Programming Models for Exascale Systems

Talk at HPC Advisory Council Switzerland Conference (2014)

by

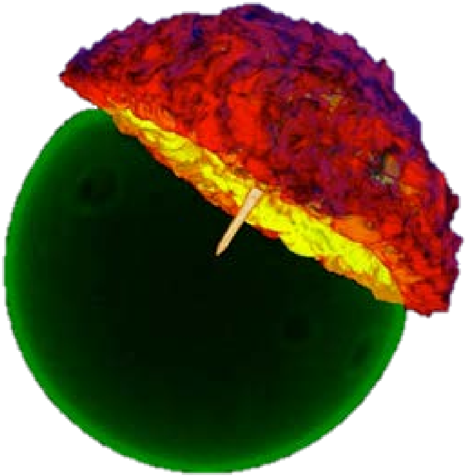
Dhabaleswar K. (DK) Panda

The Ohio State University

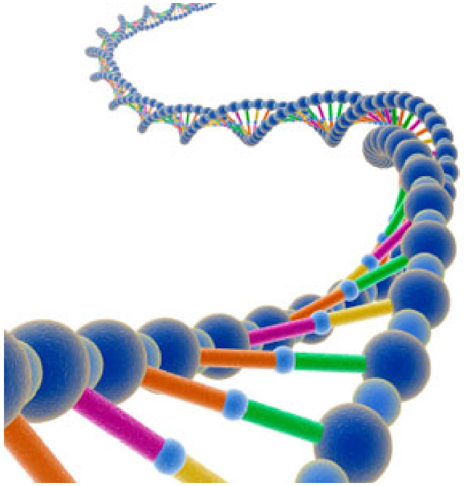
E-mail: panda@cse.ohio-state.edu

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

Current and Next Generation HPC Systems and Applications



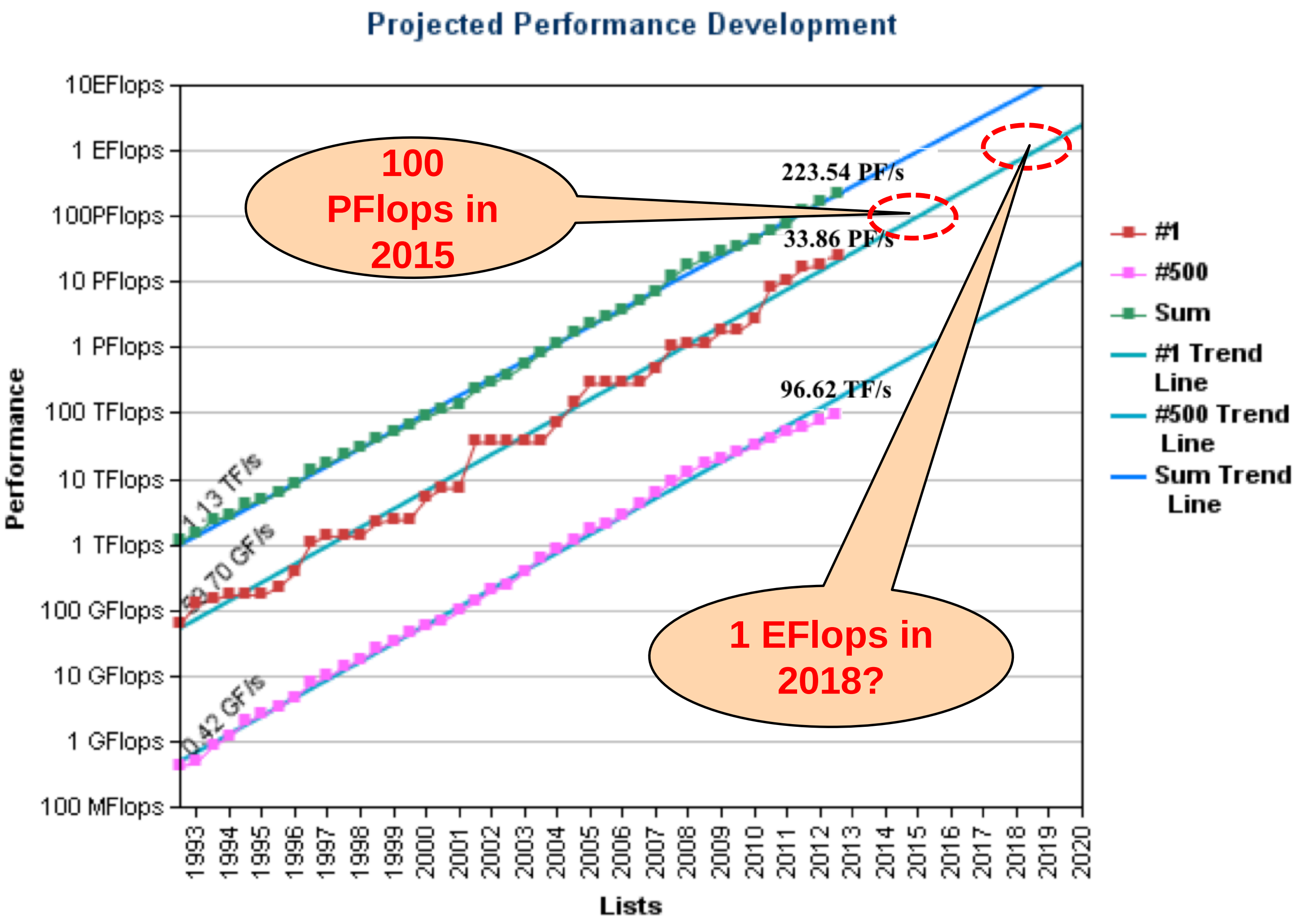
- Growth of High Performance Computing (HPC)
  - Growth in processor performance
    - Chip density doubles every 18 months
  - Growth in commodity networking
    - Increase in speed/features + reducing cost



Two Major Categories of Applications

- Scientific Computing
  - Message Passing Interface (MPI), including MPI + OpenMP, is the Dominant Programming Model
  - Many discussions towards Partitioned Global Address Space (PGAS)
    - UPC, OpenSHMEM, CAF, etc.
  - Hybrid Programming: MPI + PGAS (OpenSHMEM, UPC)
- Big Data/Enterprise/Commercial Computing
  - Focuses on large data and data analysis
  - Hadoop (HDFS, HBase, MapReduce) environment is gaining a lot of momentum
  - Memcached is also used for Web 2.0

High-End Computing (HEC): PetaFlop to ExaFlop

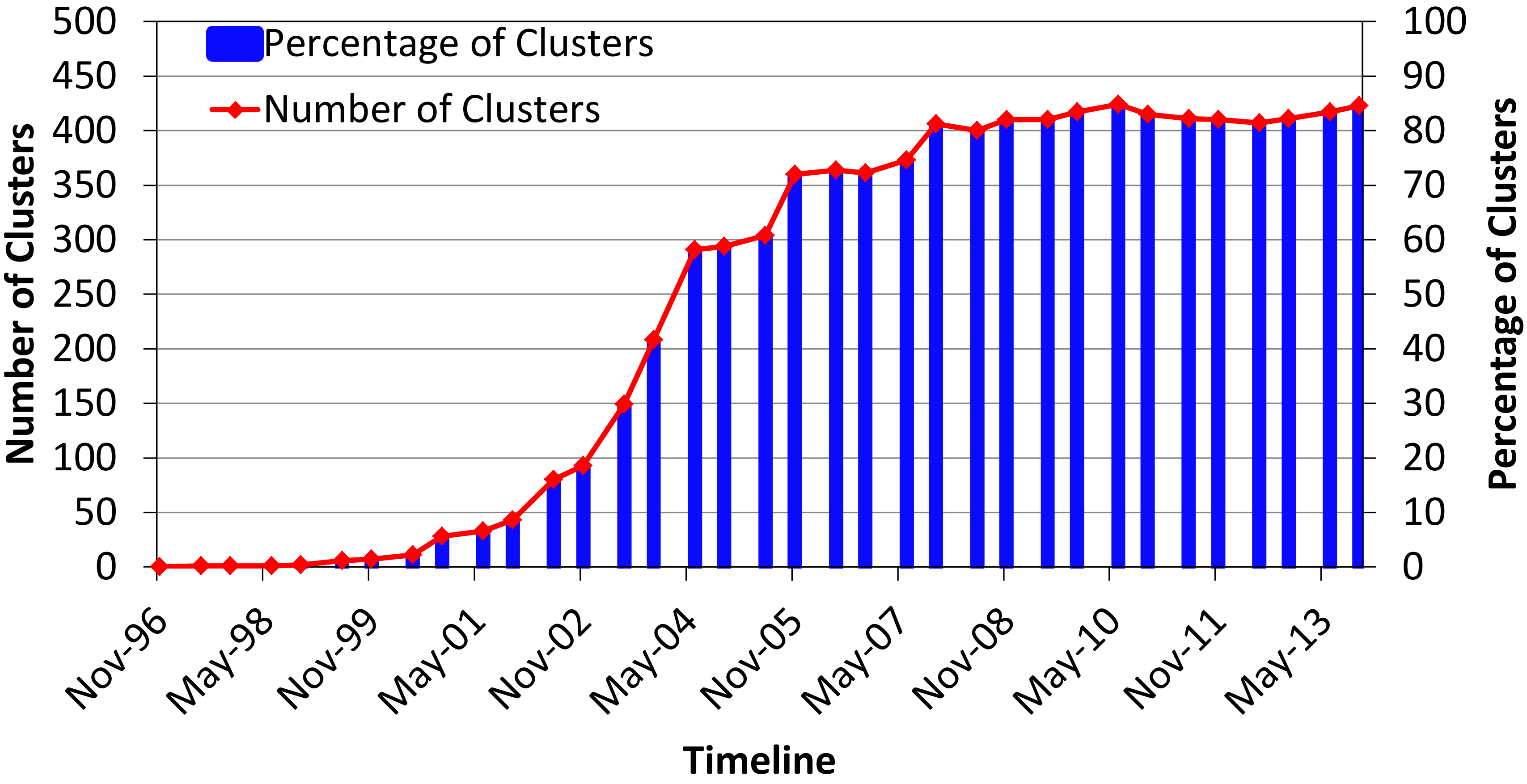


100 PFlops in 2015

1 EFlops in 2018?

Expected to have an ExaFlop system in 2020-2022!

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 (1)

Titan (2)

Stampede (6)

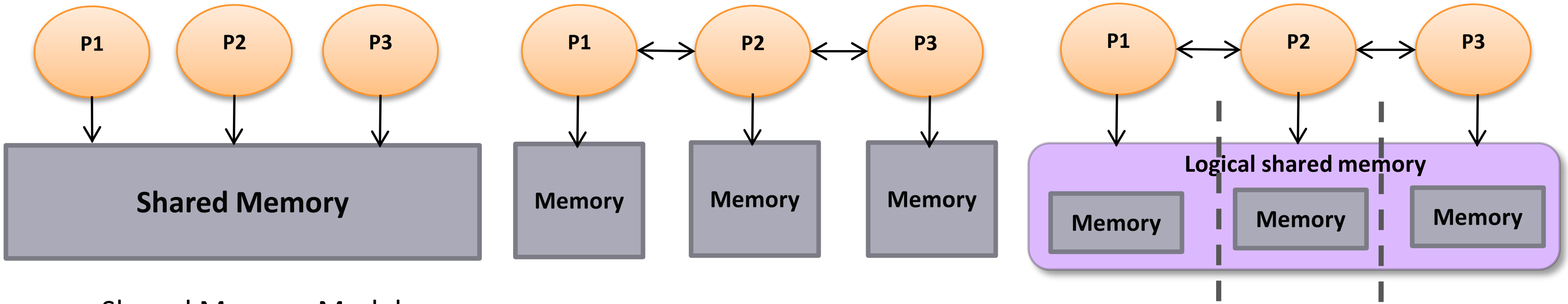
Tianhe – 1A (10)

Large-scale InfiniBand Installations

- 207 IB Clusters (41%) in the November 2013 Top500 list (http://www.top500.org)
- Installations in the Top 40 (19 systems):

462,462 cores (Stampede) at TACC (7th)	70,560 cores (Helios) at Japan/IFERC (24th)
147, 456 cores (Super MUC) in Germany (10th)	138,368 cores (Tera-100) at France/CEA (29th)
74,358 cores (Tsubame 2.5) at Japan/GSIC (11th)	60,000-cores, iDataPlex DX360M4 at Germany/Max-Planck (31st)
194,616 cores (Cascade) at PNNL (13th)	53,504 cores (PRIMERGY) at Australia/NCI (32nd)
110,400 cores (Pangea) at France/Total (14th)	77,520 cores (Conte) at Purdue University (33rd)
96,192 cores (Pleiades) at NASA/Ames (16th)	48,896 cores (MareNostrum) at Spain/BSC (34th)
73,584 cores (Spirit) at USA/Air Force (18th)	222,072 (PRIMERGY) at Japan/Kyushu (36th)
77,184 cores (Curie thin nodes) at France/CEA (20th)	78,660 cores (Lomonosov) in Russia (37th)
120, 640 cores (Nebulae) at China/NSCS (21st)	137,200 cores (Sunway Blue Light) in China 40th)
72,288 cores (Yellowstone) at NCAR (22nd)	**and many more!**

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.
- In this presentation, we concentrate on MPI first, then on PGAS and Hybrid MPI+PGAS

MPI Overview and History

- Message Passing Library standardized by MPI Forum
  - C and Fortran
- Goal: portable, efficient and flexible standard for writing parallel applications
- Not IEEE or ISO standard, but widely considered “industry standard” for HPC application
- Evolution of MPI
  - MPI-1: 1994
  - MPI-2: 1996
  - MPI-3.0: 2008 – 2012, standardized before SC ‘12
  - Next plans for MPI 3.1, 3.2, ….

Major MPI Features

- Point-to-point Two-sided Communication
- Collective Communication
- One-sided Communication
- Job Startup
- Parallel I/O

Towards Exascale System (Today and Target)

Systems	2014 Tianhe-2	2020-2022	Difference Today & Exascale
System peak	55 PFlop/s	1 EFlop/s	~20x
Power	18 MW (3 Gflops/W)	~20 MW (50 Gflops/W)	O(1) ~15x
System memory	1.4 PB (1.024PB CPU + 0.384PB CoP)	32 – 64 PB	~50X
Node performance	3.43TF/s (0.4 CPU + 3 CoP)	1.2 or 15 TF	O(1)
Node concurrency	24 core CPU + 171 cores CoP	O(1k) or O(10k)	~5x - ~50x
Total node interconnect BW	6.36 GB/s	200 – 400 GB/s	~40x -~60x
System size (nodes)	16,000	O(100,000) or O(1M)	~6x - ~60x

Total concurrency	3.12M
	12.48M threads (4 /core)
		O(billion)
		for latency hiding
			~100x

MTTI	Few/day	Many/day	O(?)

Courtesy: Prof. Jack Dongarra

Basic Design Challenges for Exascale Systems

- DARPA Exascale Report – Peter Kogge, Editor and Lead
- Energy and Power Challenge
  - Hard to solve power requirements for data movement
- Memory and Storage Challenge
  - Hard to achieve high capacity and high data rate
- Concurrency and Locality Challenge
  - Management of very large amount of concurrency (***billion*** threads)
- Resiliency Challenge
  - Low voltage devices (for low power) introduce more faults

How does MPI Plan to Meet Exascale Challenges?

- Power required for data movement operations is one of the main challenges
- Non-blocking collectives
  - Overlap computation and communication
- Much improved One-sided interface
  - Reduce synchronization of sender/receiver
- Manage concurrency
  - Improved interoperability with PGAS (e.g. UPC, Global Arrays, OpenSHMEM)
- Resiliency
  - New interface for detecting failures

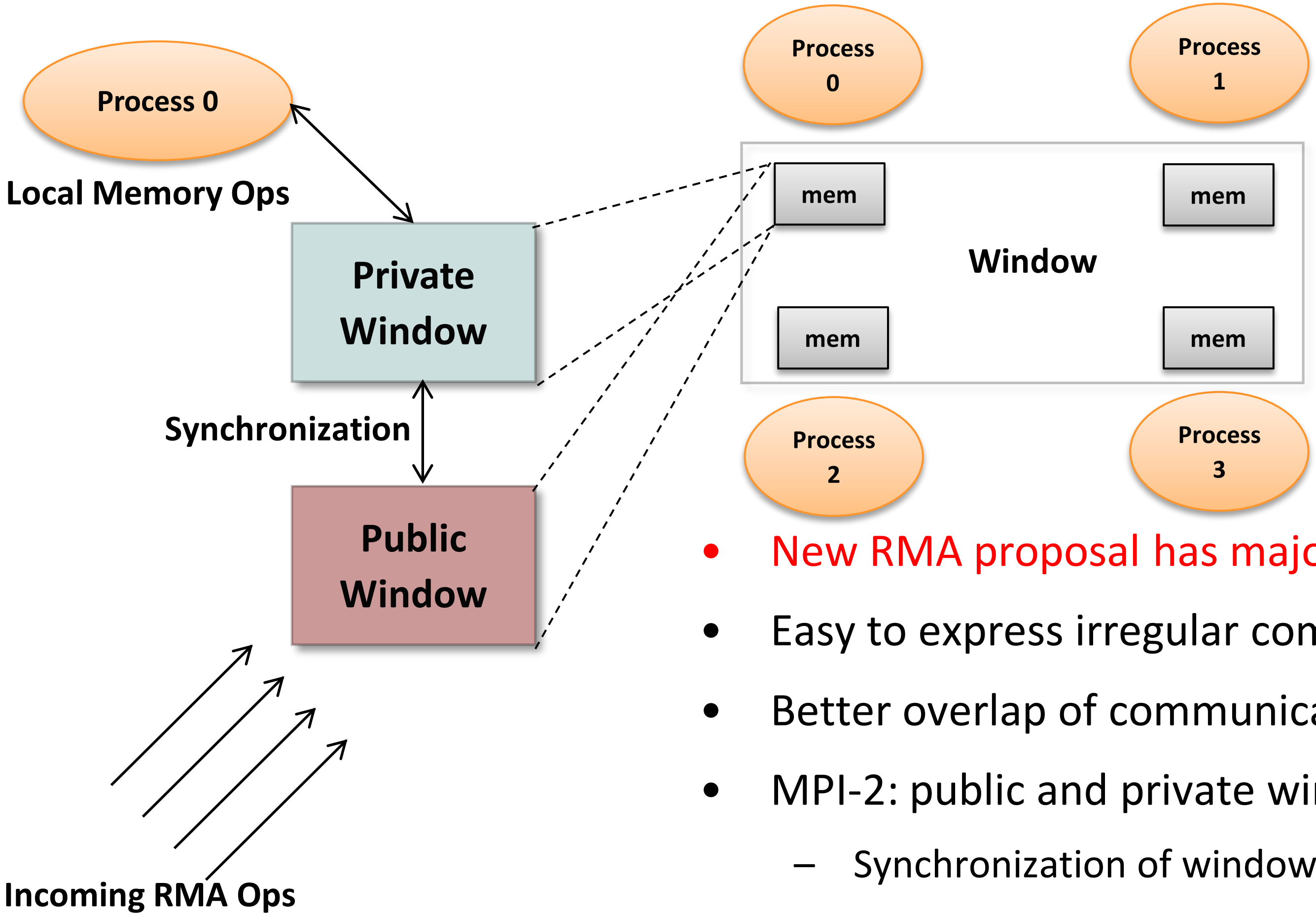
Major New Features in MPI-3

- Major features
  - Non-blocking Collectives
  - Improved One-Sided (RMA) Model
  - MPI Tools Interface
- Specification is available from: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

Non-blocking Collective Operations

- Enables overlap of computation with communication
- Removes synchronization effects of collective operations (exception of barrier)
- Non-blocking calls do not match blocking collective calls
  - MPI implementation may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: optimized for latency
  - Non-blocking collectives: optimized for overlap
- User must call collectives in same order on all ranks
- Progress rules are same as those for point-to-point
- Example new calls: MPI_Ibarrier, MPI_Iallreduce, …

Improved One-sided (RMA) Model



- New RMA proposal has major improvements
- Easy to express irregular communication pattern
- Better overlap of communication & computation
- MPI-2: public and private windows
  - Synchronization of windows explicit
- MPI-2: works for non-cache coherent systems
- MPI-3: two types of windows
  - Unified and Separate
  - Unified window leverages hardware cache coherence

MPI Tools Interface

- Extended tools support in MPI-3, beyond the PMPI interface
- Provide standardized interface (MPIT) to access MPI internal information
  - Configuration and control information
    - Eager limit, buffer sizes, . . .
  - Performance information
    - Time spent in blocking, memory usage, . . .
  - Debugging information
    - Packet counters, thresholds, . . .
- External tools can build on top of this standard interface

Partitioned Global Address Space (PGAS) Models

- Key features
  - Simple shared memory abstractions
  - Light weight one-sided communication
  - Easier to express irregular communication
- Different approaches to PGAS
  - Languages
    - Unified Parallel C (UPC)
    - Co-Array Fortran (CAF)
    - X10
  - Libraries
    - OpenSHMEM
    - Global Arrays
    - Chapel

Compiler-based: Unified Parallel C

- UPC: a parallel extension to the C standard
- UPC Specifications and Standards:
  - Introduction to UPC and Language Specification, 1999
  - UPC Language Specifications, v1.0, Feb 2001
  - UPC Language Specifications, v1.1.1, Sep 2004
  - UPC Language Specifications, v1.2, June 2005
  - UPC Language Specifications, v1.3, Nov 2013
- UPC Consortium
  - Academic Institutions: GWU, MTU, UCB, U. Florida, U. Houston, U. Maryland…
  - Government Institutions: ARSC, IDA, LBNL, SNL, US DOE…
  - Commercial Institutions: HP, Cray, Intrepid Technology, IBM, …
- Supported by several UPC compilers
  - Vendor-based commercial UPC compilers: HP UPC, Cray UPC, SGI UPC
  - Open-source UPC compilers: Berkeley UPC, GCC UPC, Michigan Tech MuPC
- Aims for: high performance, coding efficiency, irregular applications, …

OpenSHMEM

- SHMEM implementations – Cray SHMEM, SGI SHMEM, Quadrics SHMEM, HP SHMEM, GSHMEM
- Subtle differences in API, across versions – example:

	SGI SHMEM	Quadrics SHMEM	Cray SHMEM
Initialization	*start_pes(0)*	*shmem_init*	*start_pes*
Process ID	*_my_pe*	*my_pe*	*shmem_my_pe*

- Made applications codes non-portable
- OpenSHMEM is an effort to address this:

"A new, open specification to consolidate the various extant SHMEM versions into a widely accepted standard." – OpenSHMEM Specification v1.0

by University of Houston and Oak Ridge National Lab

SGI SHMEM is the baseline

MPI+PGAS for Exascale Architectures and Applications

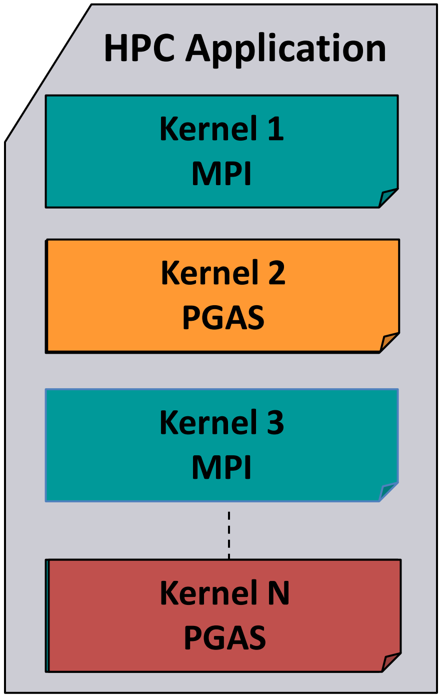
- Hierarchical architectures with multiple address spaces
- (MPI + PGAS) Model
  - MPI across address spaces
  - PGAS within an address space
- MPI is good at moving data between address spaces
- Within an address space, MPI can interoperate with other shared memory programming models

- Applications can have kernels with different communication patterns
- Can benefit from different models

- Re-writing complete applications can be a huge effort
- Port critical kernels to the desired model instead

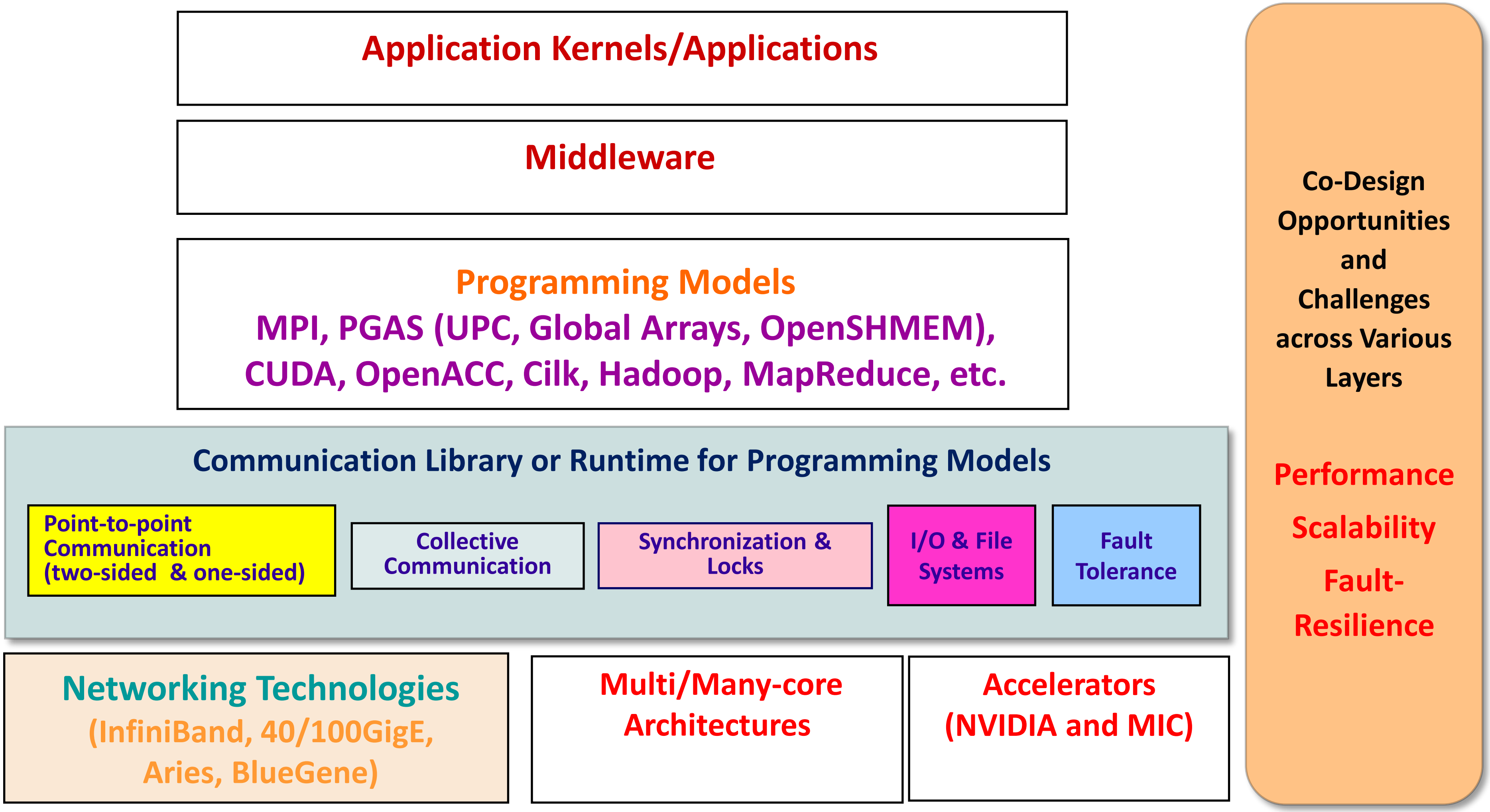
Hybrid (MPI+PGAS) Programming

- Application sub-kernels can be re-written in MPI/PGAS based on communication characteristics
- Benefits:
  - Best of Distributed Computing Model
  - Best of Shared Memory Computing Model
- Exascale Roadmap*:
  - "Hybrid Programming is a practical way to program exascale systems"



** The International Exascale Software Roadmap, Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420*

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



Application Kernels/Applications

Middleware

Programming Models

MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenACC, Cilk, Hadoop, MapReduce, etc.

