MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

High Performance and Scalable MPI+X Library for Emerging HPC Clusters

Talk at Intel HPC Developer Conference (SC '16)

by

Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

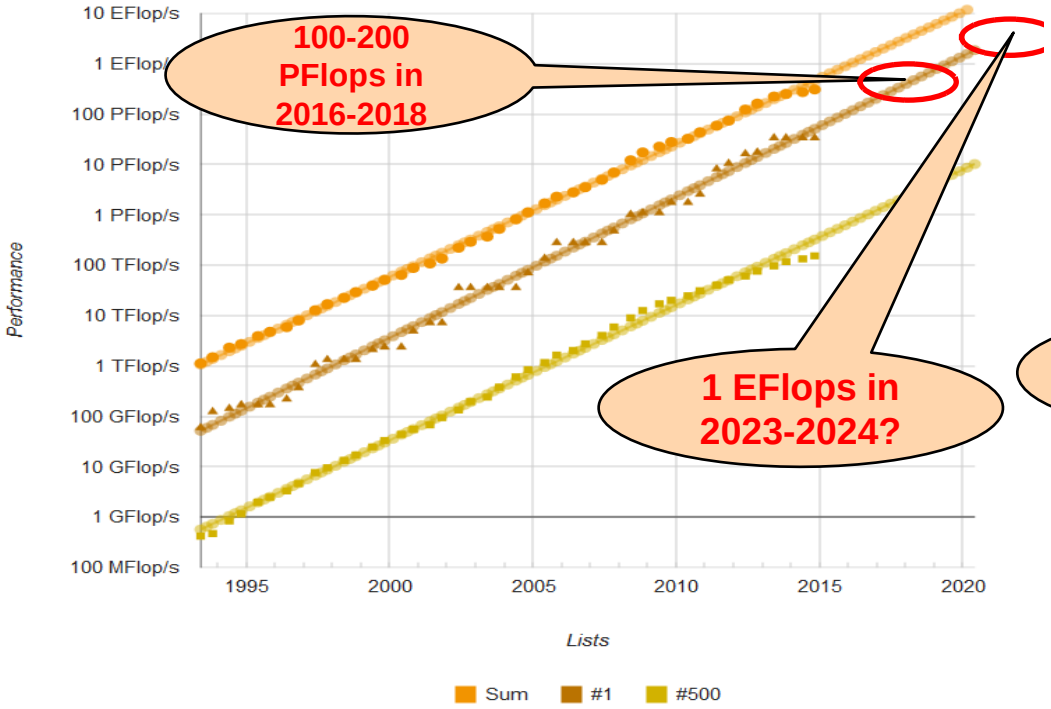
Khaled Hamidouche

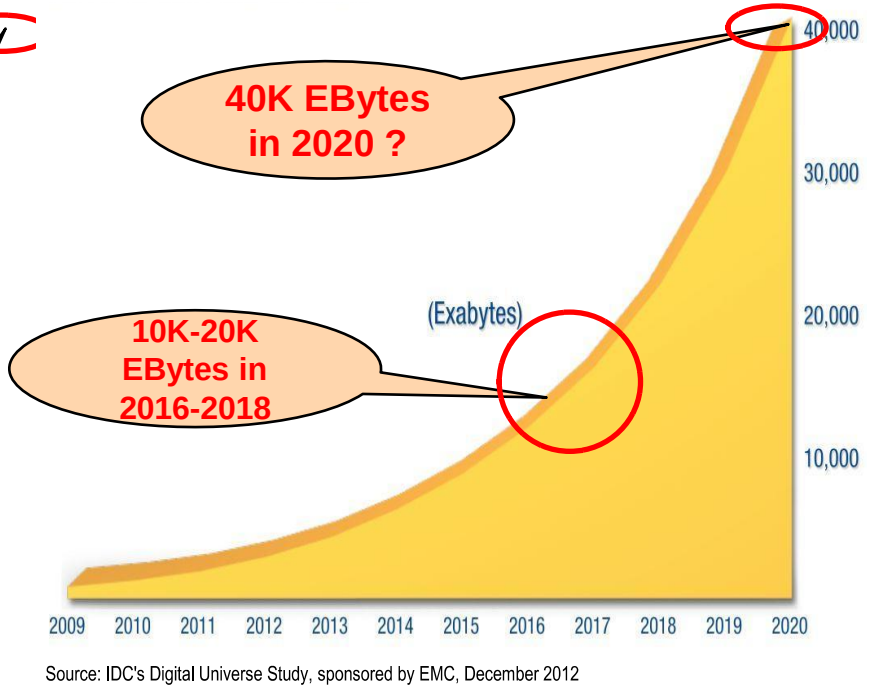
The Ohio State University

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

