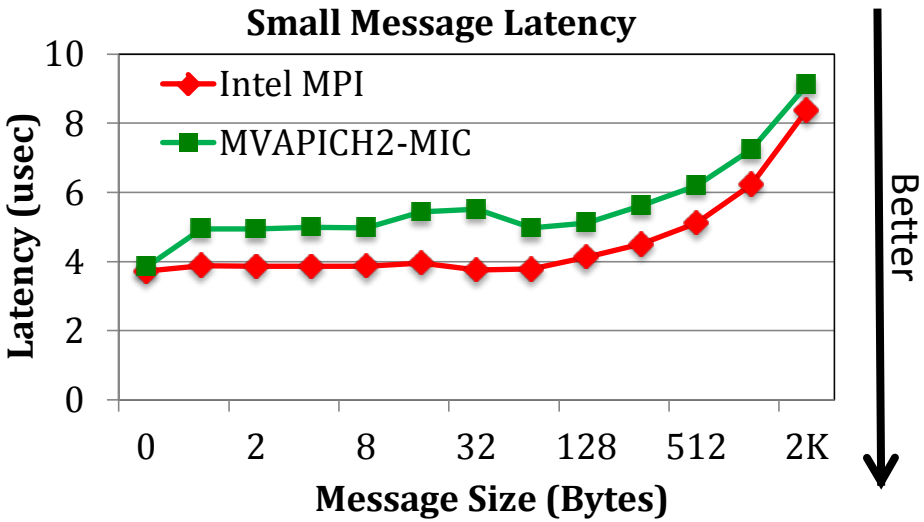


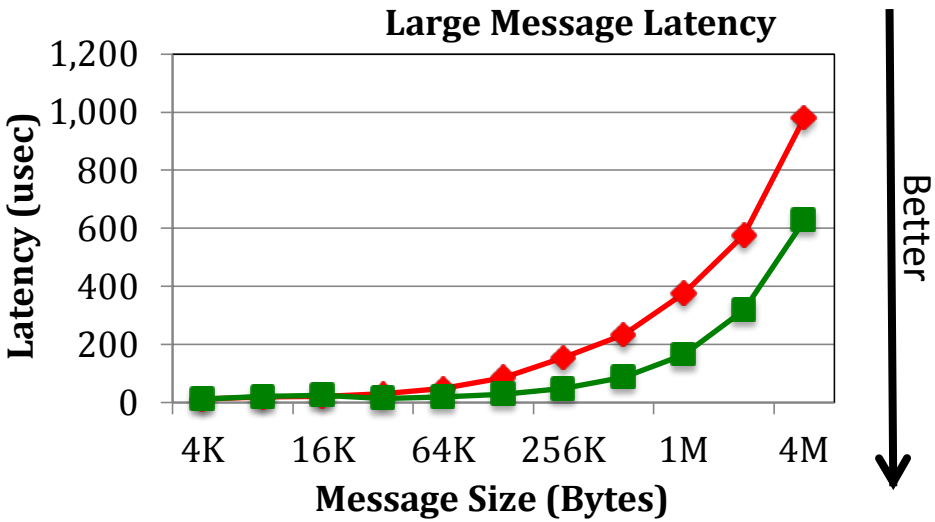


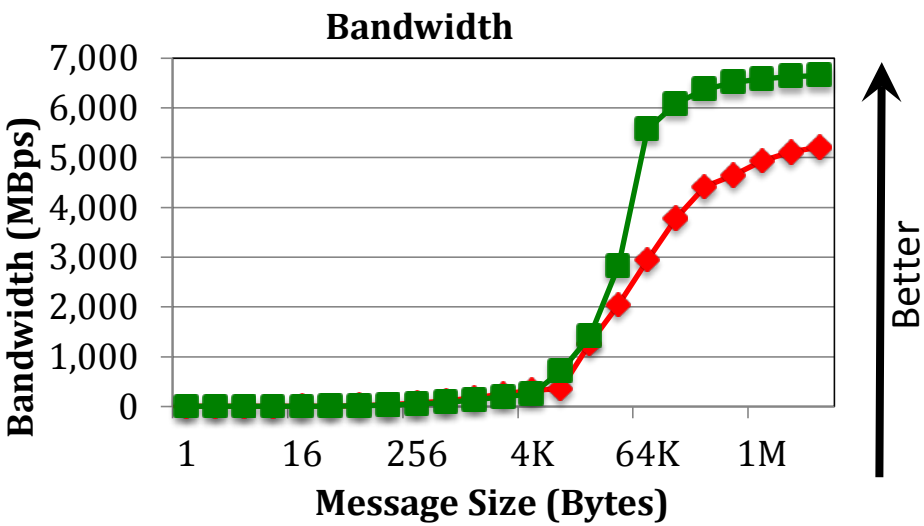


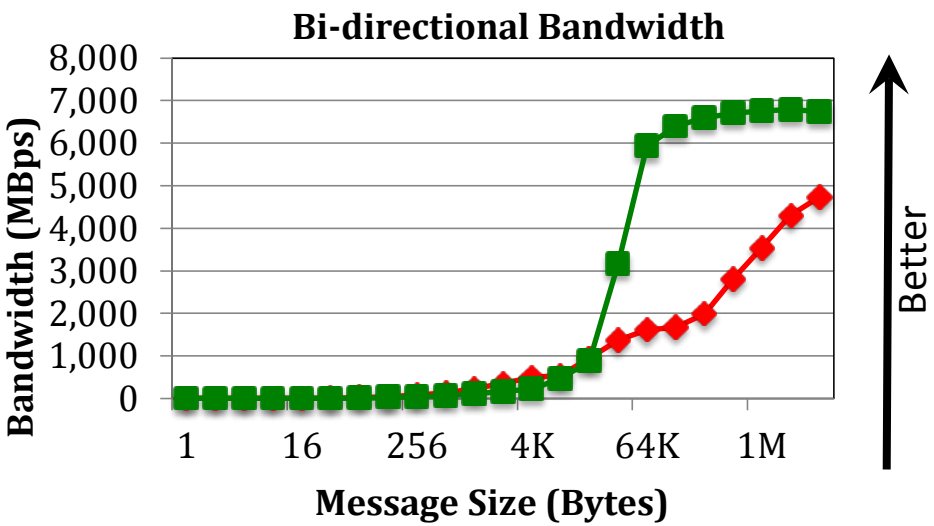


MVAPICH2 and Intel MPI – Intranode – Host-MIC



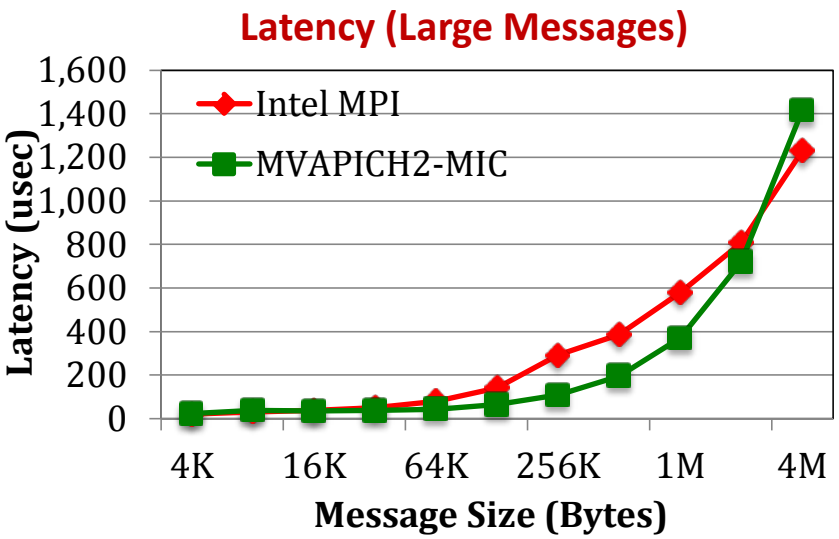


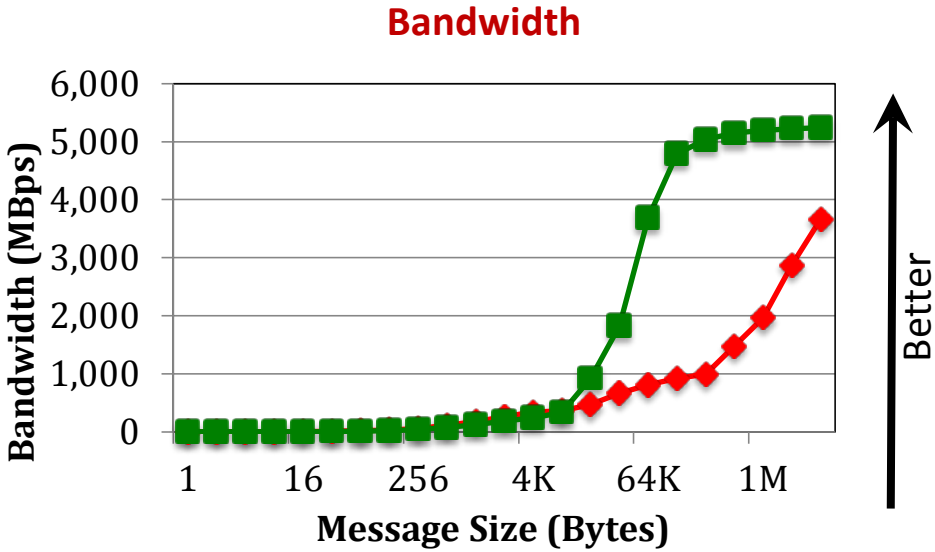




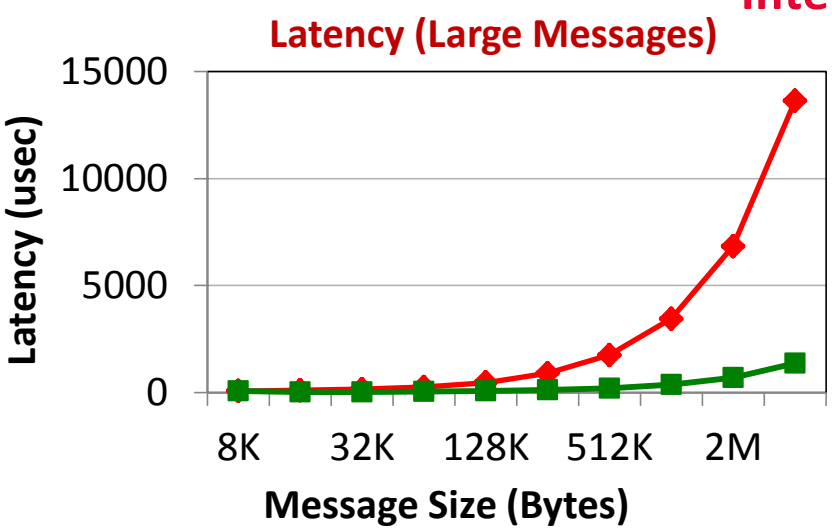
MVAPICH2 and Intel MPI – Internode MIC-MIC

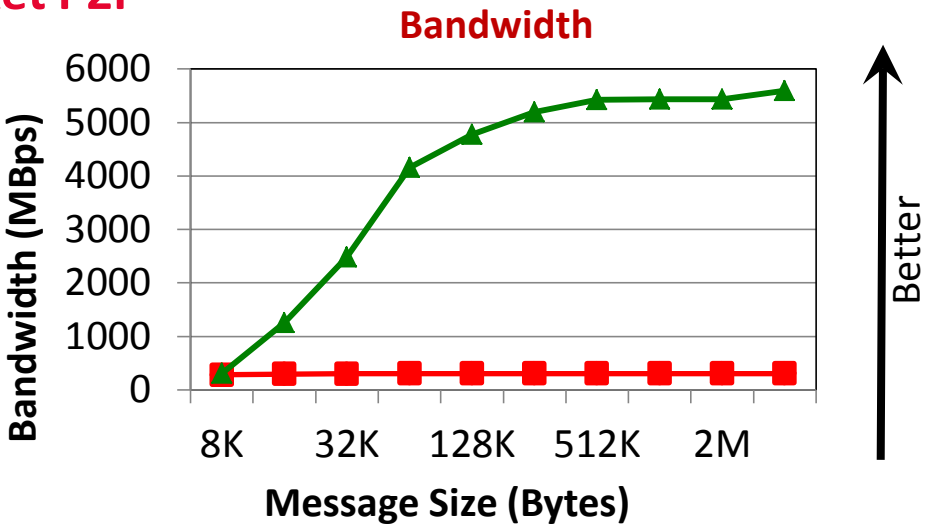
Intra-socket P2P





Inter-socket P2P





S. Potluri, D. Bureddy, K. Hamidouche, A. Venkatesh, K. Kandalla, H. Subramoni, D. K. Panda; MVAPICH-PRISM: A Proxy-Based Communication Framework Using InfiniBand and SCIF for Intel MIC Clusters; SC'13

Status of MVAPICH2-MIC