Communication Library or Runtime for Programming Models

Point-to-point Communication (two-sided & one-sided)

Collective Communication

Synchronization & Locks

I/O & File Systems

Fault Tolerance

Networking Technologies (InfiniBand, 40/100GigE, Aries, BlueGene)

Multi/Many-core Architectures

Accelerators (NVIDIA and MIC)

Co-Design Opportunities and Challenges across Various Layers

Performance

Scalability

Fault-Resilience

Challenges in Designing (MPI+X) at Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Balancing intra-node and inter-node communication for next generation multi-core (128-1024 cores/node)
  - Multiple end-points per node
- Support for efficient multi-threading
- Support for GPGPUs and Accelerators
- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
  - Power-aware
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, …)

Additional Challenges for Designing Exascale Middleware

- **Extreme Low Memory Footprint**
  - Memory per core continues to decrease
- **D-L-A Framework**
  - **D**iscover
    - Overall network topology (fat-tree, 3D, …)
    - Network topology for processes for a given job
    - Node architecture
    - Health of network and node
  - **L**earn
    - Impact on performance and scalability
    - Potential for failure
  - **A**dapt
    - Internal protocols and algorithms
    - Process mapping
    - Fault-tolerance solutions
  - Low overhead techniques while delivering performance, scalability and fault-tolerance

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
  - Support for GPGPUs and MIC
  - **Used by more than 2,150 organizations in 72 countries**
  - **More than 206,000 downloads from OSU site directly**
  - Empowering many TOP500 clusters
    - 7th ranked 462,462-core cluster (Stampede) at TACC
    - 11th ranked 74,358-core cluster (Tsubame 2.5) at Tokyo Institute of Technology
    - 16th ranked 96,192-core cluster (Pleiades) at NASA
    - 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
- Partner in the U.S. NSF-TACC Stampede System

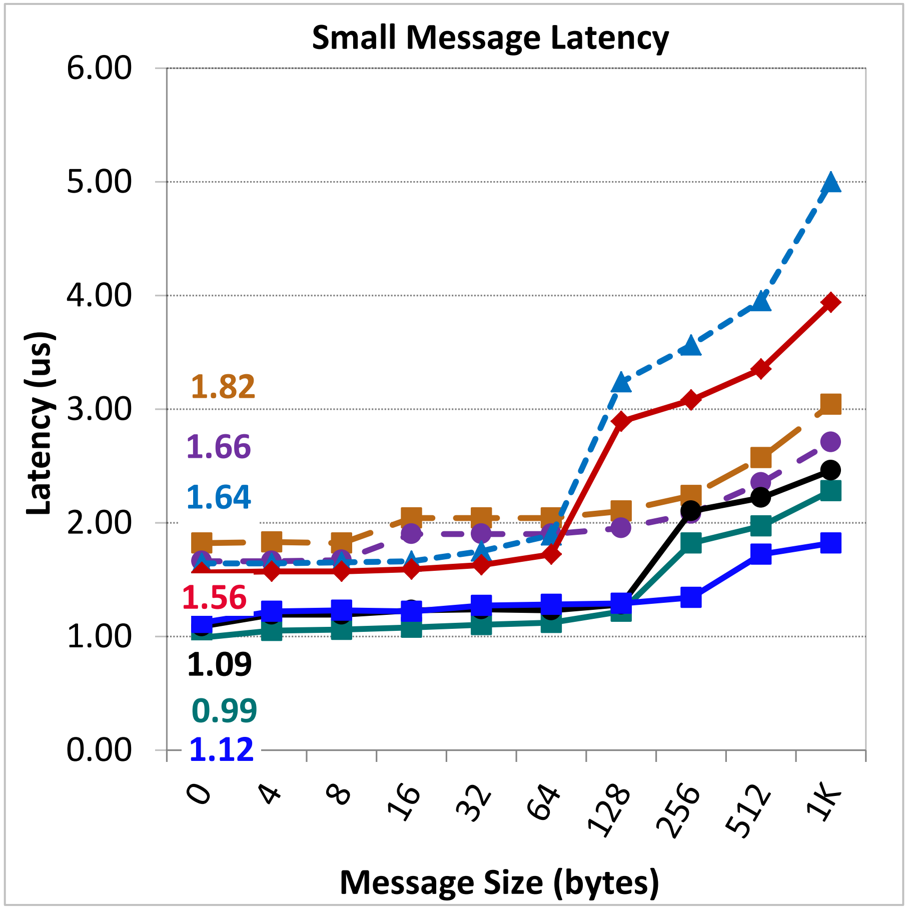
MVAPICH2 2.0RC1 and MVAPICH2-X 2.0RC1

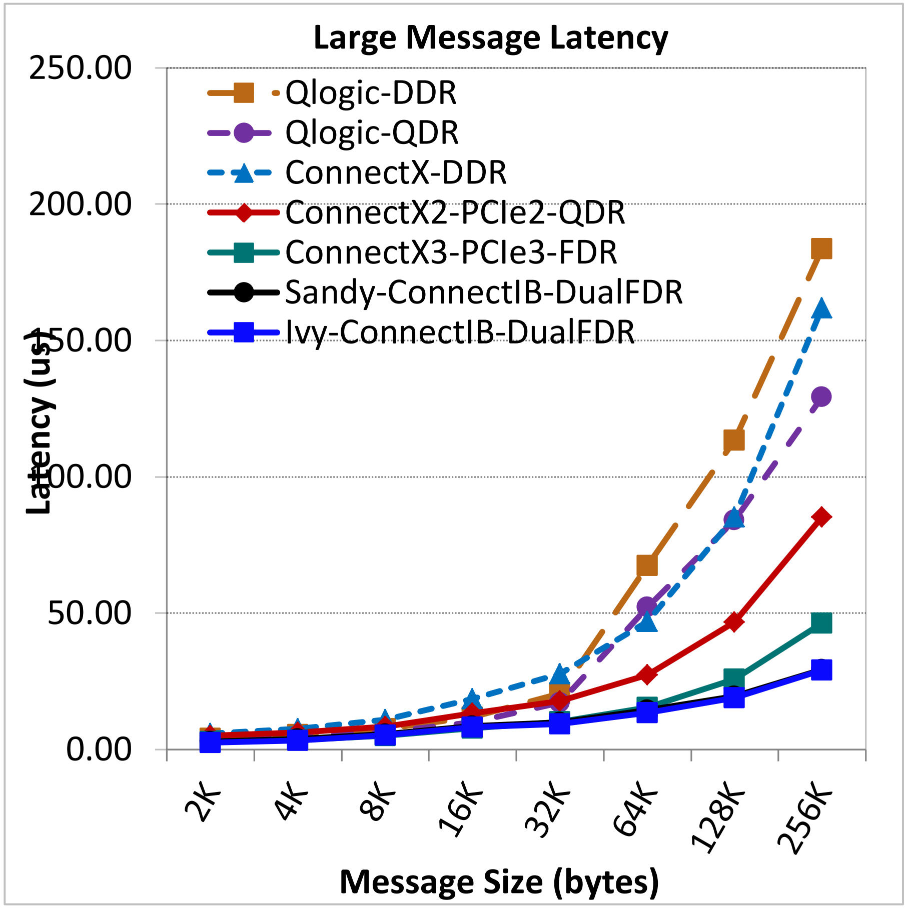
- Released on 03/24/14
- Major Features and Enhancements
  - Based on MPICH-3.1
  - Improved performance for MPI_Put and MPI_Get operations in CH3 channel
  - Enabled MPI-3 RMA support in PSM channel
  - Enabled multi-rail support for UD-Hybrid channel
  - Optimized architecture based tuning for blocking and non-blocking collectives
  - Optimized bcast and reduce collectives designs
  - Improved hierarchical job startup time
  - Optimization for sub-array data-type processing for GPU-to-GPU communication
  - Updated hwloc to version 1.8
- MVAPICH2-X 2.0RC1 supports hybrid MPI + PGAS (UPC and OpenSHMEM) programming models
  - Based on MVAPICH2 2.0RC1 including MPI-3 features; Compliant with UPC 2.18.0 and OpenSHMEM v1.0f
  - Improved intra-node performance using Shared memory and Cross Memory Attach (CMA)
  - Optimized UPC collectives

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Fault-tolerance
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime

One-way Latency: MPI over IB with MVAPICH2





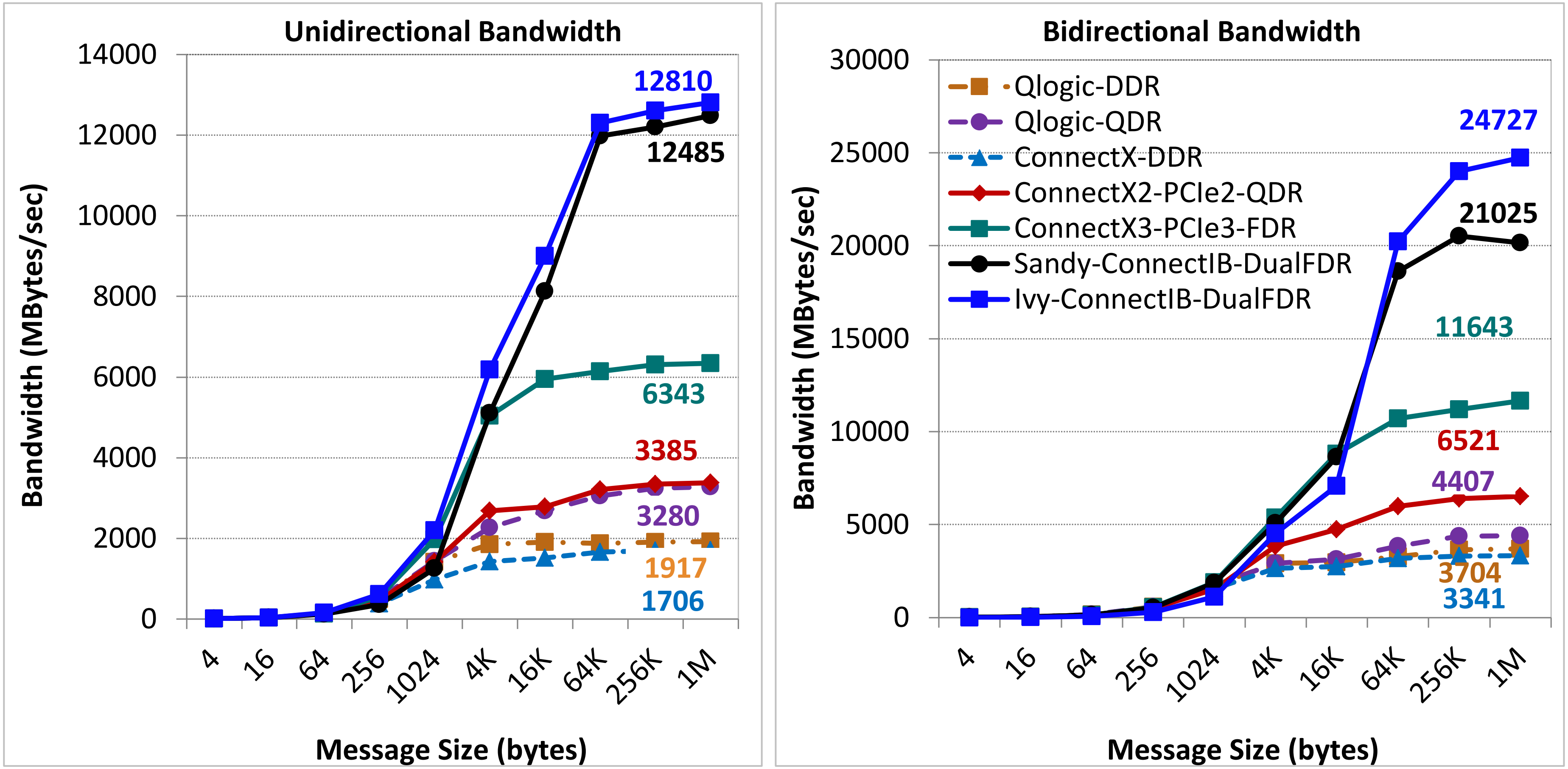
DDR, QDR - 2.4 GHz Quad-core (Westmere) Intel PCI Gen2 with IB switch

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



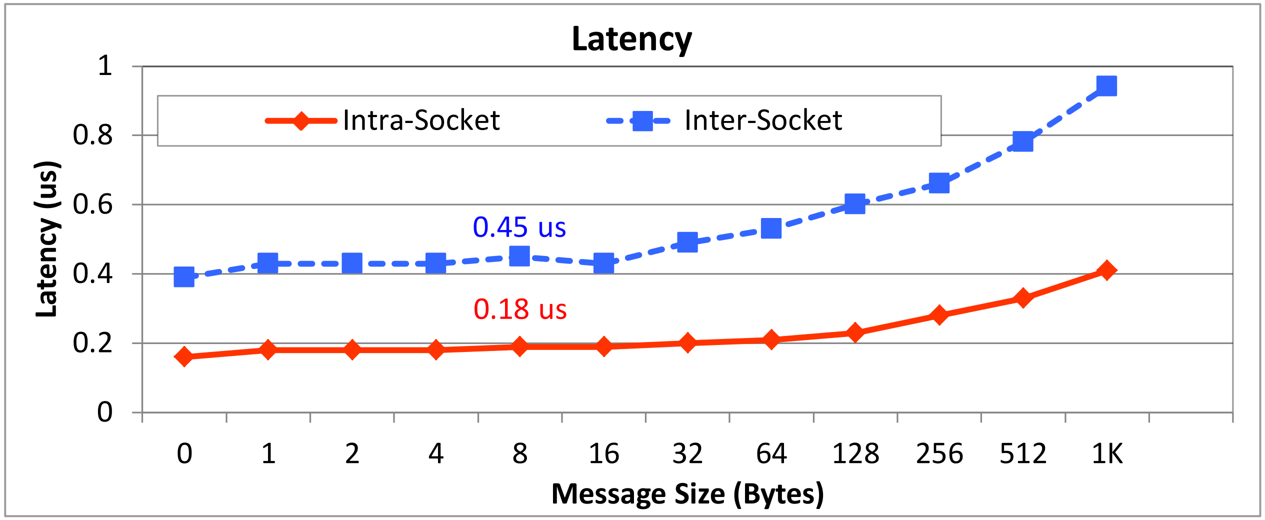
DDR, QDR - 2.4 GHz Quad-core (Westmere) Intel PCI Gen2 with IB switch

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

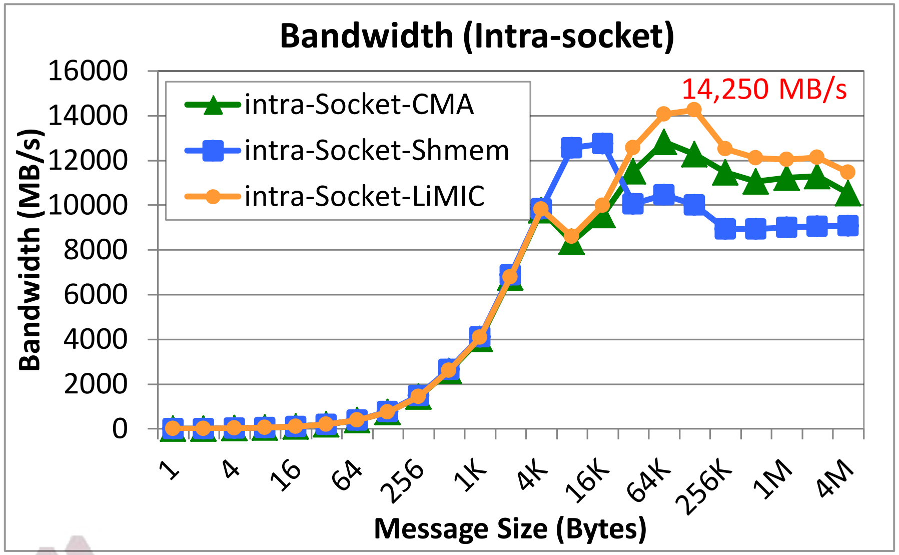
ConnectIB-Dual FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

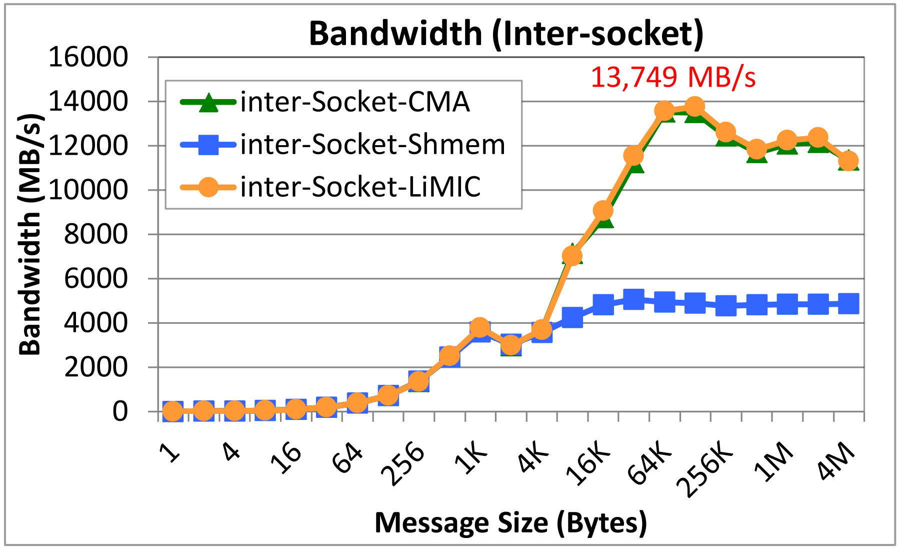
MVAPICH2 Two-Sided Intra-Node Performance
(Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA))