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

High-End Computing (HEC): ExaFlop & ExaByte



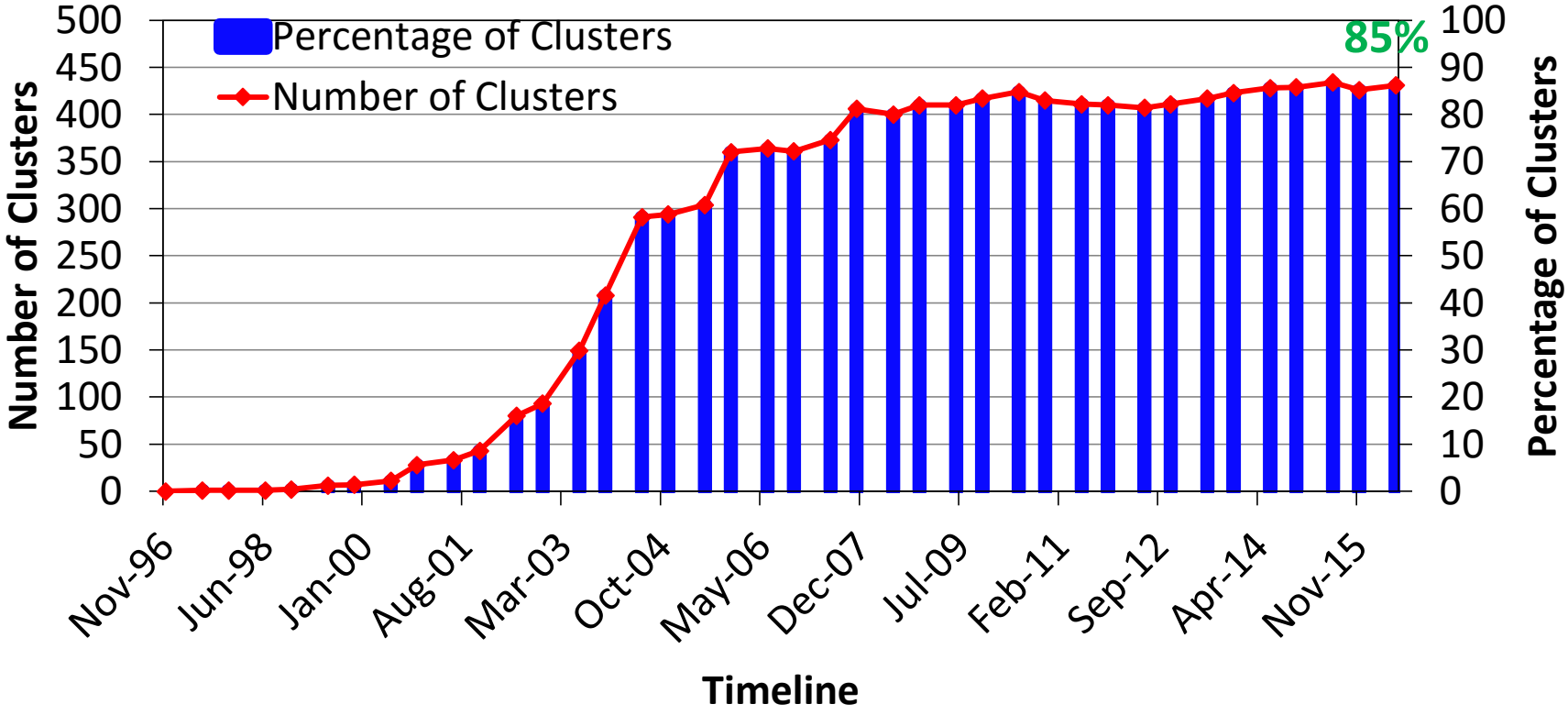


Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

ExaFlop & HPC

ExaByte & BigData

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

SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs and Intel Xeon Phi)