- An initial version is installed on TACC-Stampede for advanced users
- Based on the feedback, it is being enhanced
- Will be made available to the public in the near future

Hands-on Exercises

PDF version of exercises available from

http://www.cse.ohio-state.edu/~panda/vssce14/dk_karen_day3_exercises.pdf

Exercise #1

- Measure the latency between Device-to-Device, Host-to-Device, and Host-to-Host using OMB micro benchmark.

- Additional compilation flag:
  - D_ENABLE_CUDA_

- Running Parameters:
  - D D (D2D), H D (H2D), H H (H2H)

Exercise #2

- Implement the basic point-to-point communication using MPI_Send/MPI_Recv across device buffer on each GPU node
  - Naïve Design
    - cudaMemcpy(D2H)
    - MPI_Send/Recv(H2H)
    - cudaMemcpy(H2D)
  - CUDA-Aware Design
    - MPI_Send/Recv(D2D)

Exercise #3

- Design a program to carry out vector addition using CUDA-aware MPI. Basic steps will be as follows:
  1. Process 1 initializes vectors a and b on host buffer, and sends them to device buffer on process 2
     - MPI_Send(a, …), MPI_Send(b, …)
  2. Process 2 then launches the kernel to add the vectors
     - __global__ void vecAdd(float *a, float *b, float *c, int vecsize)
  3. Process 2 sends the result vector c back to process
     - MPI_Send(c,…)

Solutions and Guidelines

- Solutions for these exercises available at:
  /nfs/02/w557091/mpi-gpu-exercises/

- See README file inside the above folder for Build and Run instructions