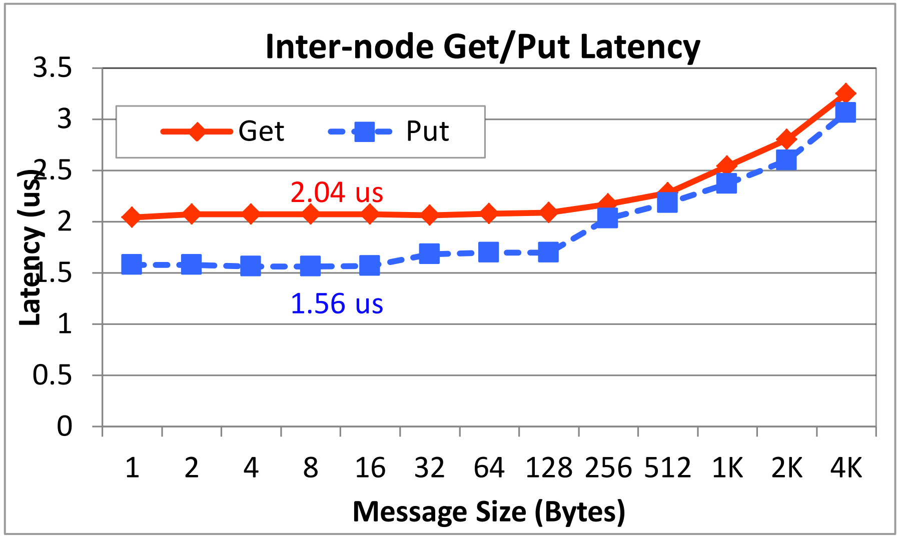
Latest MVAPICH2 2.0rc1

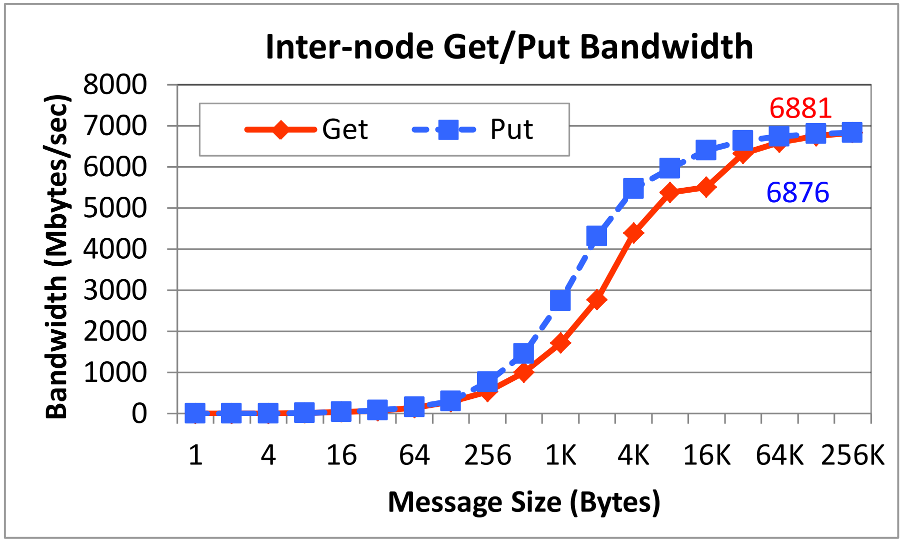
Intel Ivy-bridge

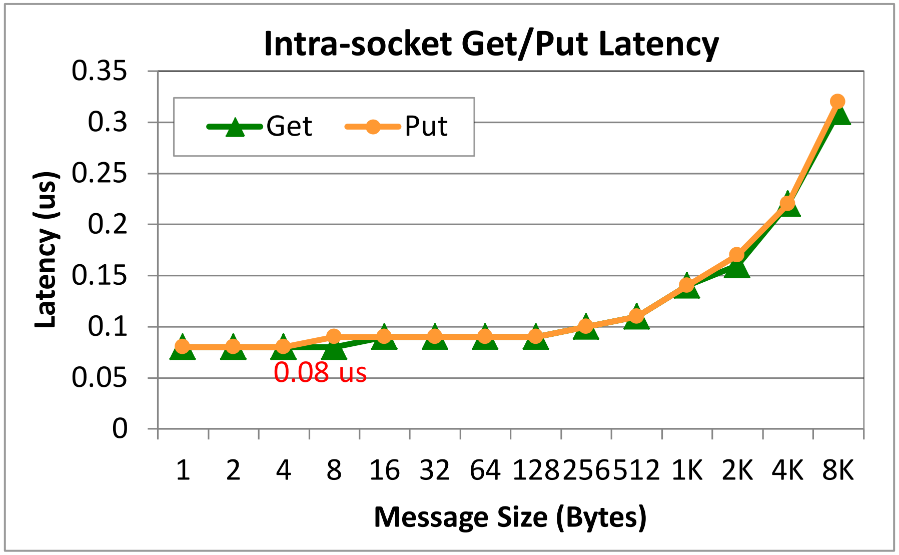


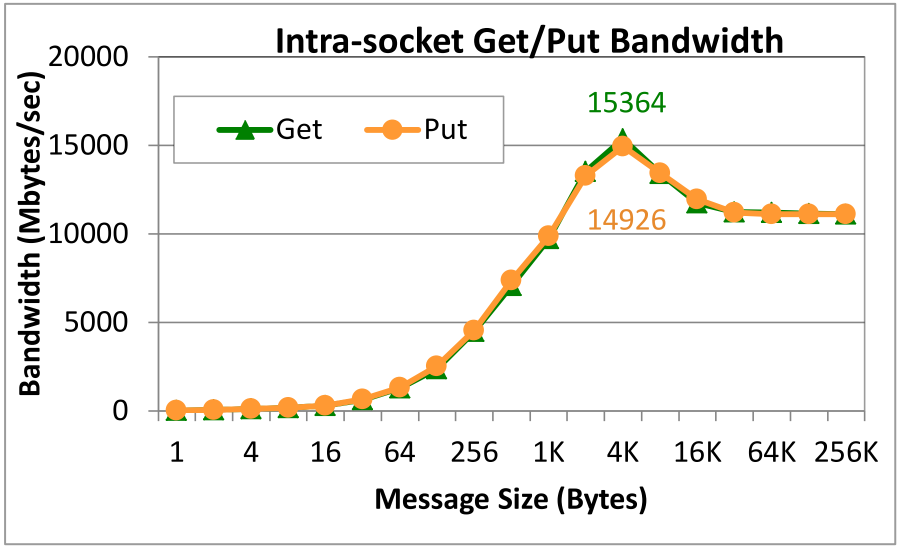


MPI-3 RMA Get/Put with Flush Performance


Inter-node Get/Put Latency

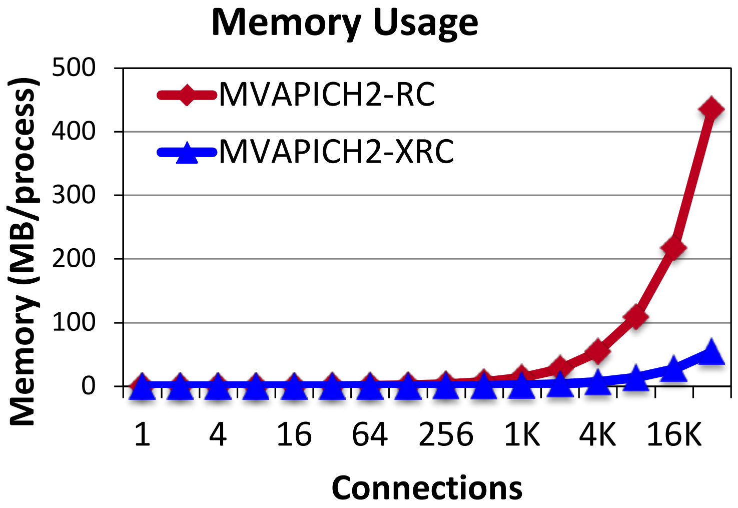

Inter-node Get/Put Bandwidth

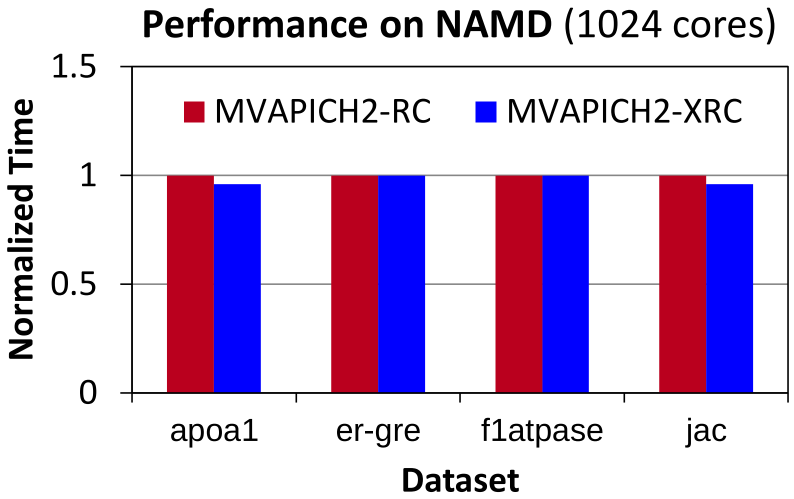

Intra-socket Get/Put Latency


Intra-socket Get/Put Bandwidth

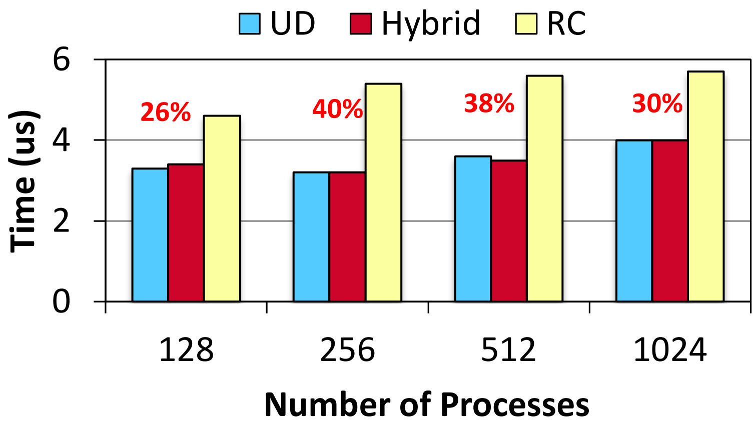
Latest MVAPICH2 2.0rc1, Intel Sandy-bridge with Connect-IB (single-port)

eXtended Reliable Connection (XRC) and Hybrid Mode





- Memory usage for 32K processes with 8-cores per node can be 54 MB/process (for connections)
- NAMD performance improves when there is frequent communication to many peers



- Both UD and RC/XRC have benefits
  - Hybrid for the best of both
- Available since MVAPICH2 1.7 as integrated interface
- Runtime Parameters: RC - default;
  - **UD - MV2_USE_ONLY_UD=1**
  - **Hybrid - MV2_HYBRID_ENABLE_THRESHOLD=1**

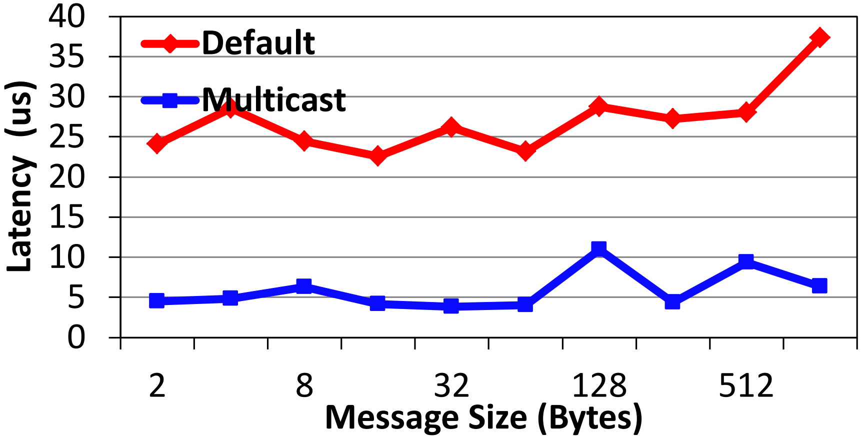
M. Koop, J. Sridhar and D. K. Panda, "Scalable MPI Design over InfiniBand using eXtended Reliable Connection," Cluster '08

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

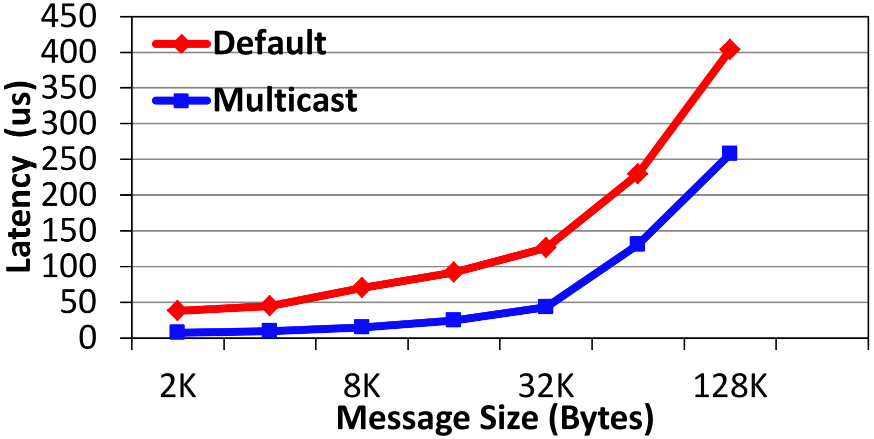
- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Fault-tolerance
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime

Hardware Multicast-aware MPI_Bcast on Stampede

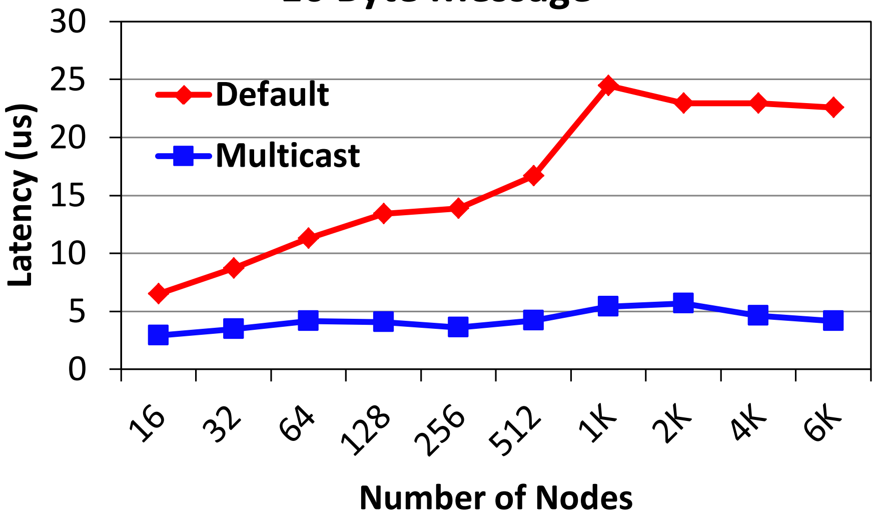
Small Messages (102,400 Cores)



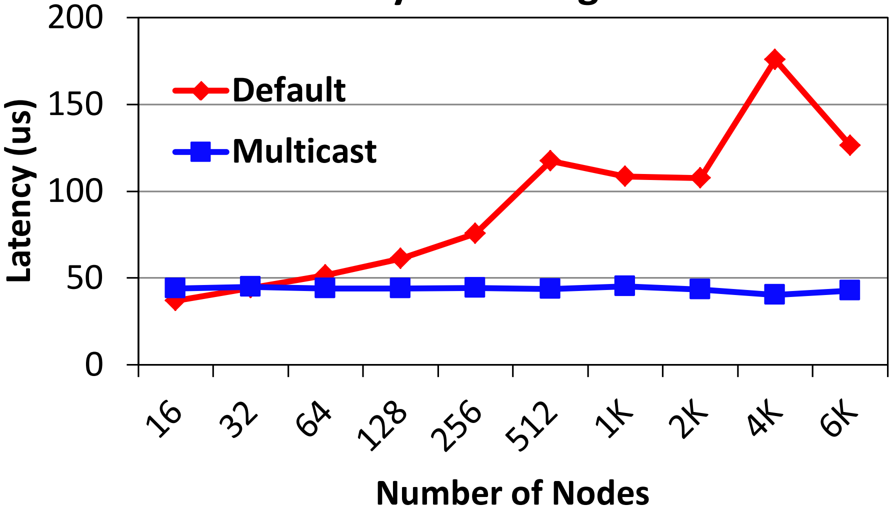
Large Messages (102,400 Cores)



16 Byte Message

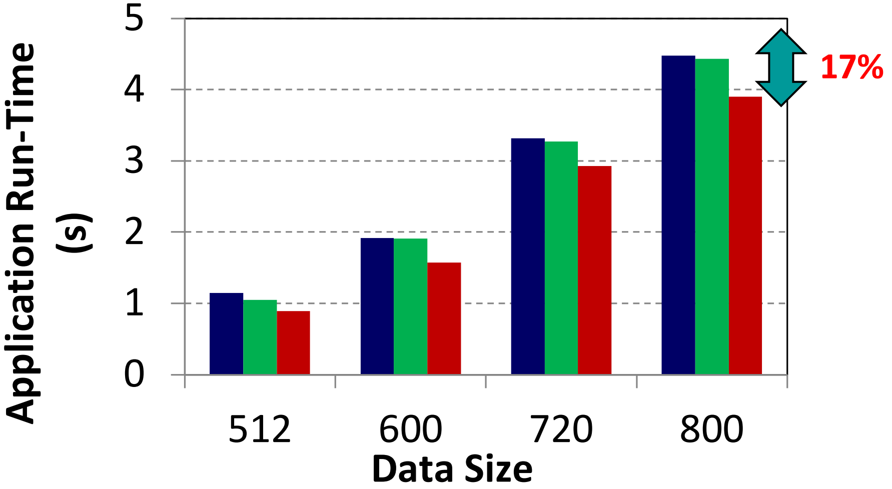


32 KByte Message

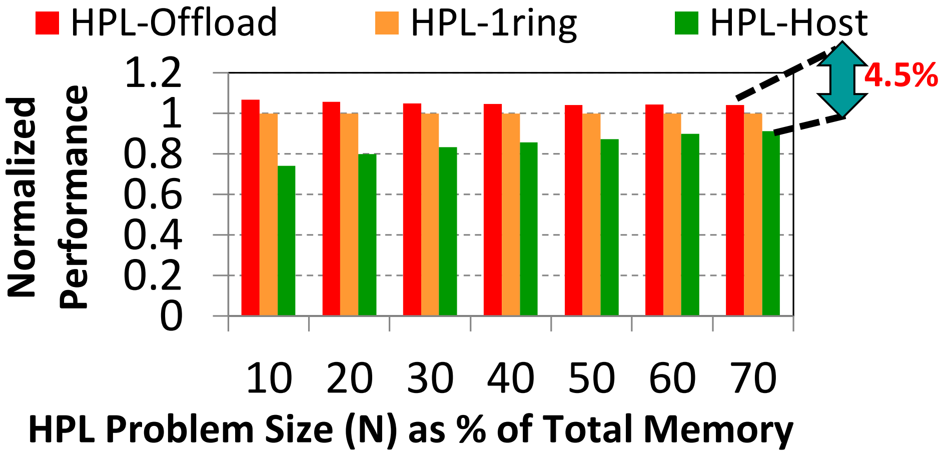


ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR switch

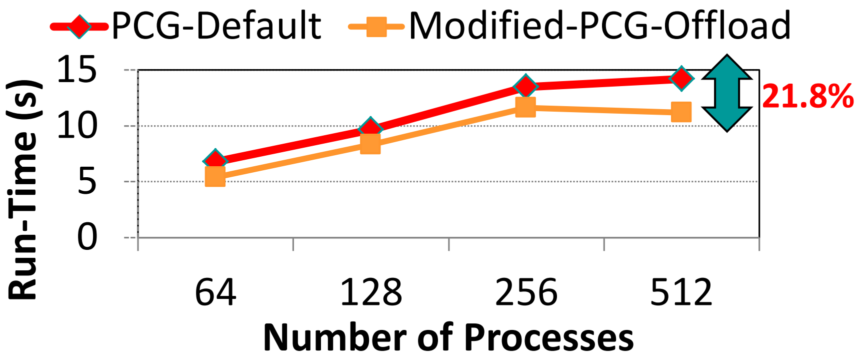
Application benefits with Non-Blocking Collectives based on CX-3 Collective Offload



Modified P3DFFT with Offload-Alltoall does up to 17% better than default version (128 Processes)



Modified HPL with Offload-Bcast does up to 4.5% better than default version (512 Processes)



Modified Pre-Conjugate Gradient Solver with Offload-Allreduce does up to 21.8% better than default version

K. Kandalla, et. al.. High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A Study with Parallel 3D FFT. ISC 2011

K. Kandalla, et. al, Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, HotI 2011

K. Kandalla, et. al., Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers, IPDPS '12

Can Network-Offload based Non-Blocking Neighborhood MPI Collectives Improve Communication Overheads of Irregular Graph Algorithms? K. Kandalla, A. Buluc, H. Subramoni, K. Tomko, J. Vienne, L. Oliker, and D. K. Panda, IWPAPS' 12

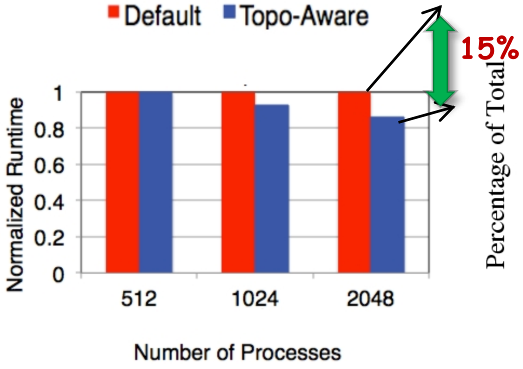
Network-Topology-Aware Placement of Processes

Can we design a highly scalable network topology detection service for IB?

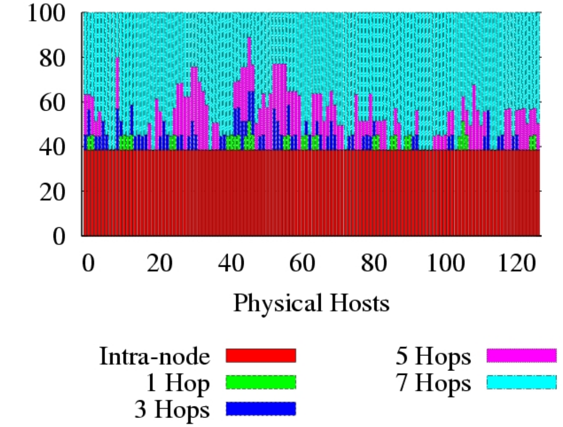
How do we design the MPI communication library in a network-topology-aware manner to efficiently leverage the topology information generated by our service?

What are the potential benefits of using a network-topology-aware MPI library on the performance of parallel scientific applications?

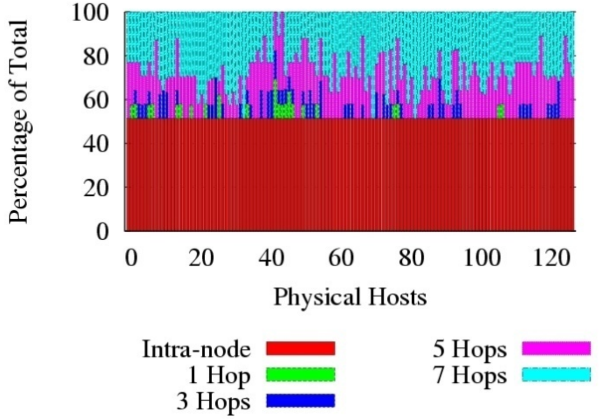
Overall performance and Split up of physical communication for MILC on Ranger



Performance for varying system sizes



Default for 2048 core run



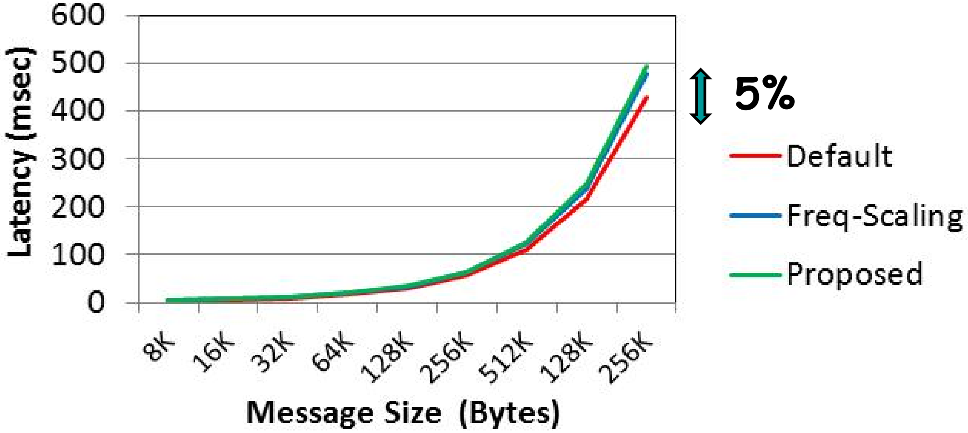
Topo-Aware for 2048 core run

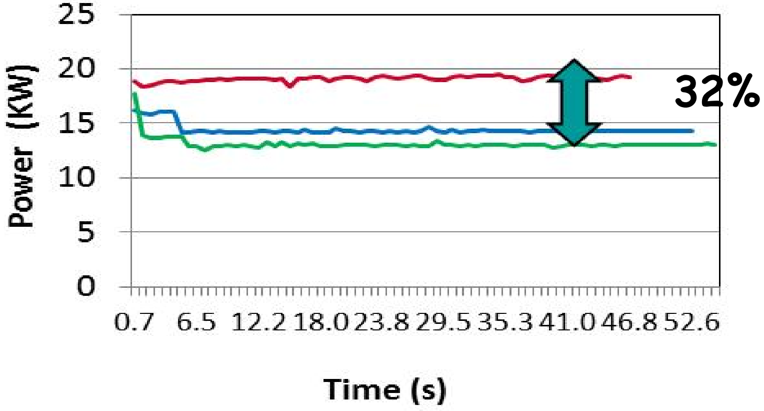
- Reduce network topology discovery time from $O(N^2_{hosts})$ to $O(N_{hosts})$
- 15% improvement in MILC execution time @ 2048 cores
- 15% improvement in Hypre execution time @ 1024 cores

H. Subramoni, S. Potluri, K. Kandalla, B. Barth, J. Vienne, J. Keasler, K. Tomko, K. Schulz, A. Moody, and D. K. Panda, Design of a Scalable InfiniBand Topology Service to Enable Network-Topology-Aware Placement of Processes, SC'12 . BEST Paper and BEST STUDENT Paper Finalist

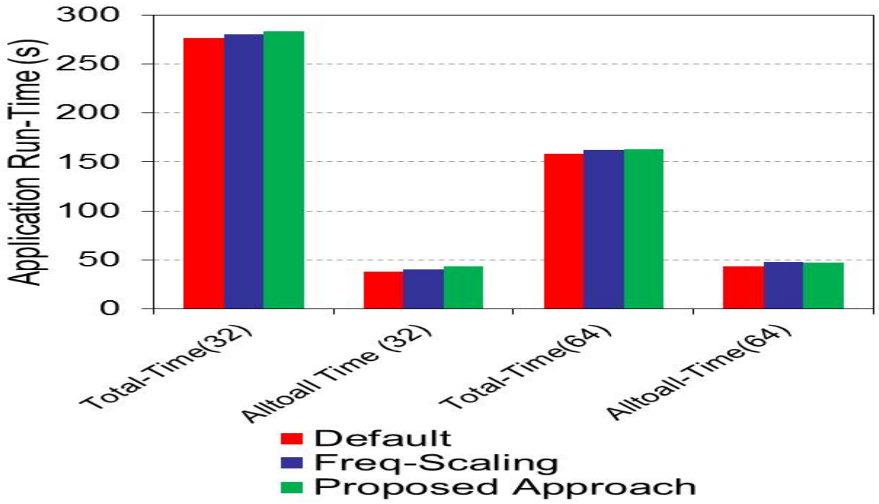
Power and Energy Savings with Power-Aware Collectives

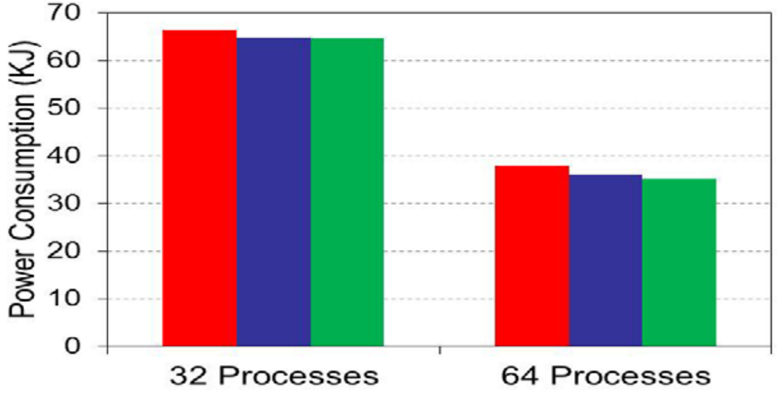
Performance and Power Comparison : MPI_Alltoall with 64 processes on 8 nodes