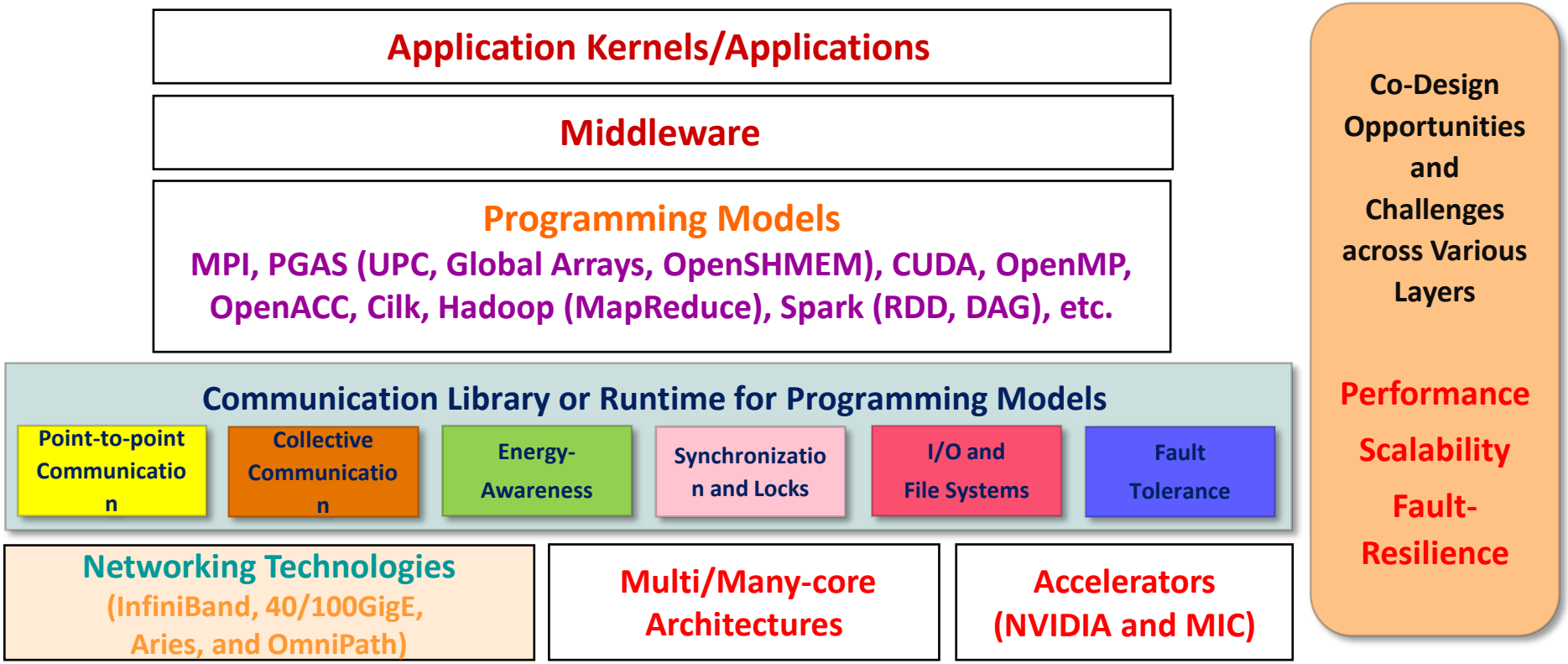
Tianhe – 2

Titan

Stampede

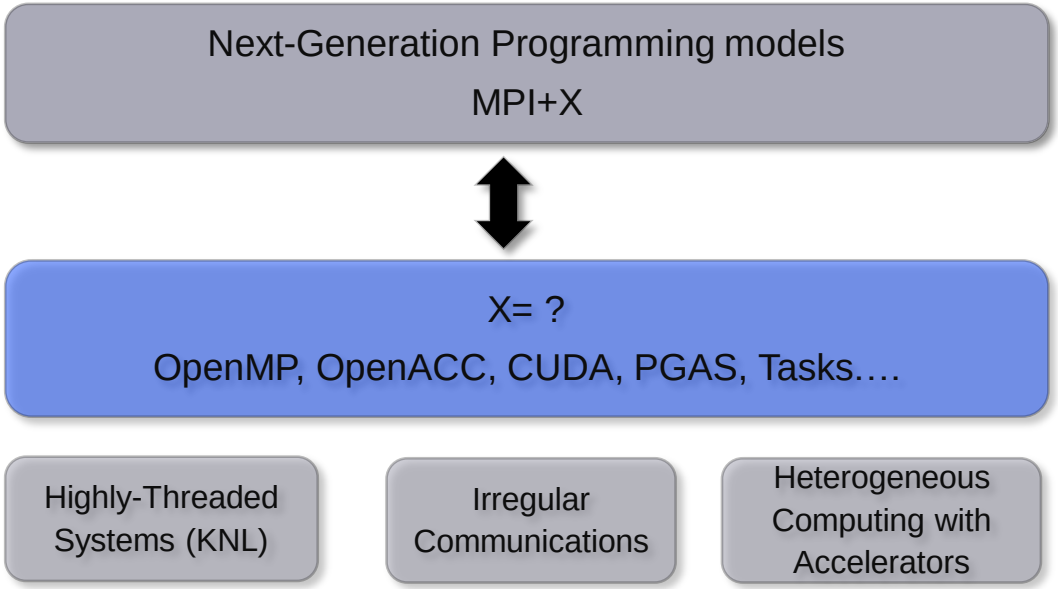
Tianhe – 1A

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



Exascale Programming models

- The community believes exascale programming model will be MPI+X
- But what is X?
  - Can it be just OpenMP?
- Many different environments and systems are emerging
  - Different `X' will satisfy the respective needs



MPI+X Programming model: Broad Challenges at Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Scalable job start-up
- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
- Balancing intra-node and inter-node communication for next generation nodes (128-1024 cores)
  - Multiple end-points per node
- Support for efficient multi-threading (OpenMP)
- Integrated Support for GPGPUs and Accelerators (CUDA)
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, …)
- Virtualization
- Energy-Awareness

Additional Challenges for Designing Exascale Software Libraries

- **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

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - **Used by more than 2,690 organizations in 83 countries**
  - **More than 402,000 (> 0.4 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Nov '16 ranking)
    - **1st ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China**
    - 13th ranked 241,108-core cluster (Pleiades) at NASA
    - 17th ranked 519,640-core cluster (Stampede) at TACC
    - 40th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu
- Empowering Top500 systems for over a decade
  - System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
    Sunway TaihuLight at NSC, Wuxi, China (1st in Nov'16, 10,649,640 cores, 93 PFlops)

Outline

- Hybrid MPI+OpenMP Models for Highly-threaded Systems
- Hybrid MPI+PGAS Models for Irregular Applications
- Hybrid MPI+GPGPUs and OpenSHMEM for Heterogeneous Computing with Accelerators