CPMD Application Performance and Power Savings





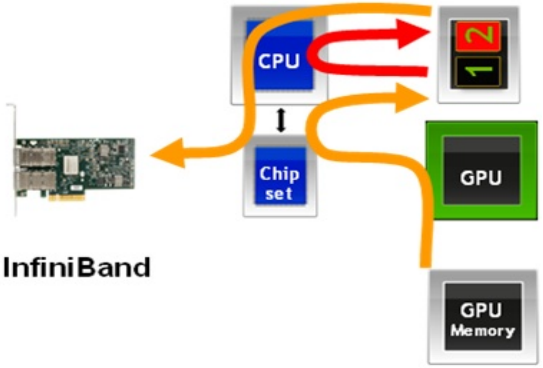
K. Kandalla, E. P. Mancini, S. Sur and D. K. Panda, “Designing Power Aware Collective Communication Algorithms for Infiniband Clusters”, ICPP ‘10

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

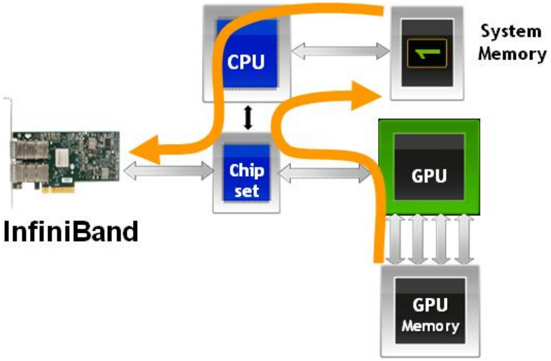
- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Fault-tolerance
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime

MVAPICH2-GPU: CUDA-Aware MPI

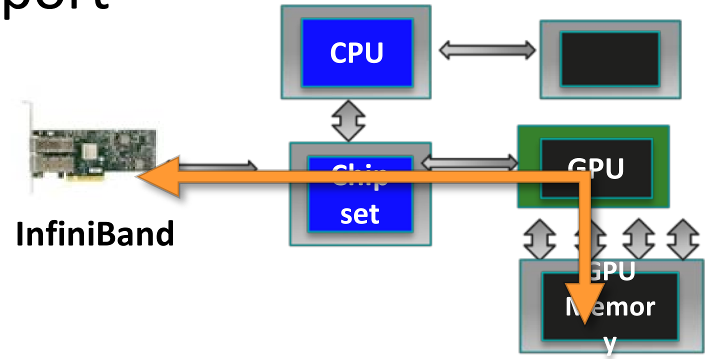
- Before CUDA 4: Additional copies
  - Low performance and low productivity



- After CUDA 4: Host-based pipeline
  - Unified Virtual Address
  - Pipeline CUDA copies with IB transfers
  - High performance and high productivity



- After CUDA 5.5: GPUDirect-RDMA support
  - GPU to GPU direct transfer
  - Bypass the host memory
  - Hybrid design to avoid PCI bottlenecks



MVAPICH2 1.8, 1.9, 2.0a-2.0rc1 Releases

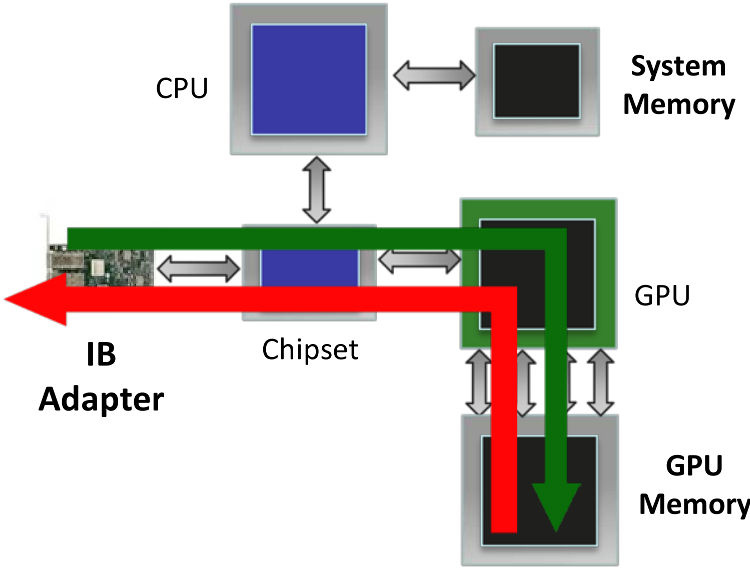
- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage 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

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

SNB E5-2670 /

IVB E5-2680V2



SNB E5-2670

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

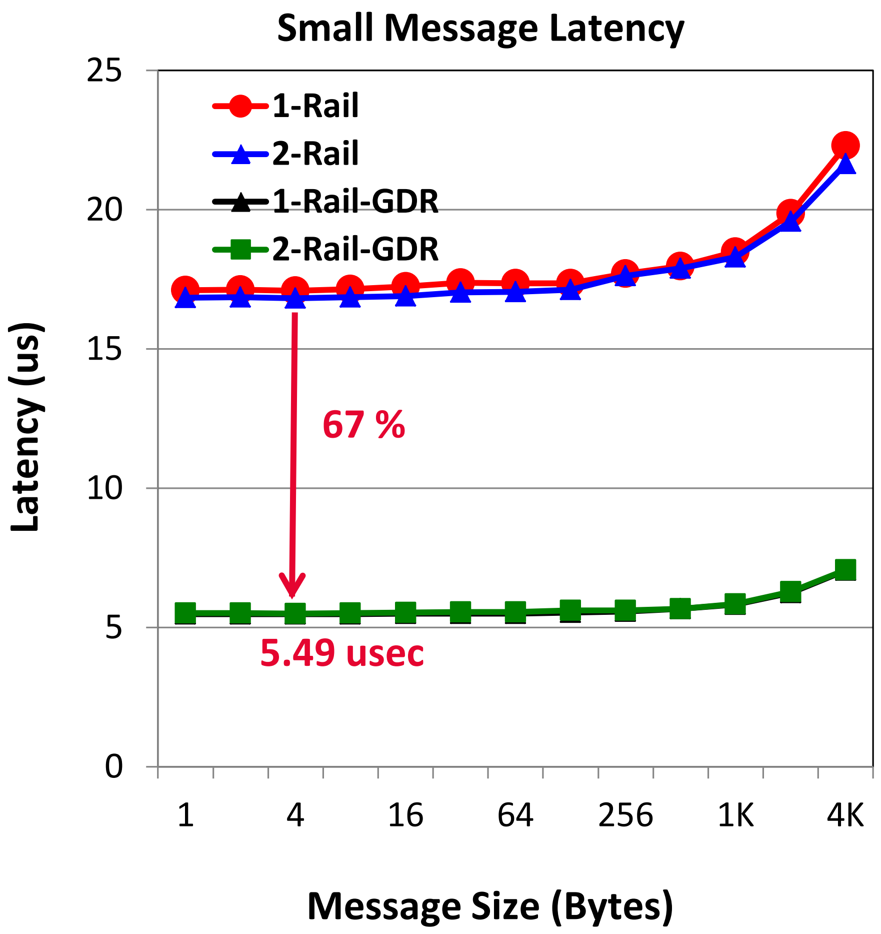
IVB E5-2680V2

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

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)

Performance of MVAPICH2 with GPUDirect-RDMA: Latency

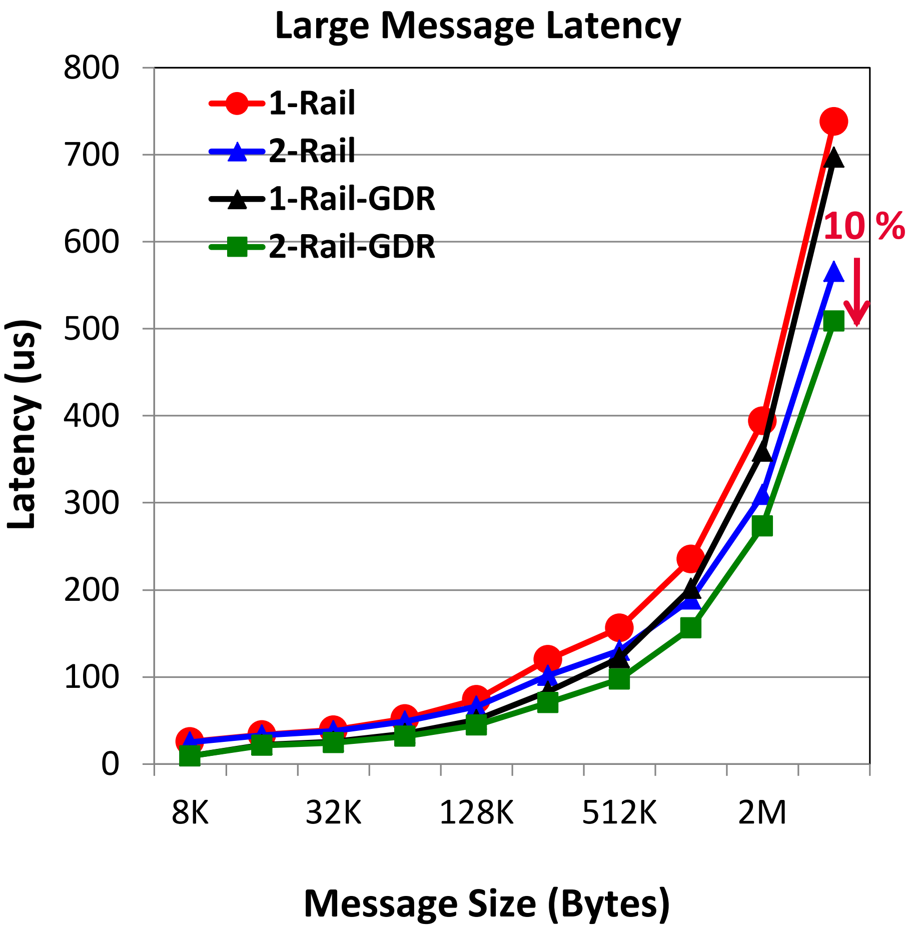
GPU-GPU Internode MPI Latency



Small Message Latency

67 %

5.49 usec



Large Message Latency

10 %

Based on MVAPICH2-2.0b

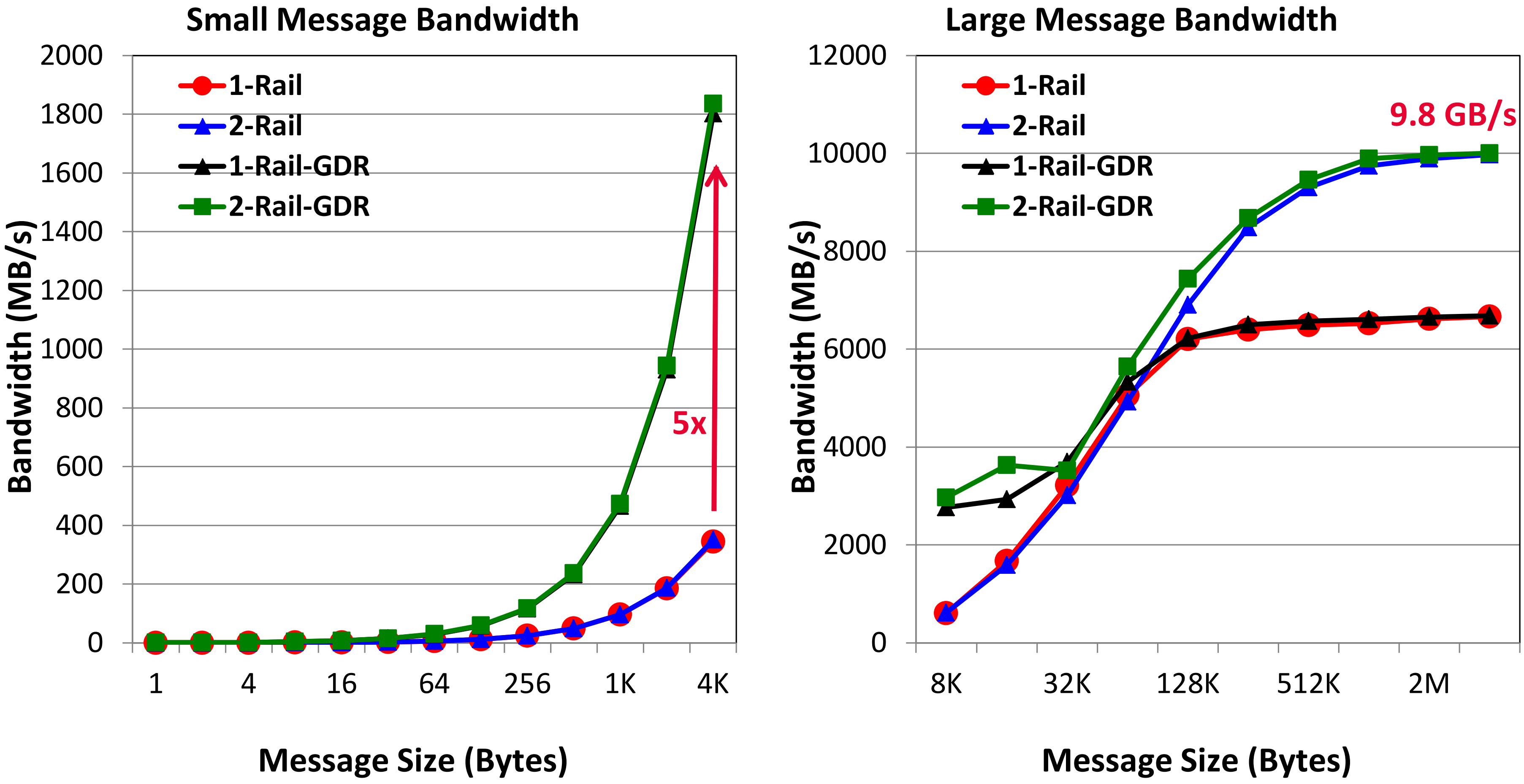
Intel Ivy Bridge (E5-2680 v2) node with 20 cores

NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA

CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Plug-in

Performance of MVAPICH2 with GPUDirect-RDMA: Bandwidth

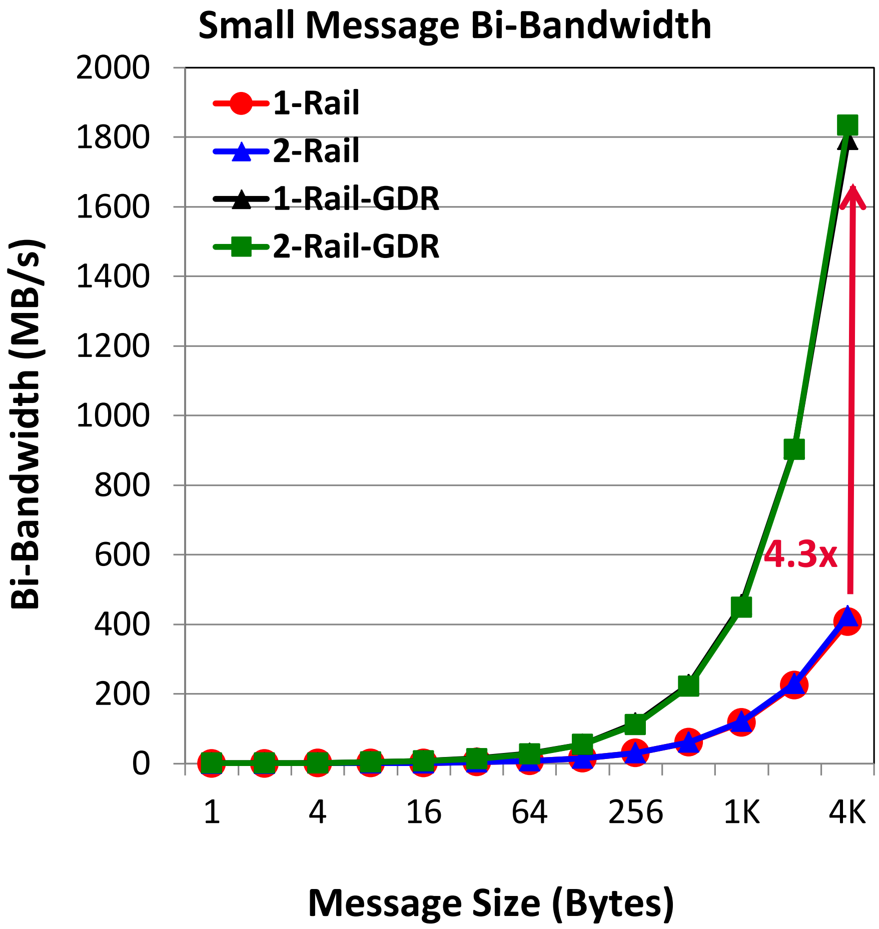
GPU-GPU Internode MPI Uni-Directional Bandwidth

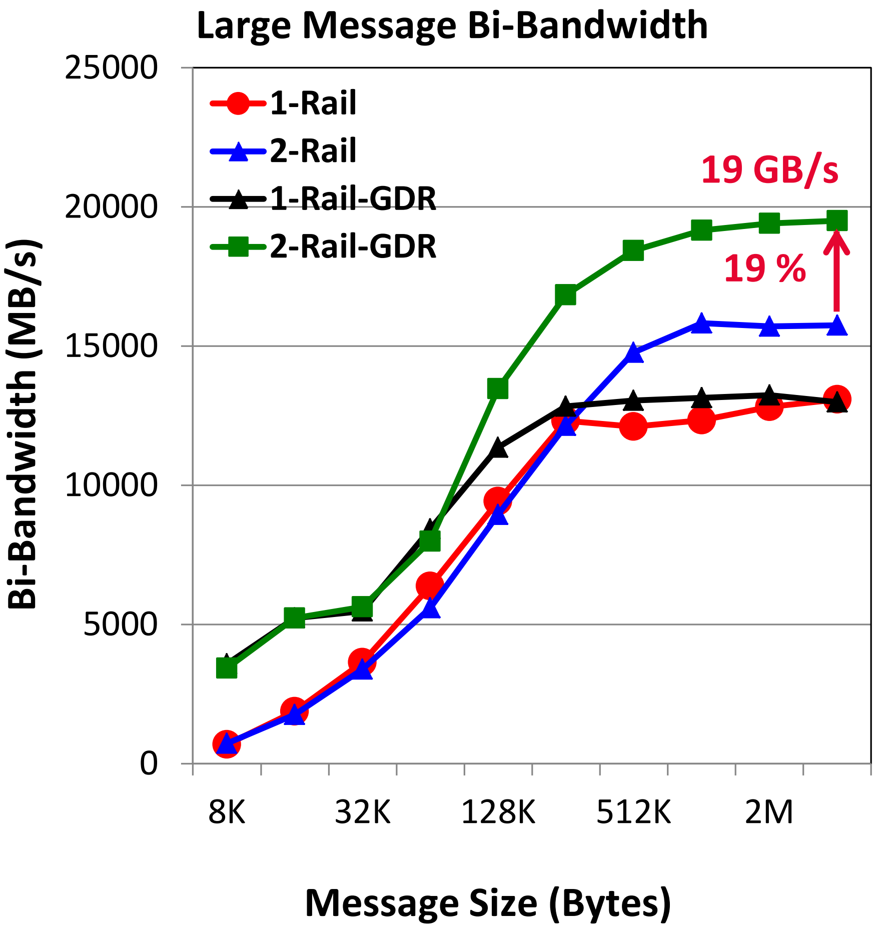


Based on MVAPICH2-2.0b
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Plug-in

Performance of MVAPICH2 with GPUDirect-RDMA: Bi-Bandwidth

GPU-GPU Internode MPI Bi-directional Bandwidth





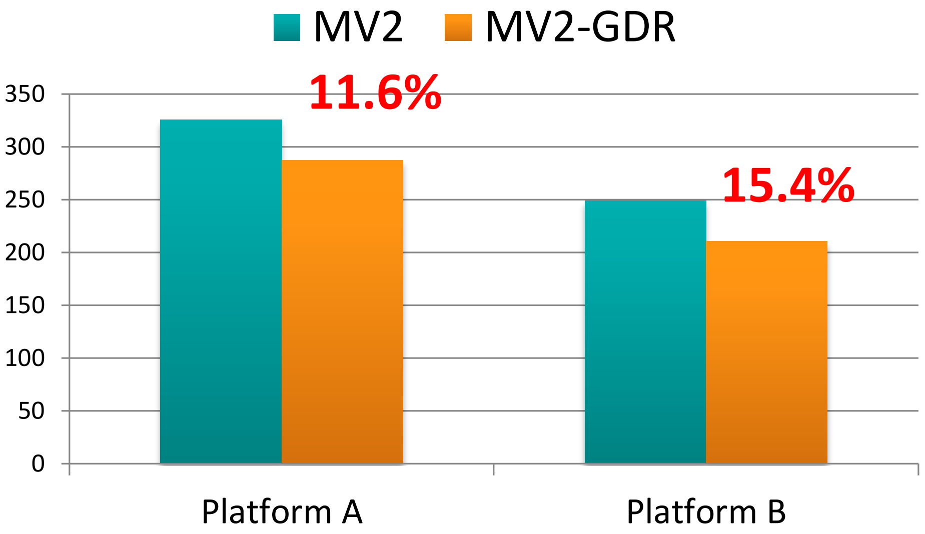
Based on MVAPICH2-2.0b

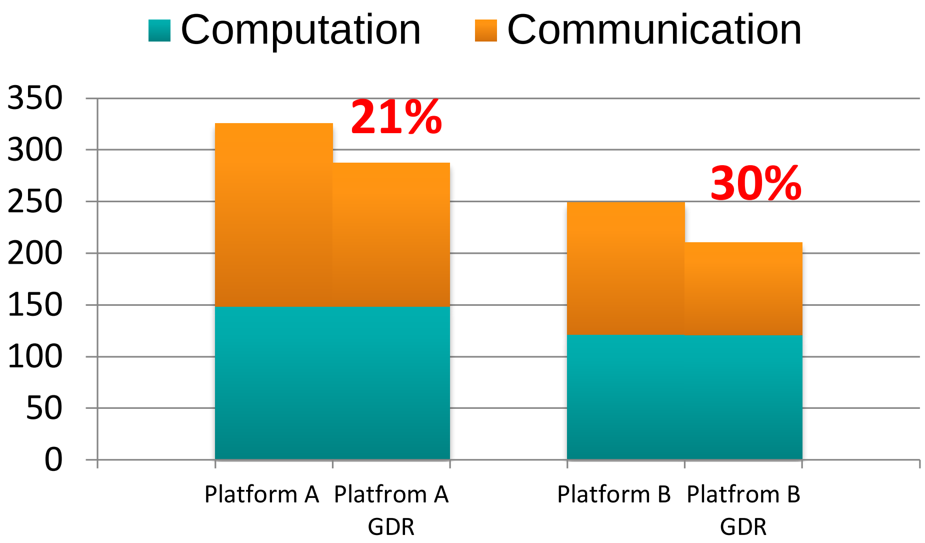
Intel Ivy Bridge (E5-2680 v2) node with 20 cores

NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA

CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Plug-in

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





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

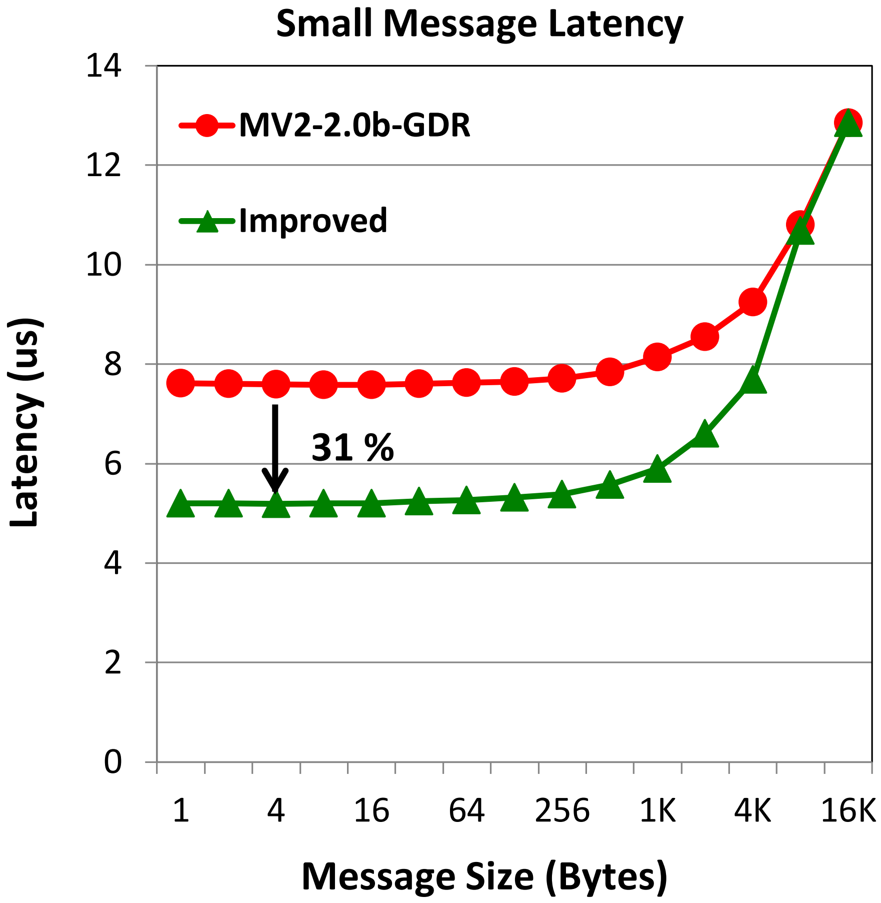
Platform B: Intel Ivy Bridge + NVIDIA Tesla K40 + Mellanox Connect-IB

- 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

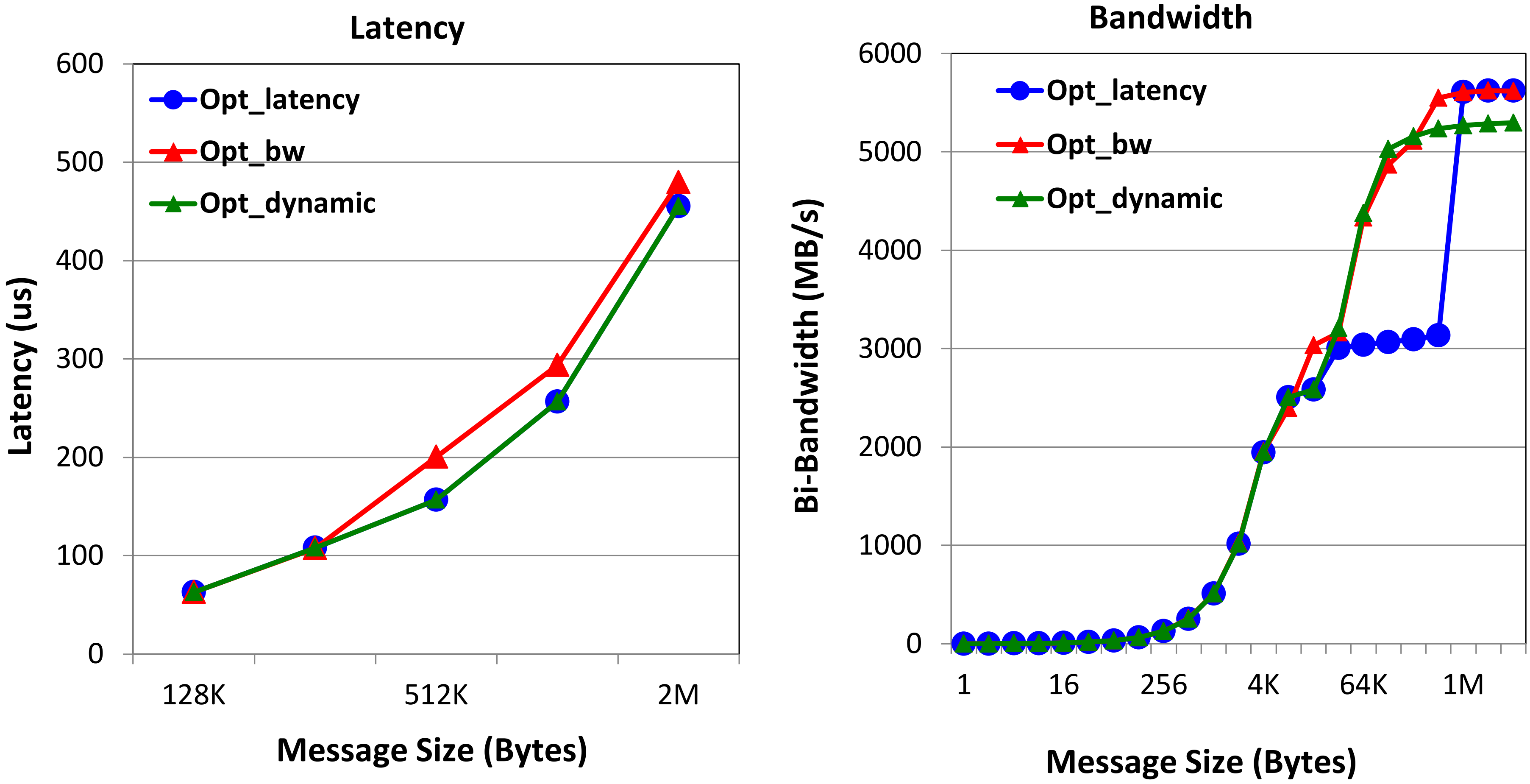


- Reduced synchronization and while avoiding expensive copies

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

Dynamic Tuning for Point-to-point Performance

GPU-GPU Internode MPI Performance



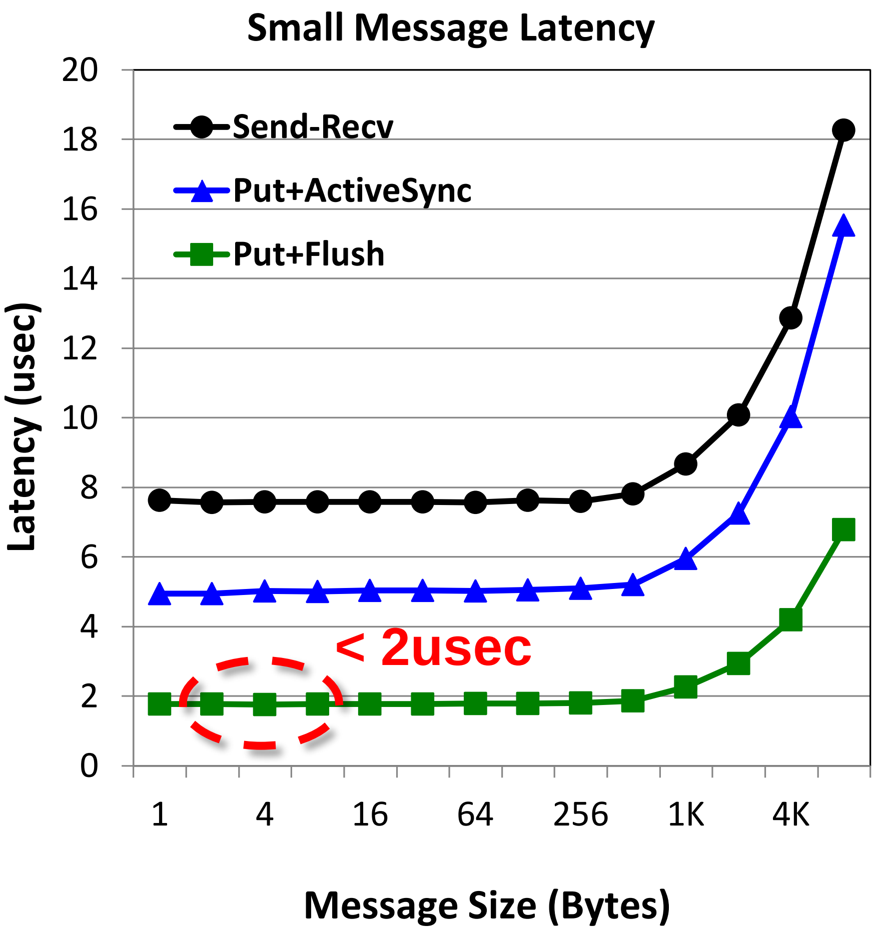
Latency

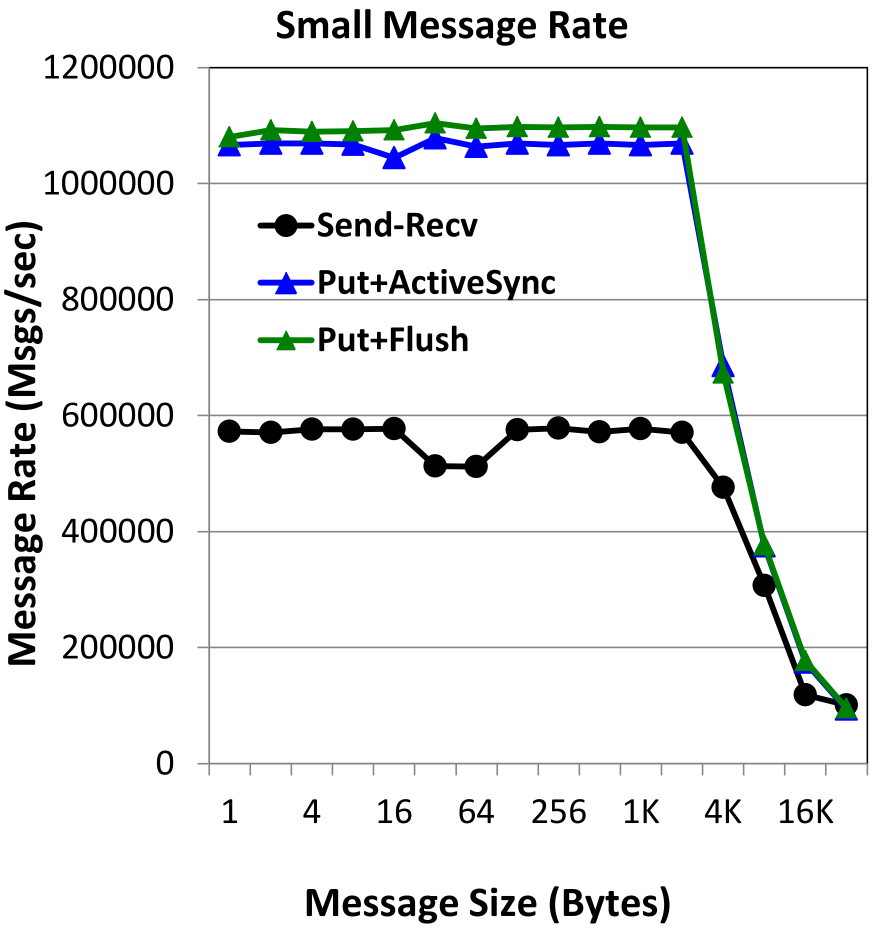
Bandwidth

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

MPI-3 RMA Support with GPUDirect RDMA

MPI-3 RMA provides flexible synchronization and completion primitives





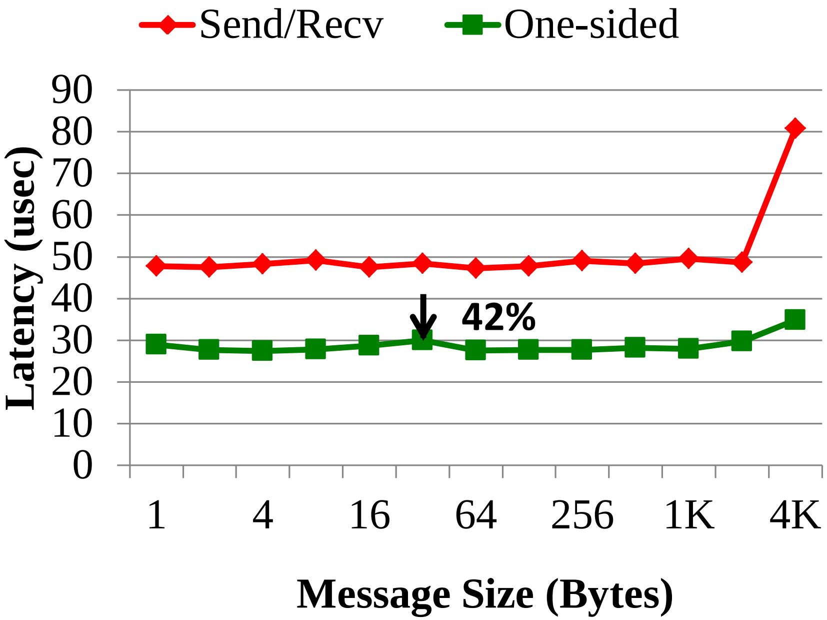
Based on MVAPICH2-2.0b + enhancements

Intel Ivy Bridge (E5-2630 v2) node with 12 cores

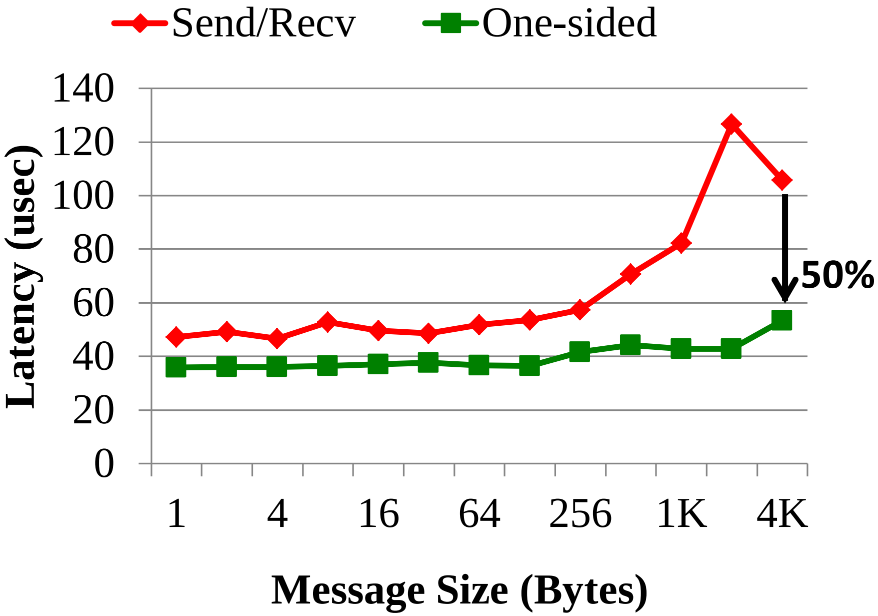
NVIDIA Tesla K40c GPU, Mellanox Connect-IB FDR HCA

CUDA 5.5, Mellanox OFED 2.0 with GPUDirect-RDMA Plug-in

Communication Kernel Evaluation: 3Dstencil and Alltoall



3D Stencil with 16 GPU nodes



AlltoAll with 16 GPU nodes

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

Advanced 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

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)

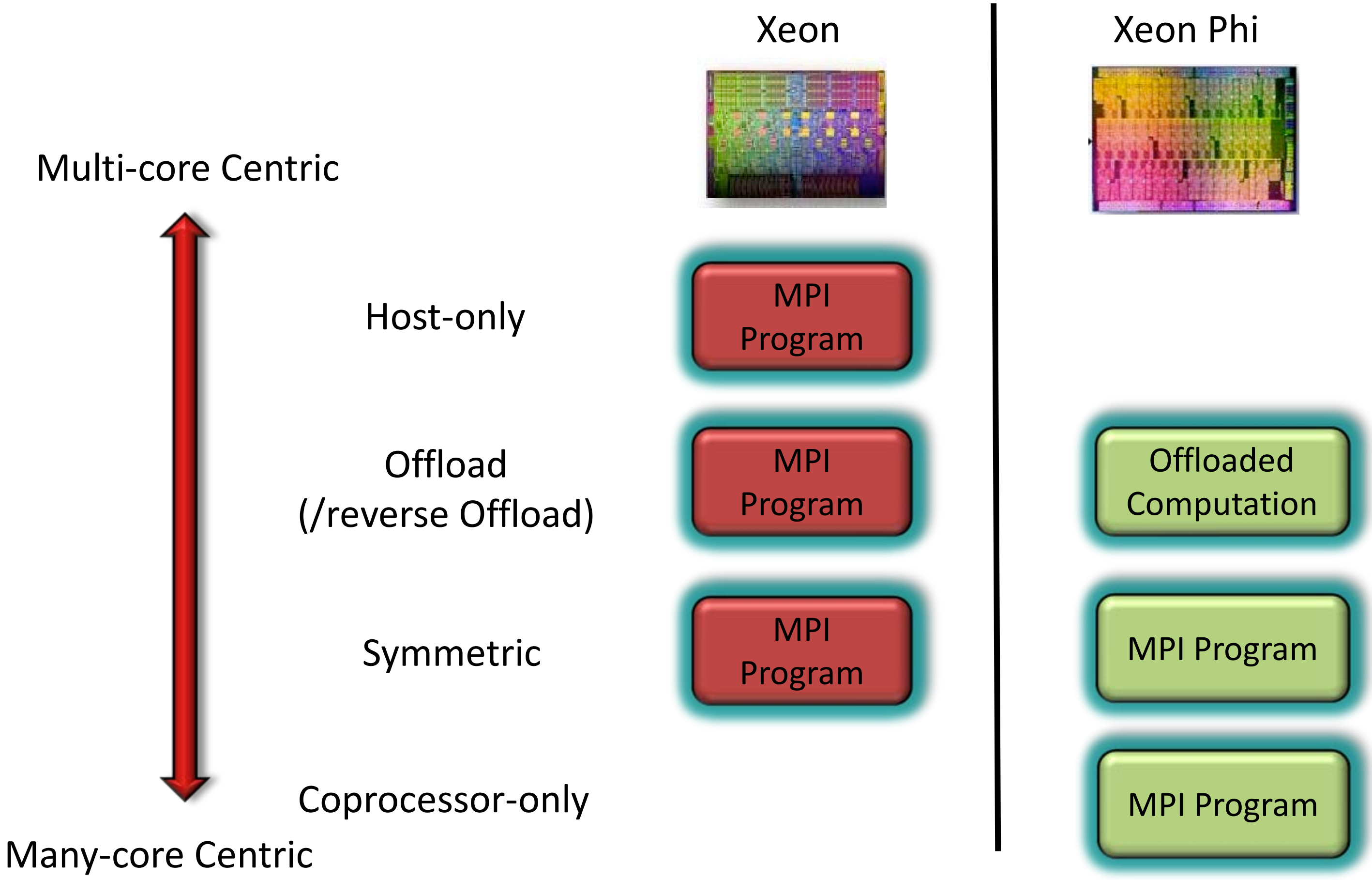
- Has optimized designs for point-to-point communication using GDR
- Work under progress for optimizing collective and one-sided communication
- Contact MVAPICH help list with any questions related to the package mvapich-help@cse.ohio-state.edu
- MVAPICH2-GDR-RC1 with additional optimizations coming soon!!

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Fault-tolerance
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime

MPI Applications on MIC Clusters

- Flexibility in launching MPI jobs on clusters with Xeon Phi



Xeon

Xeon Phi

Multi-core Centric

Host-only

MPI Program

Offload (/reverse Offload)

MPI Program

Offloaded Computation

Symmetric

MPI Program

MPI Program

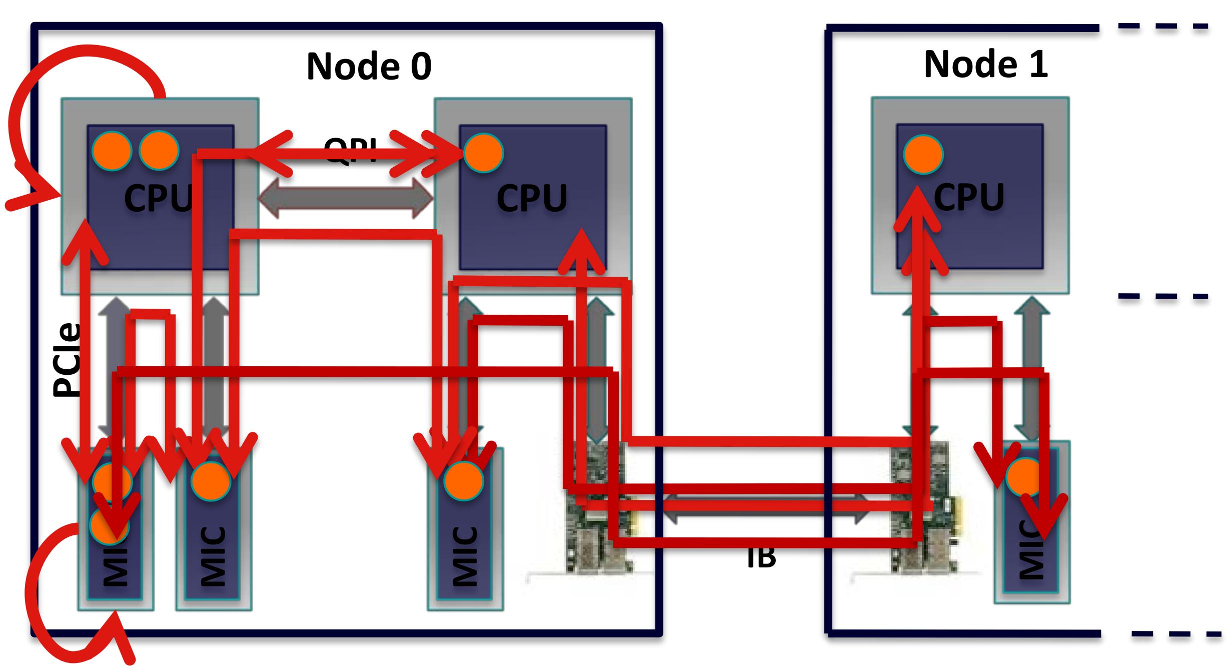
Coprocessor-only

MPI Program

Many-core Centric

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 . . .

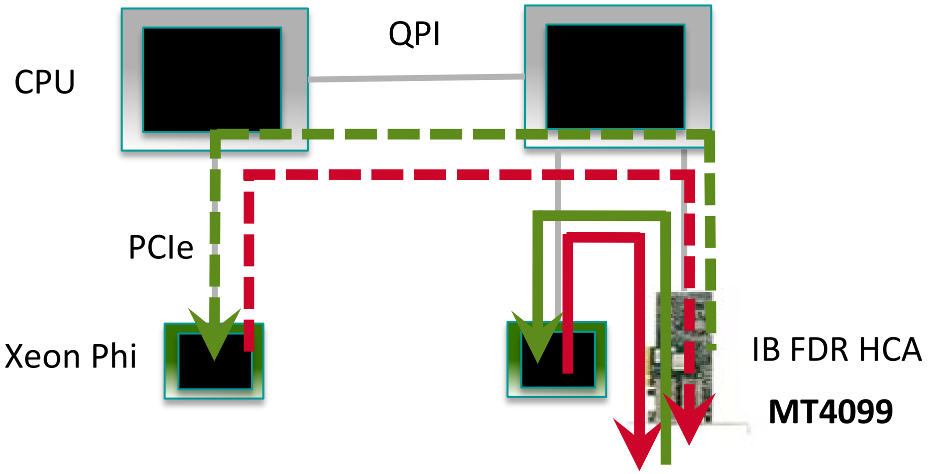
- Critical for runtimes to optimize data movement, hiding the complexity

MVAPICH2-MIC Design for Clusters with IB and MIC

- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only
  - Symmetric Mode
- Multi-MIC Node Configurations

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



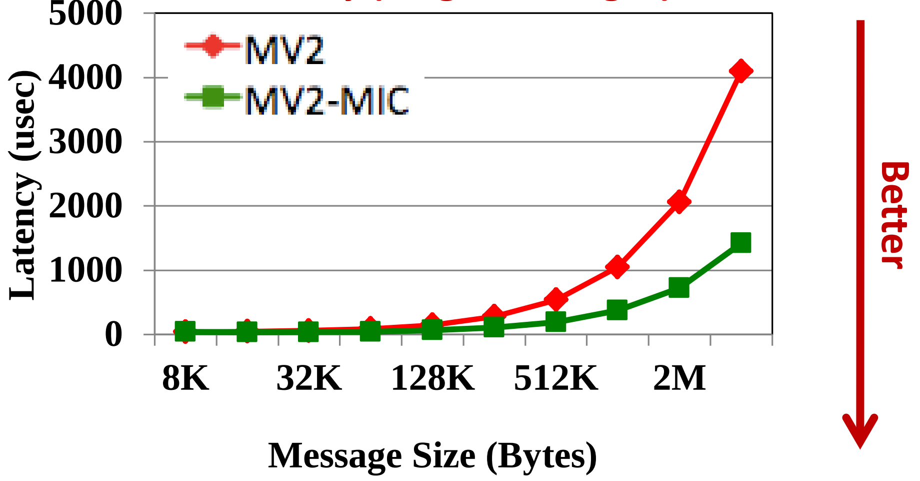
Peak IB FDR Bandwidth: 6397 MB/s

			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%)

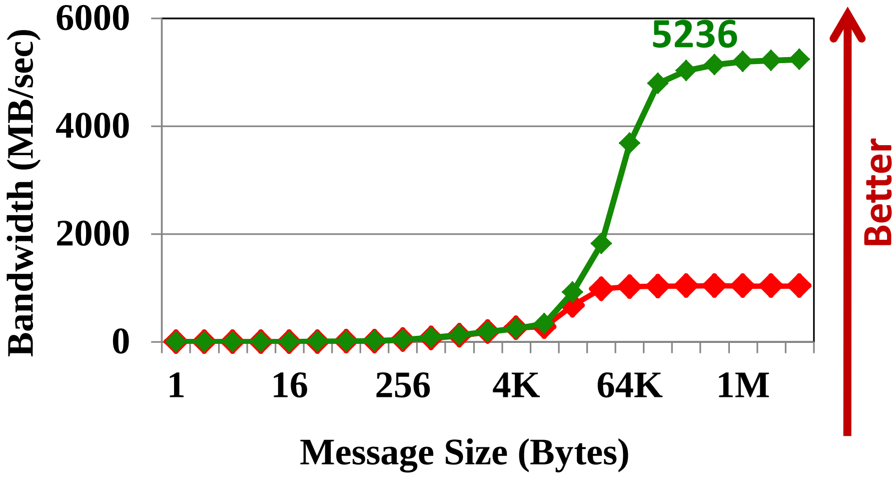
MIC-Remote-MIC P2P Communication

Intra-socket P2P

Latency (Large Messages)

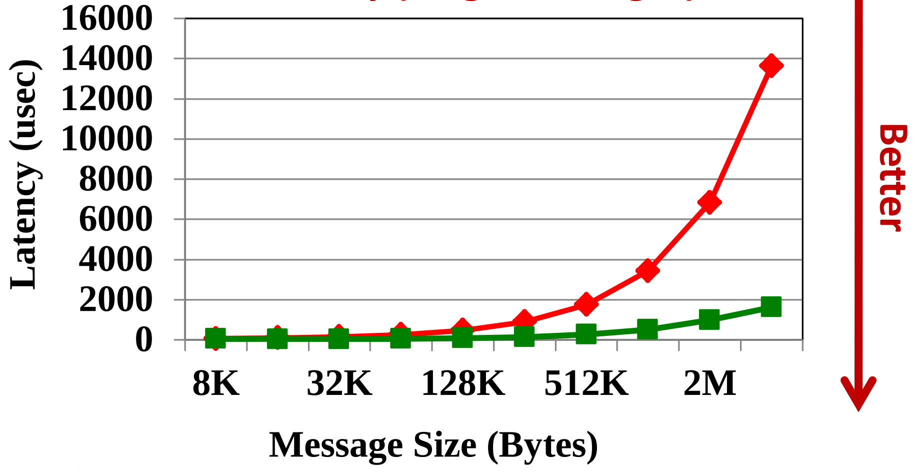


Bandwidth

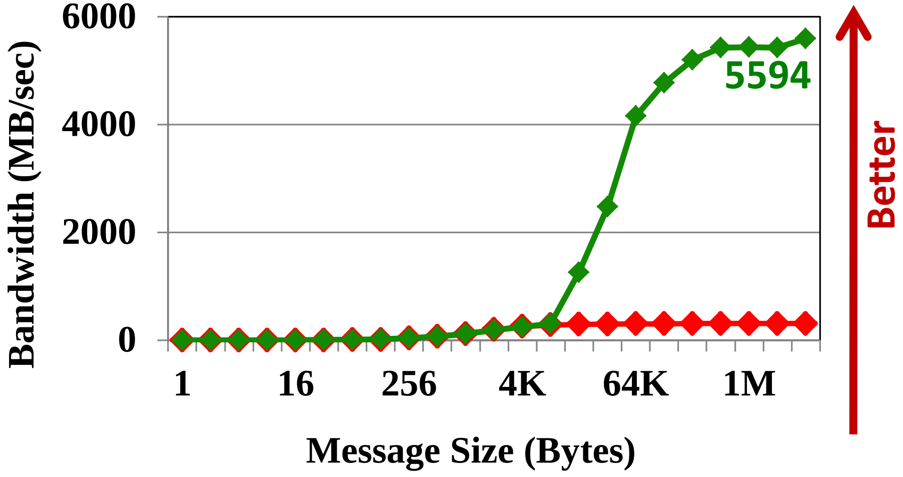


Inter-socket P2P

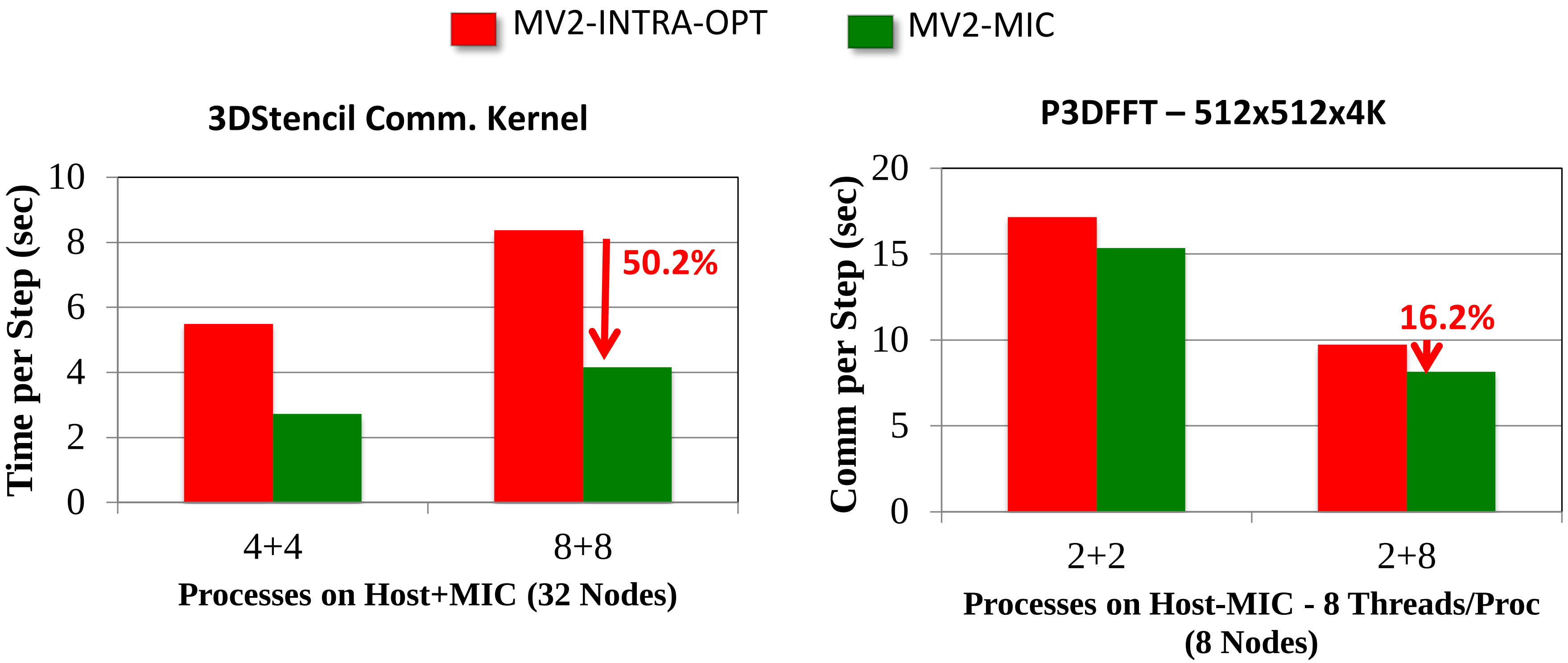
Latency (Large Messages)



Bandwidth

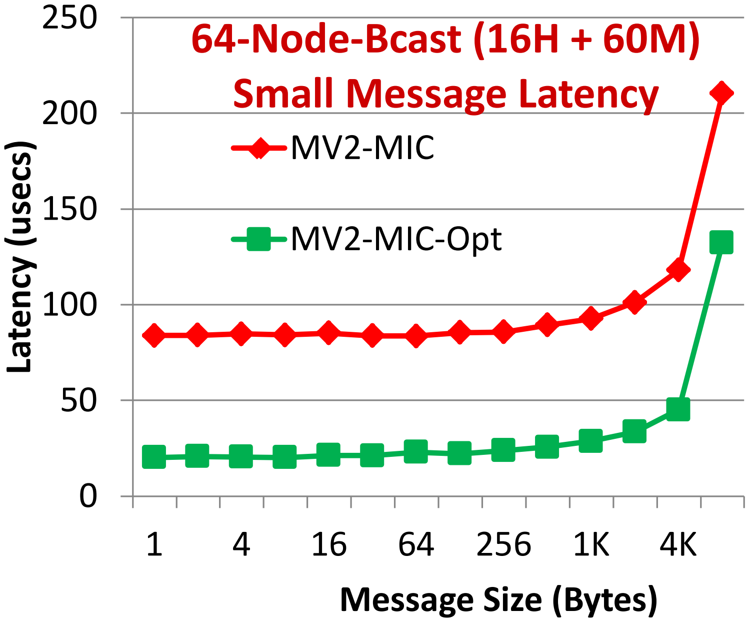


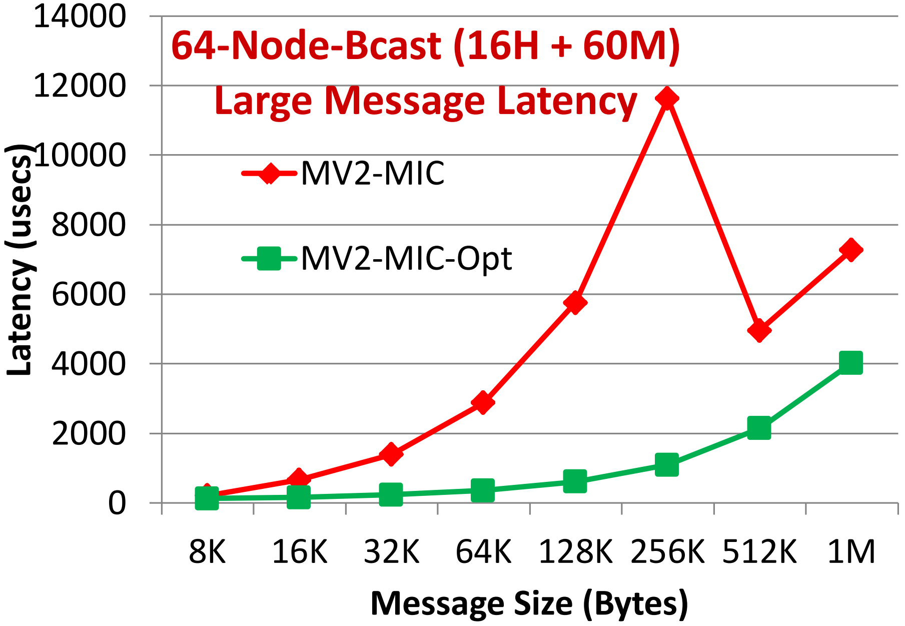
InterNode – Symmetric Mode - P3DFFT Performance

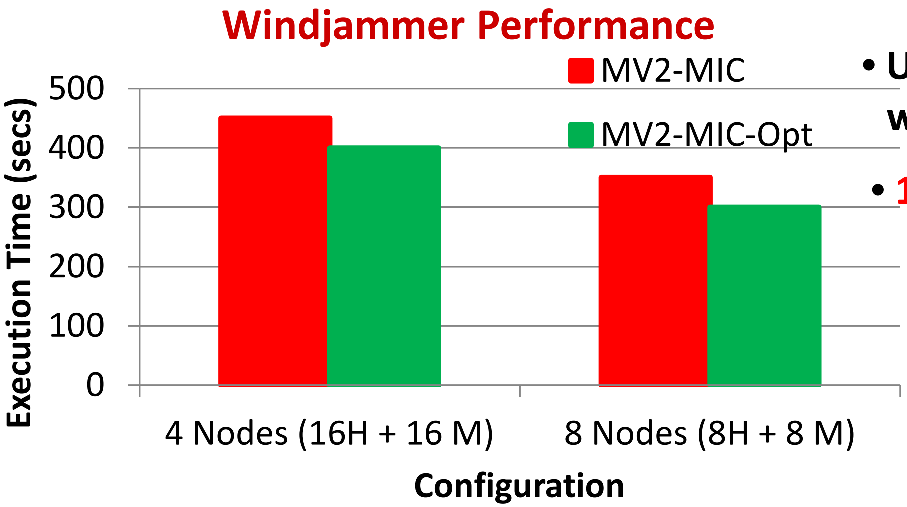


- To explore different threading levels for host and Xeon Phi

Optimized MPI Collectives for MIC Clusters (Broadcast)



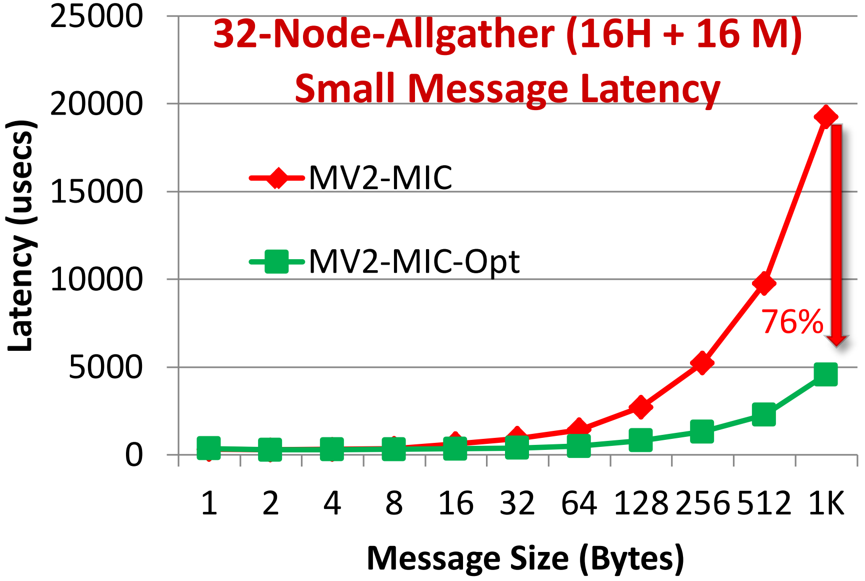




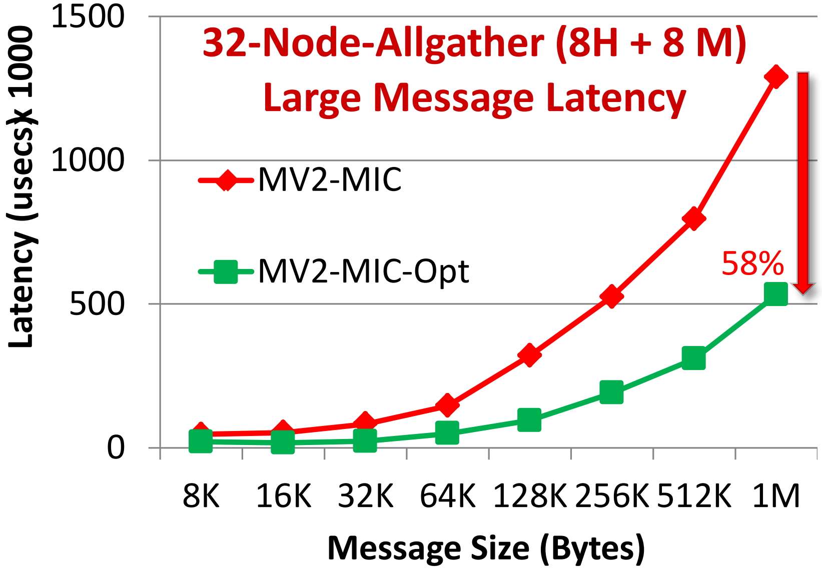
- Up to 50 - 80% improvement in MPI_Bcast latency with as many as 4864 MPI processes (Stampede)
- 12% and 16% improvement in the performance of Windjammer with 128 processes

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

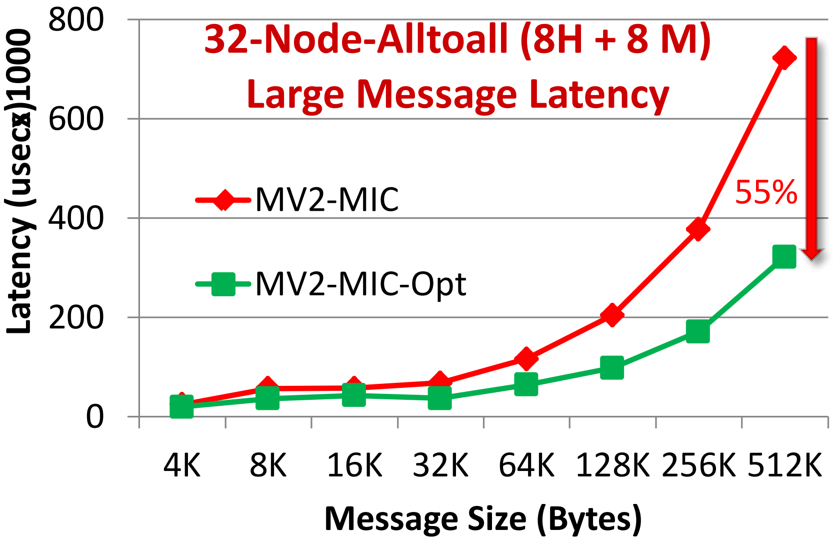
Optimized MPI Collectives for MIC Clusters (Allgather & Alltoall)



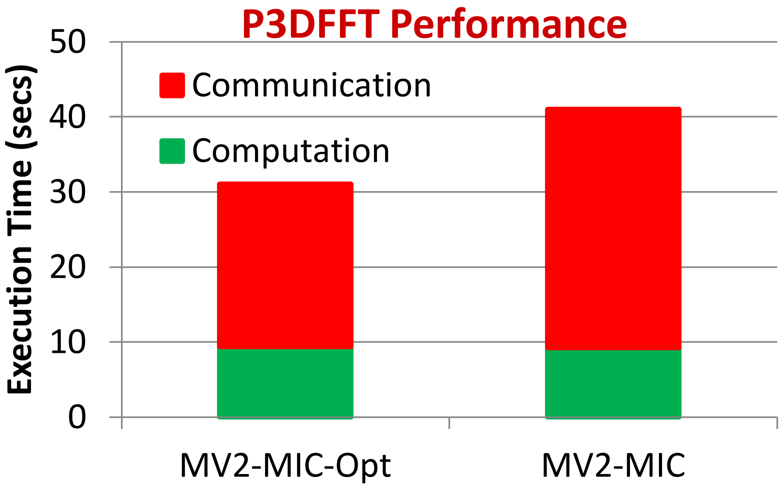
32-Node-Allgather (16H + 16 M) Small Message Latency



32-Node-Allgather (8H + 8 M) Large Message Latency



32-Node-Alltoall (8H + 8 M) Large Message Latency



P3DFFT Performance

32 Nodes (8H + 8M), Size = 2K*2K*1K

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)

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

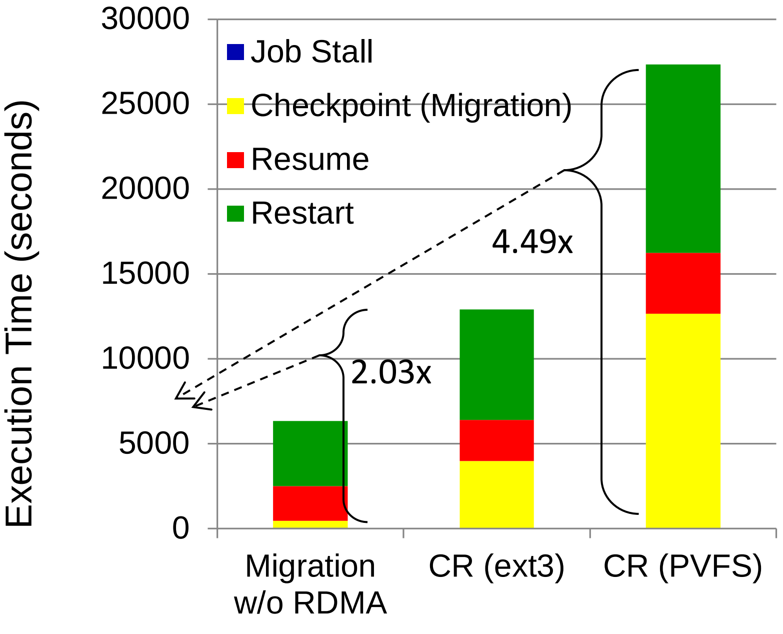
- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Fault-tolerance
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime

Fault Tolerance

- Component failures are common in large-scale clusters
- Imposes need on reliability and fault tolerance
- Multiple challenges:
  - Checkpoint-Restart vs. Process Migration
  - Benefits of Scalable Checkpoint Restart (SCR) Support
    - Application guided
    - Application transparent

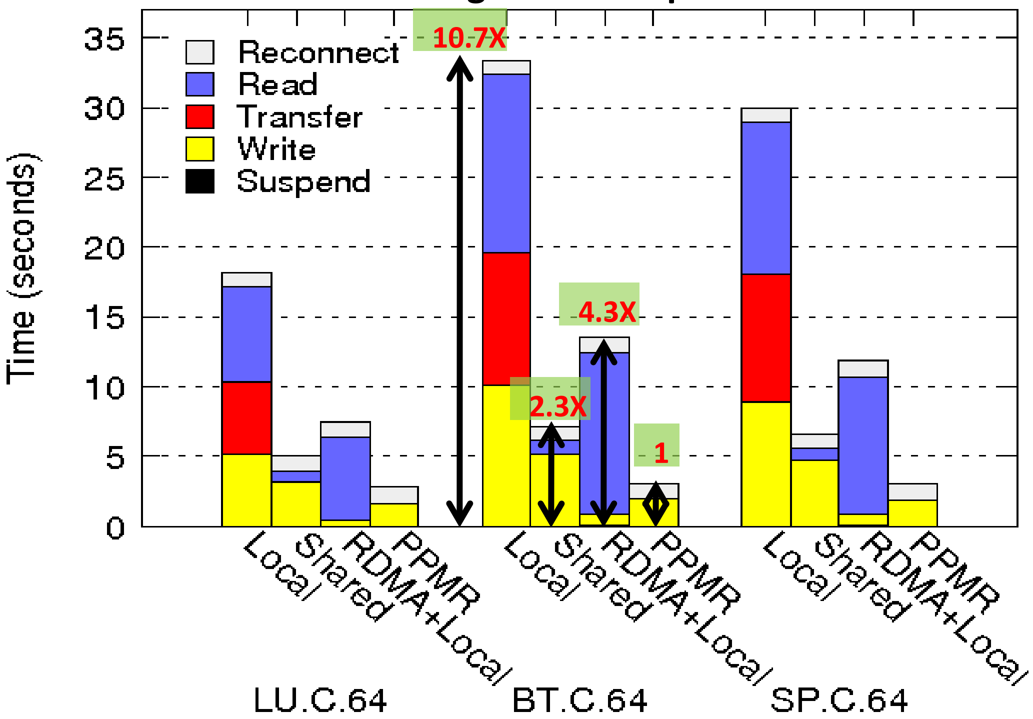
Checkpoint-Restart vs. Process Migration
Low Overhead Failure Prediction with IPMI

- Job-wide Checkpoint/Restart is not scalable
- Job-pause and Process Migration framework can deliver pro-active fault-tolerance
- Also allows for cluster-wide load-balancing by means of job compaction



LU Class C Benchmark (64 Processes)

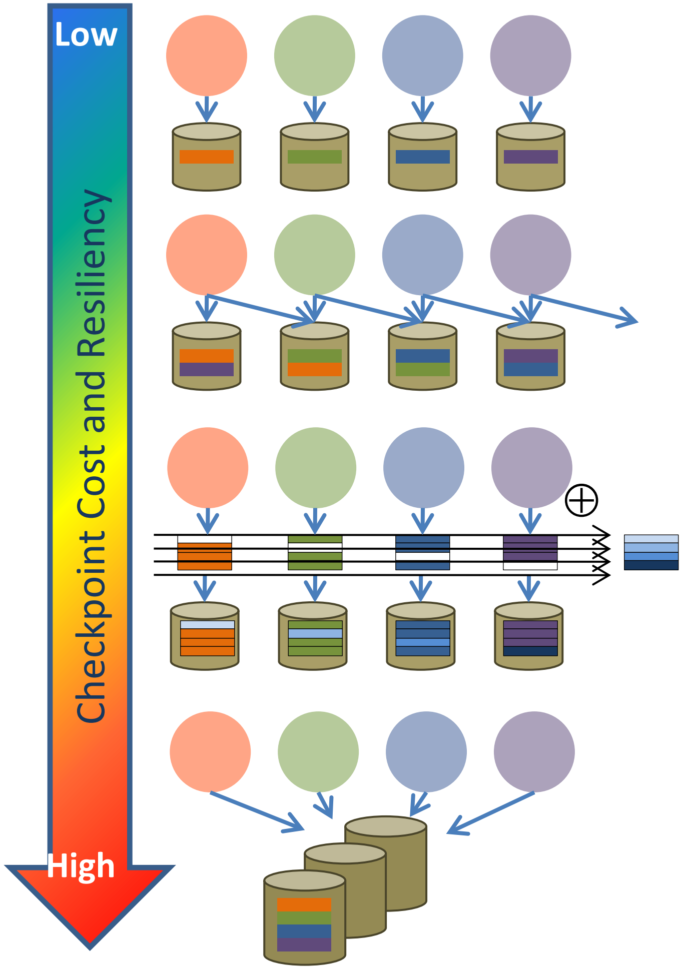
Time to migrate 8 MPI processes



X. Ouyang, R. Rajachandrasekar, X. Besseron, D. K. Panda, High Performance Pipelined Process Migration with RDMA, CCGrid 2011

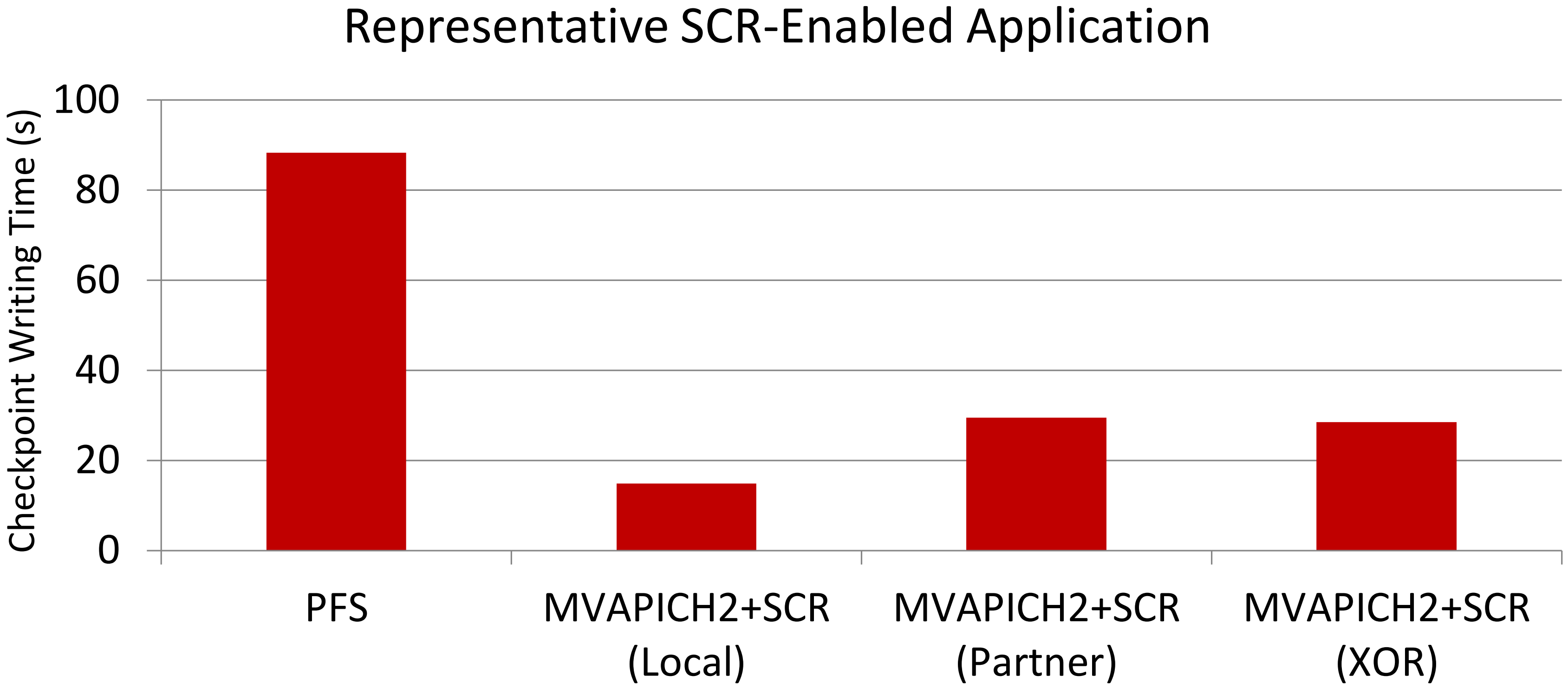
X. Ouyang, S. Marcarelli, R. Rajachandrasekar and D. K. Panda, RDMA-Based Job Migration Framework for MPI over InfiniBand, Cluster 2010

Multi-Level Checkpointing with ScalableCR (SCR)



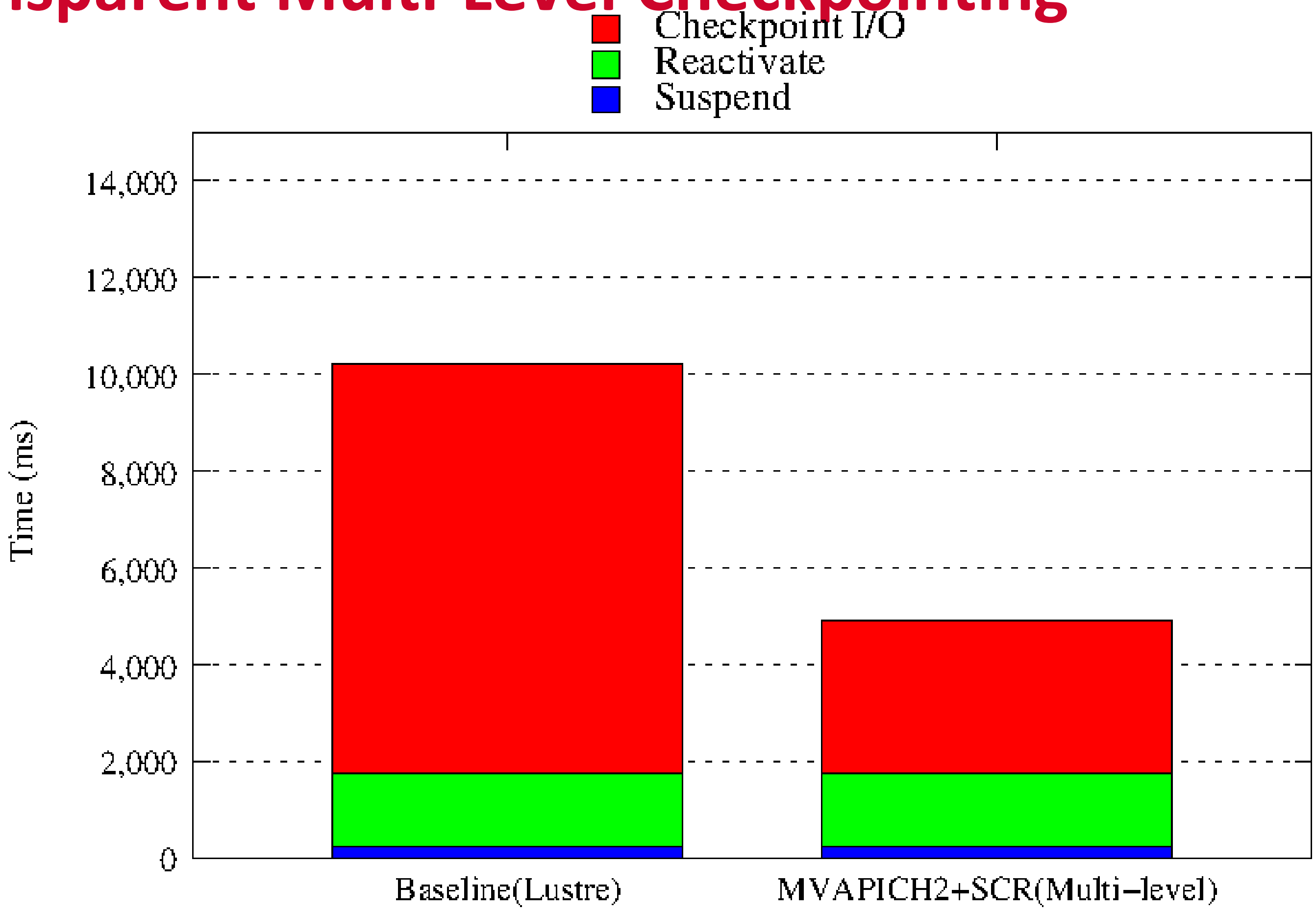
- LLNL's Scalable Checkpoint/Restart library
- Can be used for application guided and application transparent checkpointing
- Effective utilization of storage hierarchy
  - **Local:** Store checkpoint data on node's local storage, e.g. local disk, ramdisk
  - **Partner:** Write to local storage and on a partner node
  - **XOR:** Write file to local storage and small sets of nodes collectively compute and store parity redundancy data (RAID-5)
  - **Stable Storage:** Write to parallel file system

Application-guided Multi-Level Checkpointing



- Checkpoint writing phase times of representative SCR-enabled MPI application
- **512** MPI processes (8 procs/node)
- Approx. **51 GB** checkpoints

Transparent Multi-Level Checkpointing

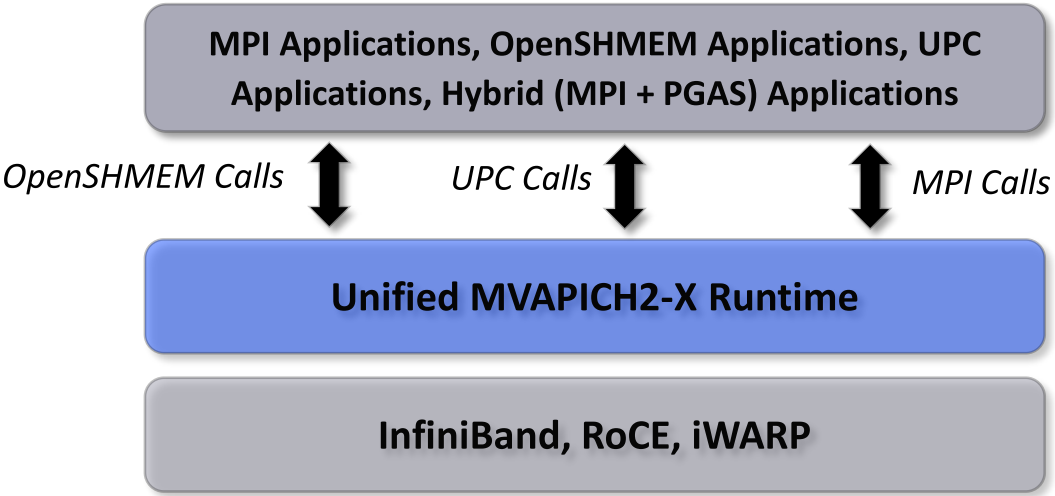


- **ENZO Cosmology application** – Radiation Transport workload
- Using MVAPICH2's CR protocol instead of the application's in-built CR mechanism
- **512** MPI processes running on the **Stampede Supercomputer@TACC**
- Approx. **12.8 GB** checkpoints: ~**2.5X reduction in checkpoint I/O costs**

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-X for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Collective communication
  - Multicore-aware and Hardware-multicast-based
  - Offload and Non-blocking
  - Topology-aware
  - Power-aware
- Support for GPGPUs
- Support for Intel MICs
- Fault-tolerance
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, …) with Unified Runtime

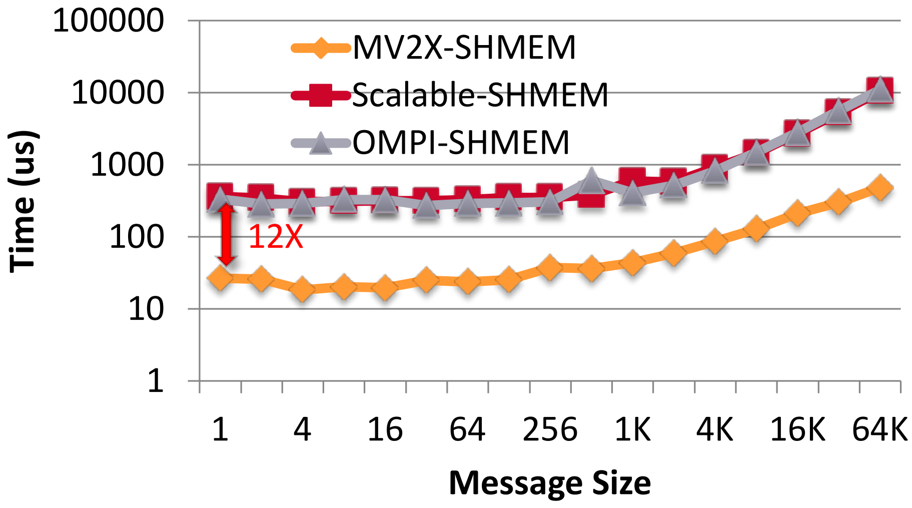
MVAPICH2-X for Hybrid MPI + PGAS Applications


MPI Applications, OpenSHMEM Applications, UPC Applications, Hybrid (MPI + PGAS) Applications
OpenSHMEM Calls
UPC Calls
MPI Calls
Unified MVAPICH2-X Runtime
InfiniBand, RoCE, iWARP

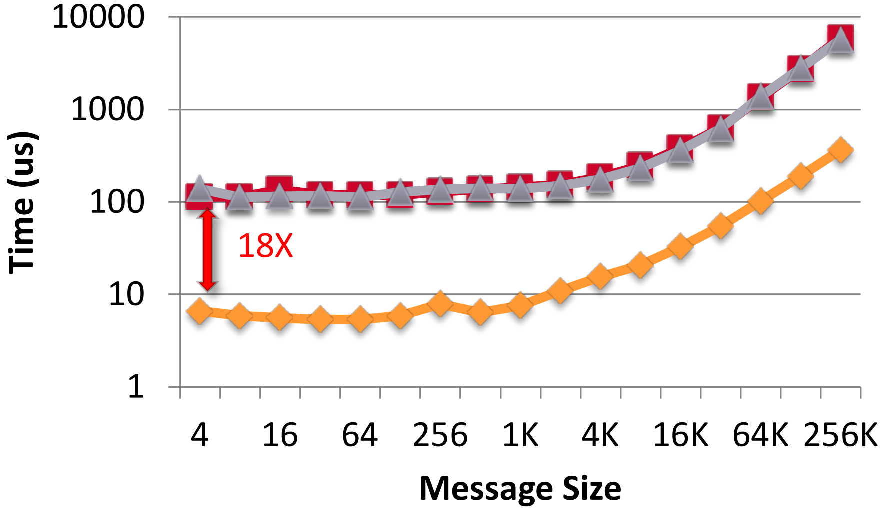
- Unified communication runtime for MPI, UPC, OpenSHMEM available with MVAPICH2-X 1.9 onwards!
  - http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant
  - Scalable Inter-node and Intra-node communication – point-to-point and collectives

OpenSHMEM Collective Communication Performance

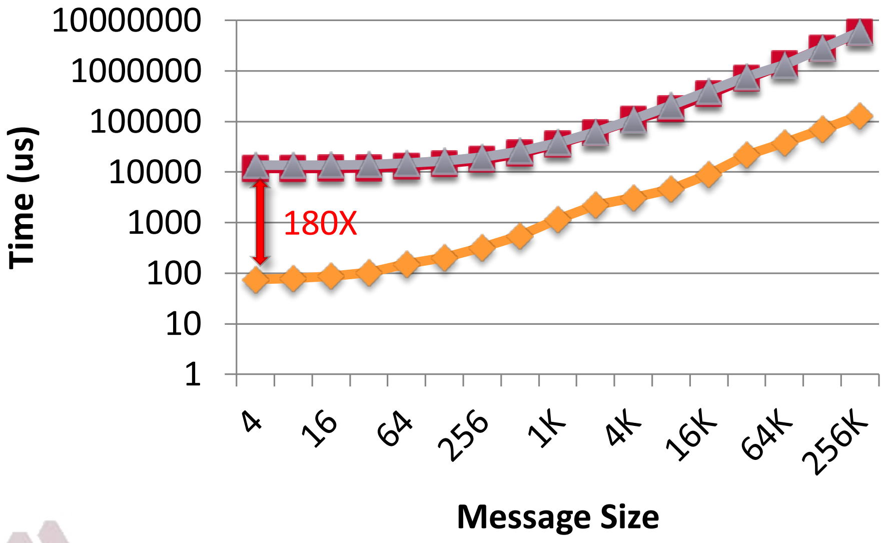
Reduce (1,024 processes)



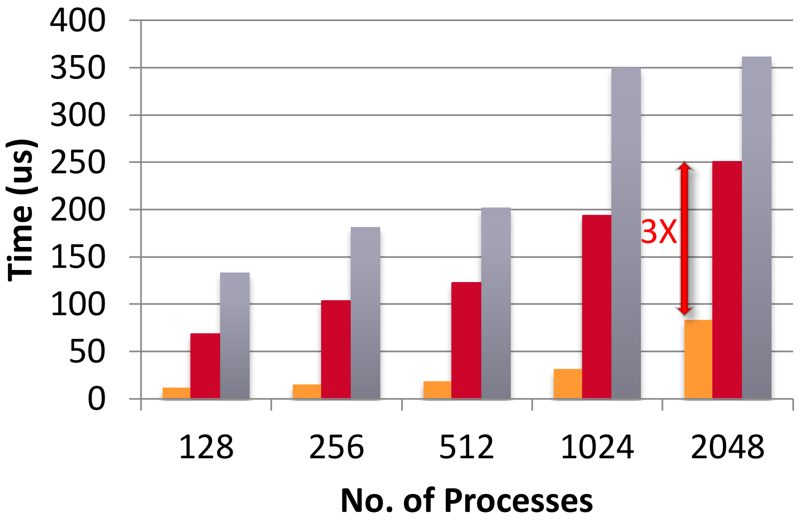
Broadcast (1,024 processes)



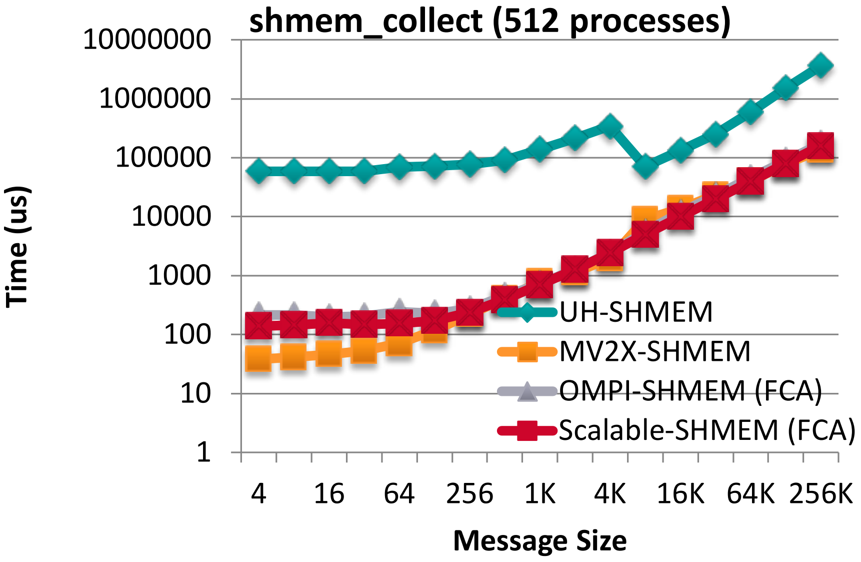
Collect (1,024 processes)

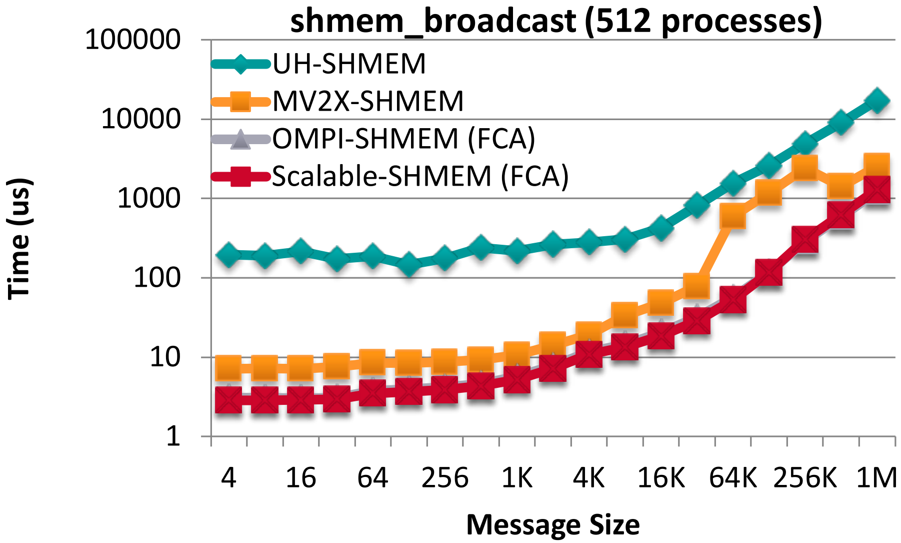


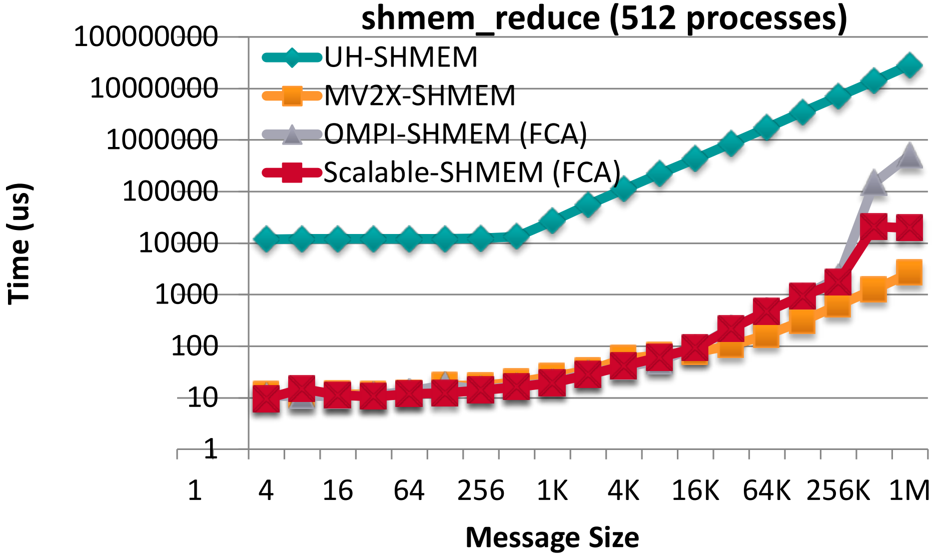
Barrier



OpenSHMEM Collectives: Comparison with FCA

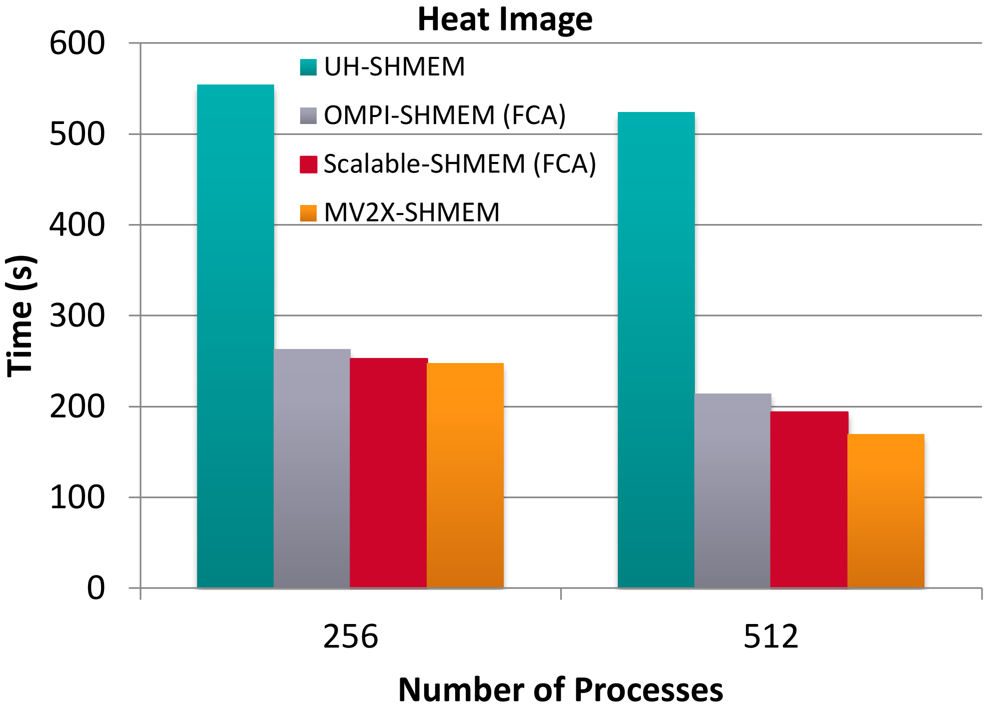

shmem_collect (512 processes)

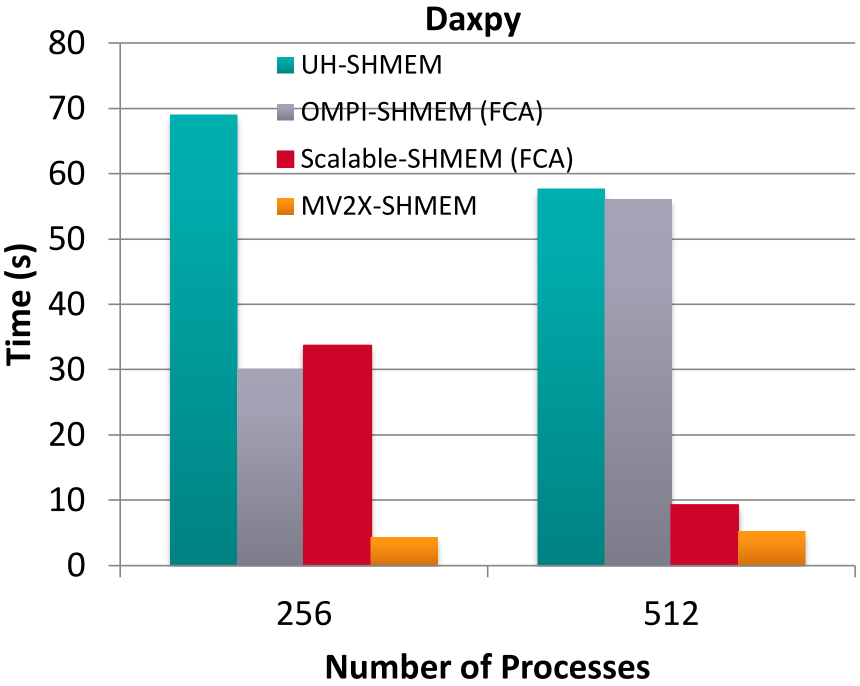

shmem_broadcast (512 processes)


shmem_reduce (512 processes)

- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- 8-byte reduce operation with 512 processes (us): MV2X-SHMEM – 10.9, OMPI-SHMEM – 10.79, Scalable-SHMEM – 11.97

OpenSHMEM Application Evaluation

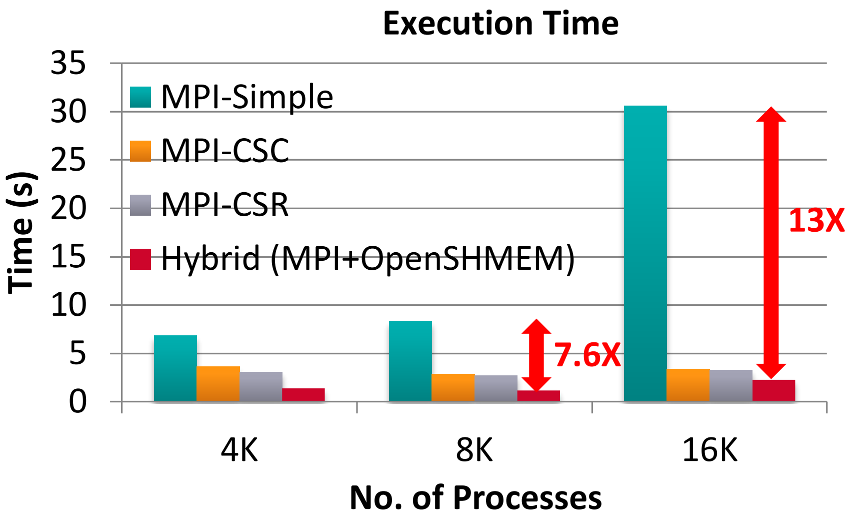


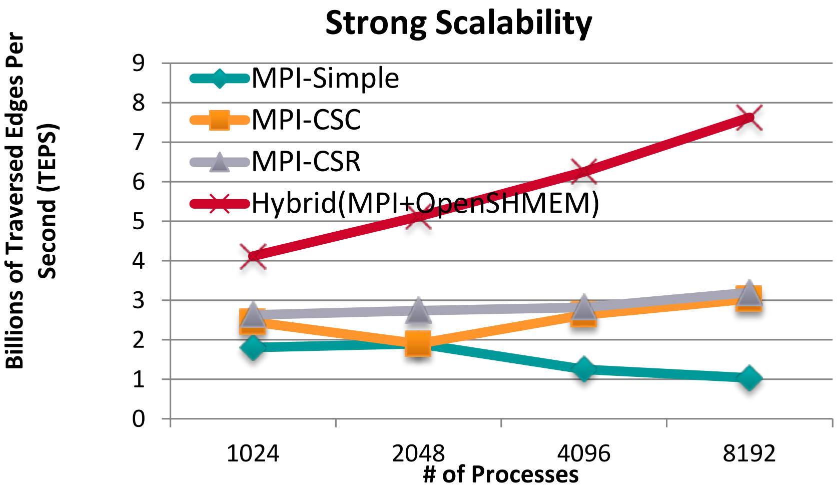


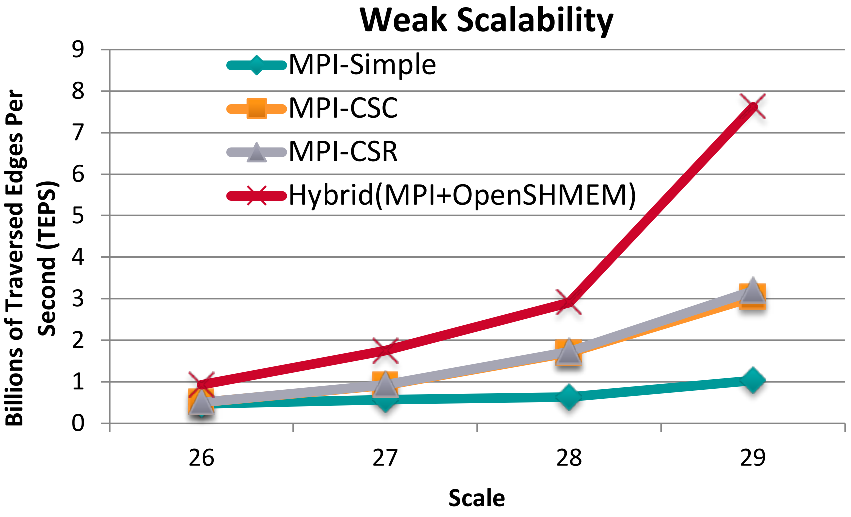
- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- Execution time for 2DHeat Image at 512 processes (sec):
  - UH-SHMEM – 523, OMPI-SHMEM – 214, Scalable-SHMEM – 193, MV2X-SHMEM – 169
- Execution time for DAXPY at 512 processes (sec):
  - UH-SHMEM – 57, OMPI-SHMEM – 56, Scalable-SHMEM – 9.2, MV2X-SHMEM – 5.2

J. Jose, J. Zhang, A. Venkatesh, S. Potluri, and D. K. Panda, A Comprehensive Performance Evaluation of OpenSHMEM Libraries on InfiniBand Clusters, OpenSHMEM Workshop (OpenSHMEM'14), March 2014

Hybrid MPI+OpenSHMEM Graph500 Design





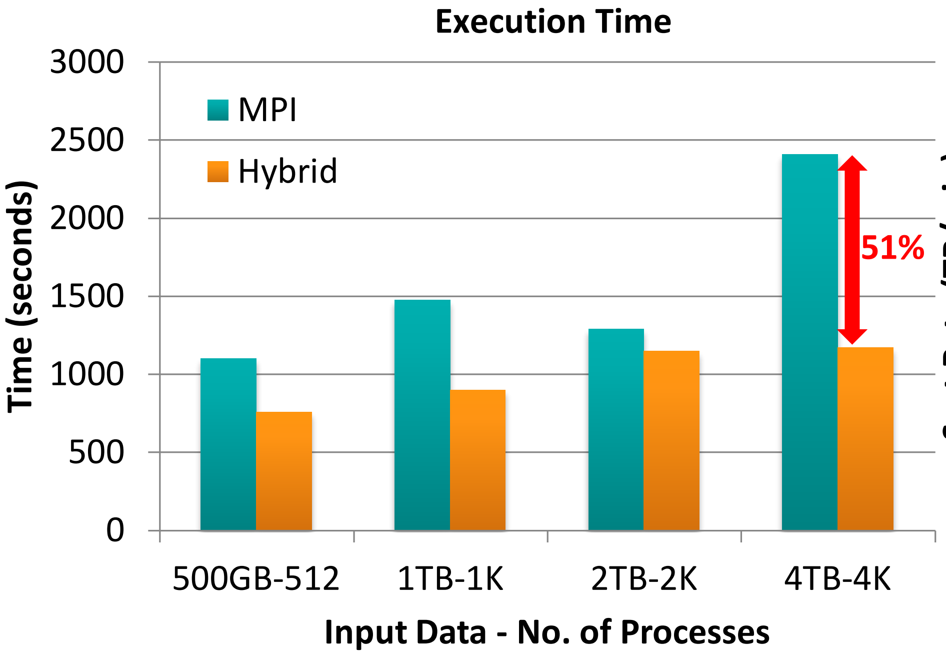


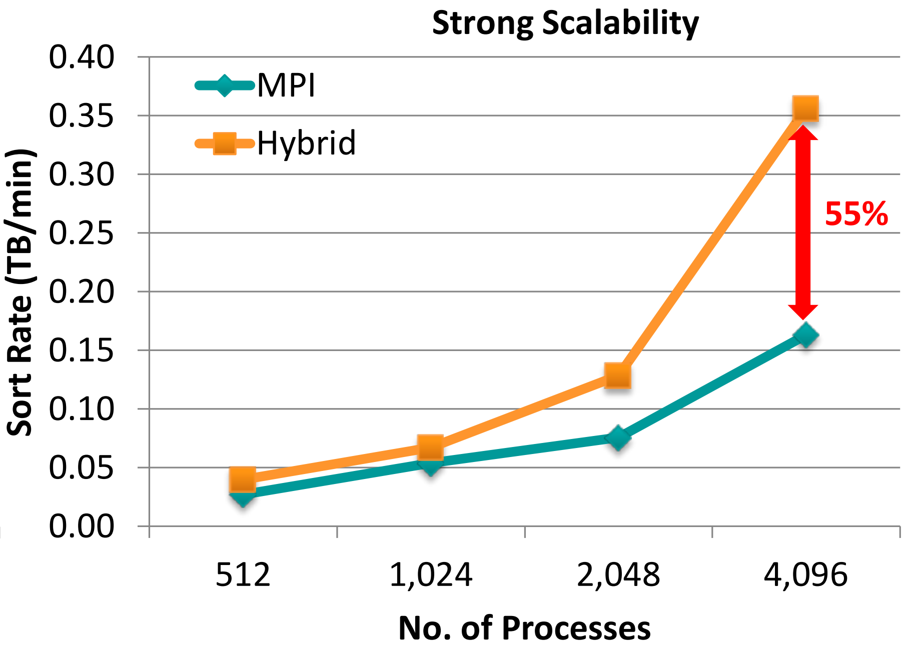
- Performance of Hybrid (MPI+OpenSHMEM) Graph500 Design
  - 8,192 processes
    - 2.4X improvement over MPI-CSR
    - 7.6X improvement over MPI-Simple
  - 16,384 processes
    - 1.5X improvement over MPI-CSR
    - 13X improvement over MPI-Simple

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC'13), June 2013

J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012

Hybrid MPI+OpenSHMEM Sort Application

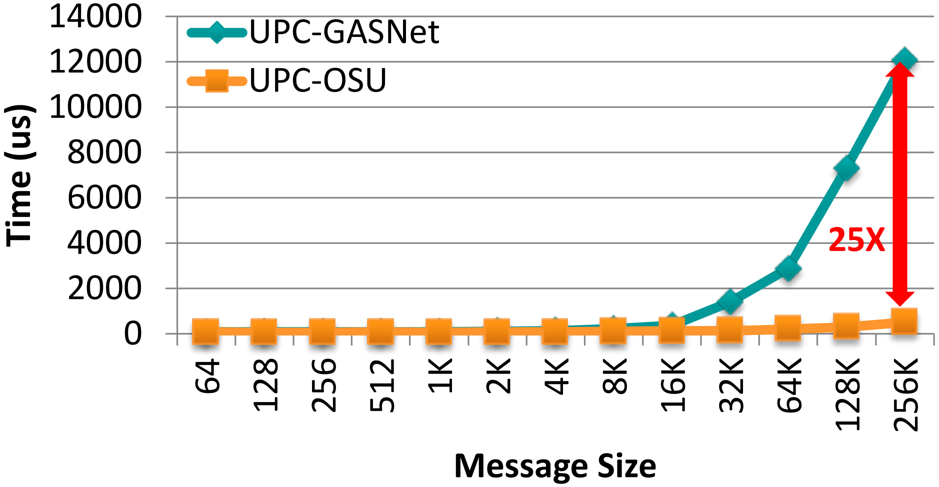




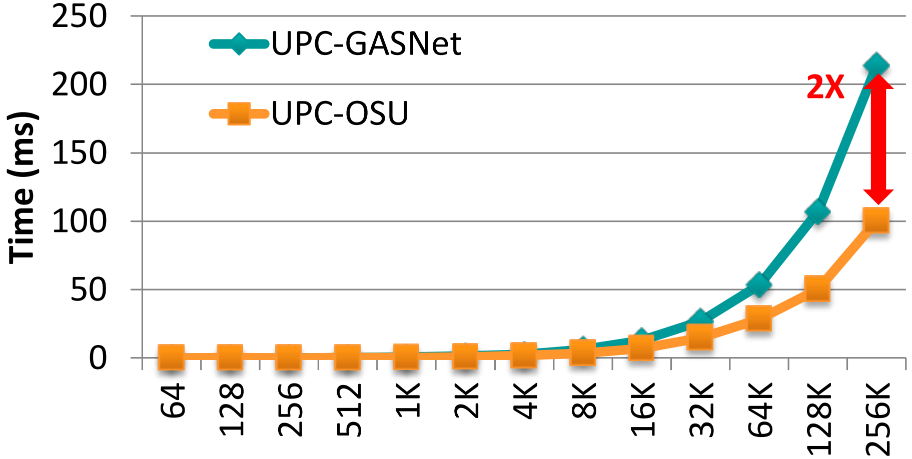
- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - Execution Time
    - 4TB Input size at 4,096 cores: MPI – 2408 seconds, Hybrid: 1172 seconds
    - **51%** improvement over MPI-based design
  - Strong Scalability **(**configuration: constant input size of 500GB)
    - **At 4,096 cores:** MPI – 0.16 TB/min, Hybrid – 0.36 TB/min
    - 55% improvement over MPI based design

UPC Collective Performance

Broadcast (2048 processes)

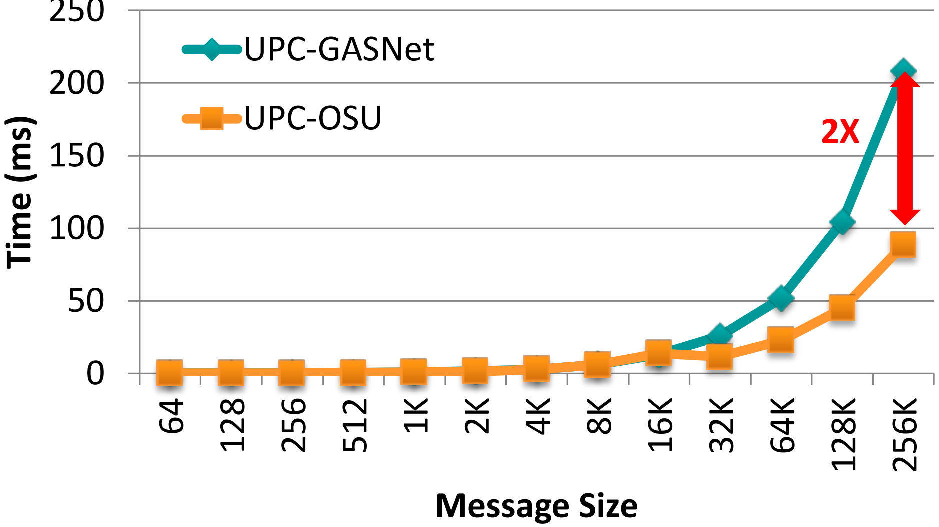


Scatter (2048 processes)

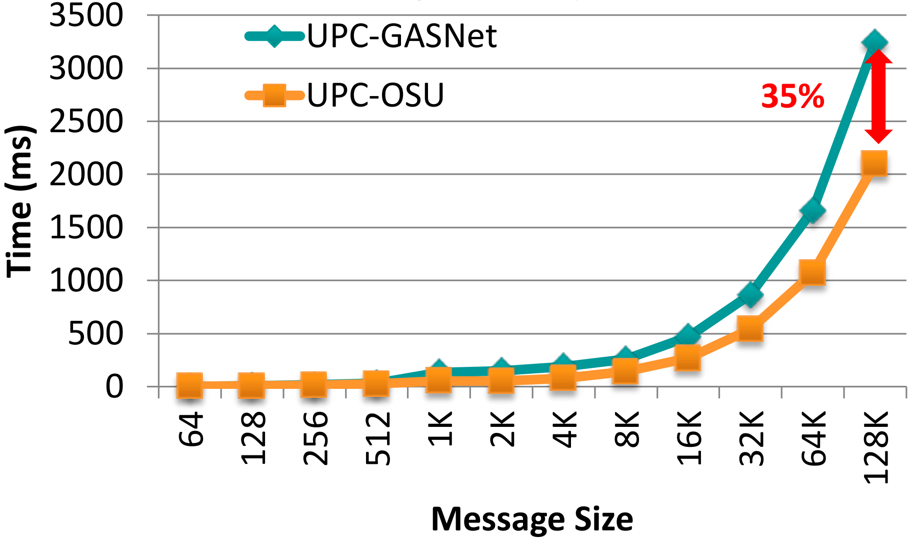


Message Size

Gather (2048 processes)

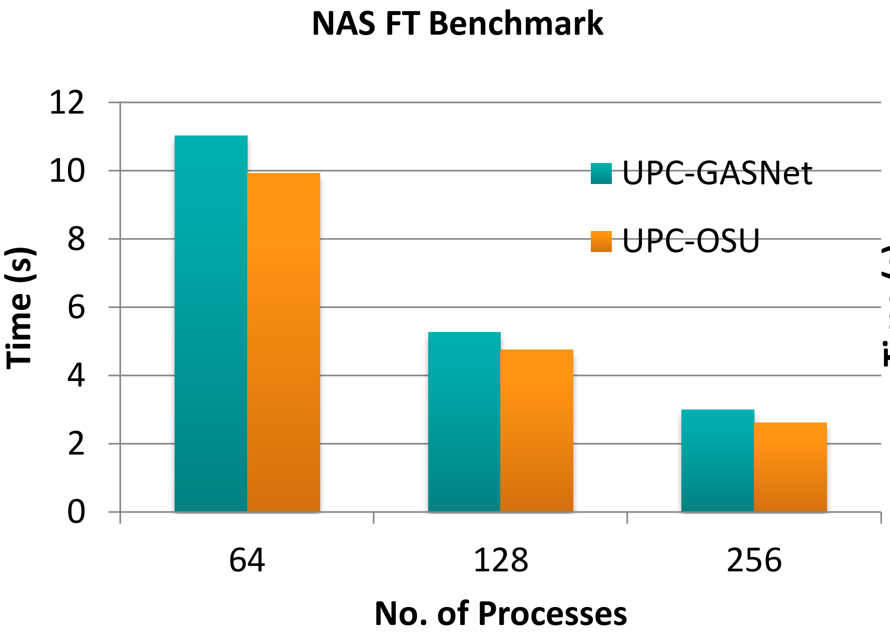


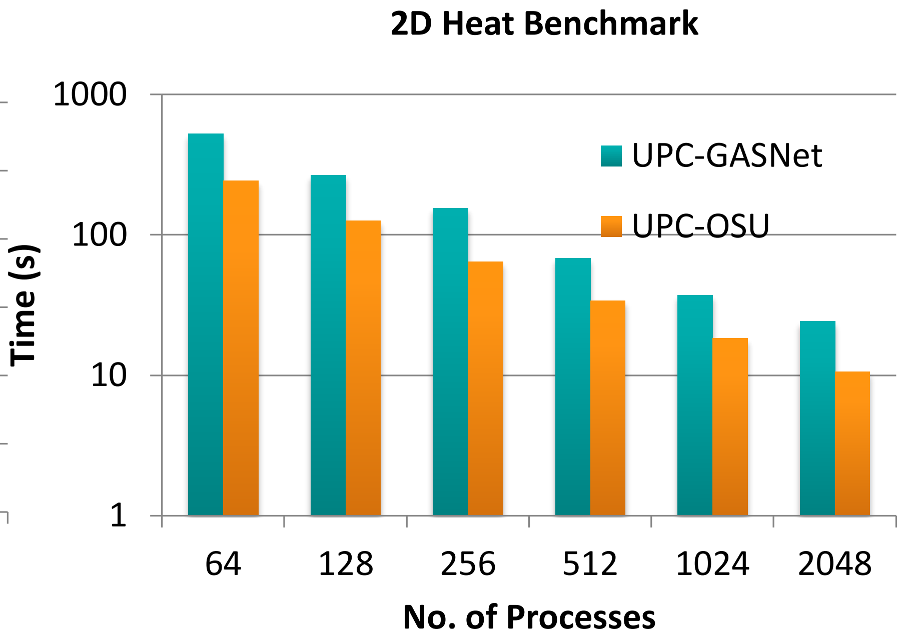
Exchange (2048 processes)



- Improved Performance for UPC collectives with OSU design
- OSU design takes advantage of well-researched MPI collectives

UPC Application Evaluation

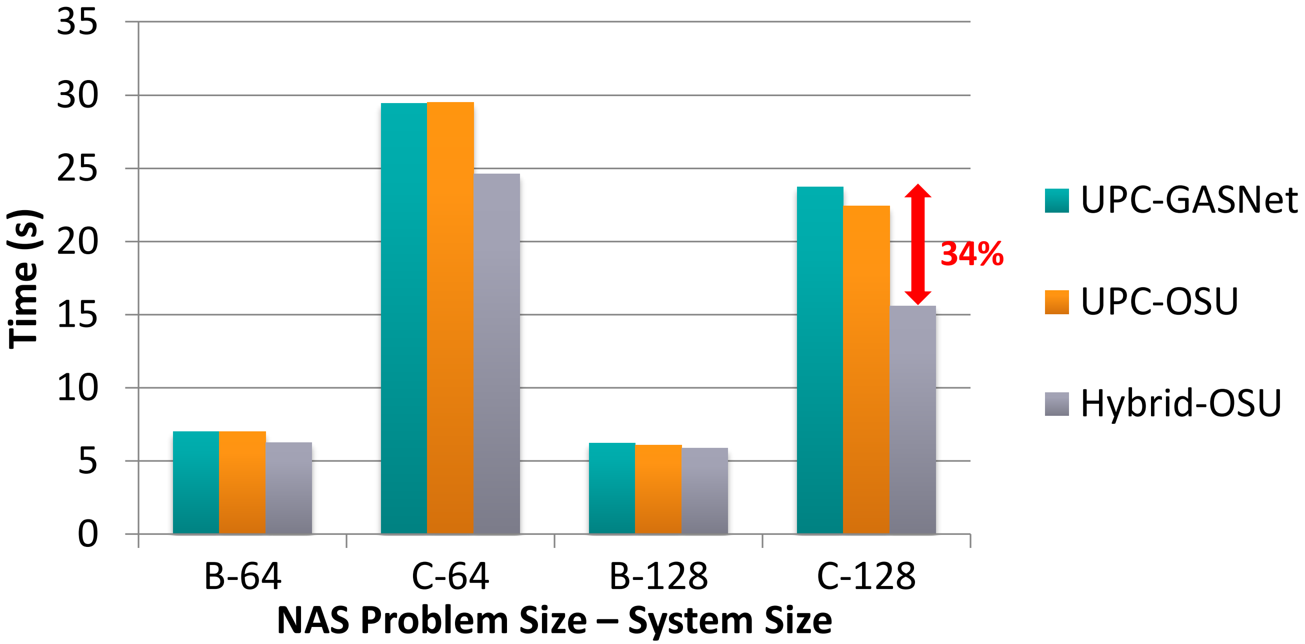




- Improved Performance with UPC applications
- NAS FT Benchmark (Class C)
  - Execution time at 256 processes: UPC-GASNet – 2.9 s, UPC-OSU – 2.6 s
- 2D Heat Benchmark
  - Execution time at 2,048 processes: UPC-GASNet – 24.1 s, UPC-OSU – 10.5 s

J. Jose, K. Hamidouche, J. Zhang, A. Venkatesh, and D. K. Panda, Optimizing Collective Communication in UPC, Int'l Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS '14), held in conjunction with International Parallel and Distributed Processing Symposium (IPDPS'14), May 2014

Hybrid MPI+UPC NAS-FT



- Modified NAS FT UPC all-to-all pattern using MPI_Alltoall
- Truly hybrid program
- For FT (Class C, 128 processes)
  - 34% improvement over UPC-GASNet
  - 30% improvement over UPC-OSU

J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, Fourth Conference on Partitioned Global Address Space Programming Model (PGAS '10), October 2010

MVAPICH2/MVPICH2-X – Plans for Exascale

- Performance and Memory scalability toward 500K-1M cores
  - Dynamically Connected Transport (DCT) service with Connect-IB
- Enhanced Optimization for GPGPU and Coprocessor Support
  - Extending the GPGPU support (GPU-Direct RDMA) with CUDA 6.0 and Beyond
  - Support for Intel MIC (Knight Landing)
- Taking advantage of Collective Offload framework
  - Including support for non-blocking collectives (MPI 3.0)
- RMA support (as in MPI 3.0)
- Extended topology-aware collectives
- Power-aware collectives
- Support for MPI Tools Interface (as in MPI 3.0)
- Checkpoint-Restart and migration support with in-memory checkpointing
- Hybrid MPI+PGAS programming support with GPGPUs and Accelertors

Accelerating Big Data with RDMA

- Big Data: Hadoop and Memcached

(April 2nd, 13:30-14:30pm)

Concluding Remarks

- InfiniBand with RDMA feature is gaining momentum in HPC systems with best performance and greater usage
- As the HPC community moves to Exascale, new solutions are needed in the MPI and Hybrid MPI+PGAS stacks for supporting GPUs and Accelerators
- Demonstrated how such solutions can be designed with MVAPICH2/MVAPICH2-X and their performance benefits
- Such designs will allow application scientists and engineers to take advantage of upcoming exascale systems

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- S. Chakraborthy (Ph.D.)
- N. Islam (Ph.D.)
- J. Jose (Ph.D.)
- M. Li (Ph.D.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekhar (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- R. Shir (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Current Senior Research Associate

- H. Subramoni

Current Post-Docs

- X. Lu
- K. Hamidouche

Current Programmers

- M. Arnold
- J. Perkins

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.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- 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.)
- 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.)
- 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 Post-Docs

- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne

Past Research Scientist

- S. Sur

Past Programmers

- D. Bureddy

Multiple Positions Available in My Group

- Looking for Bright and Enthusiastic Personnel to join as
  - Visiting Scientists
  - Post-Doctoral Researchers
  - PhD Students
  - MPI Programmer/Software Engineer
  - Hadoop/Big Data Programmer/Software Engineer
- If interested, please contact me at this conference and/or send an e-mail to panda@cse.ohio-state.edu

Pointers

Network Based Computing Laboratory

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

MVAPICH

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

panda@cse.ohio-state.edu

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