Highly Threaded Systems

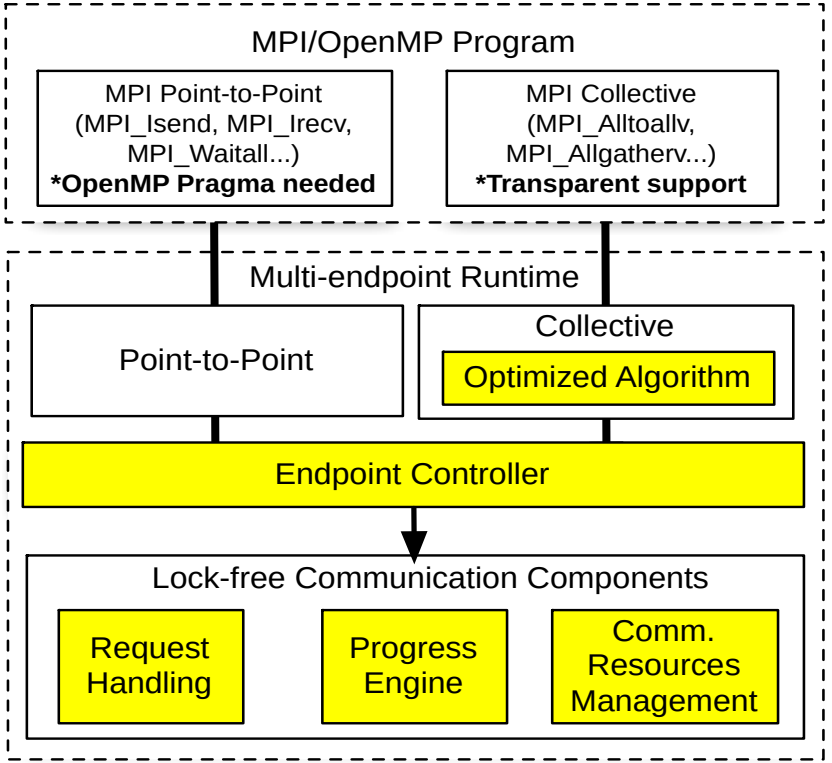
- Systems like KNL
- MPI+OpenMP is seen as the best fit
  - 1 MPI process per socket for Multi-core
  - 4-8 MPI processes per KNL
  - Each MPI process will launch OpenMP threads
- However, current MPI runtimes are not “efficiently” handling the hybrid
  - Most of the application use Funneled mode: Only the MPI processes perform communication
  - Communication phases are the bottleneck
- Multi-endpoint based designs
  - Transparently use threads inside MPI runtime
  - Increase the concurrency

MPI and OpenMP

- MPI-4 will enhance the thread support
  - Endpoint proposal in the Forum
  - Application threads will be able to efficiently perform communication
  - Endpoint is the communication entity that maps to a thread
    - Idea is to have multiple addressable communication entities within a single process
    - No context switch between application and runtime => better performance
- OpenMP 4.5 is more powerful than just traditional data parallelism
  - Supports task parallelism since OpenMP 3.0
  - Supports heterogeneous computing with accelerator targets since OpenMP 4.0
  - Supports explicit SIMD and threads affinity pragmas since OpenMP 4.0

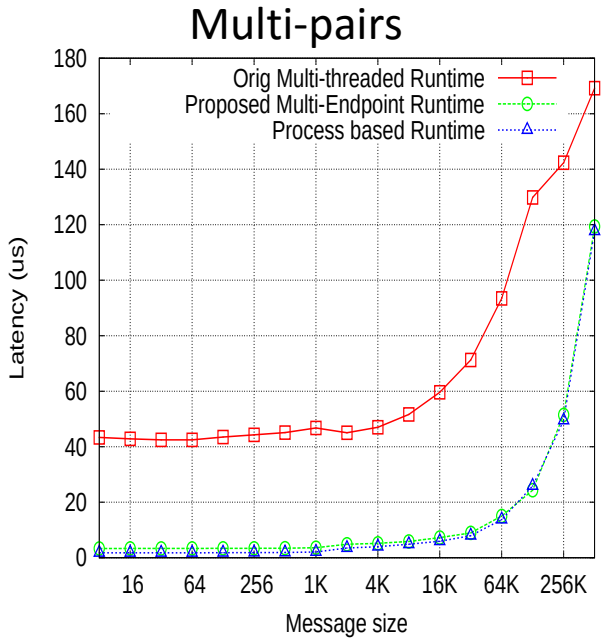
MEP-based design: MVAPICH2 Approach

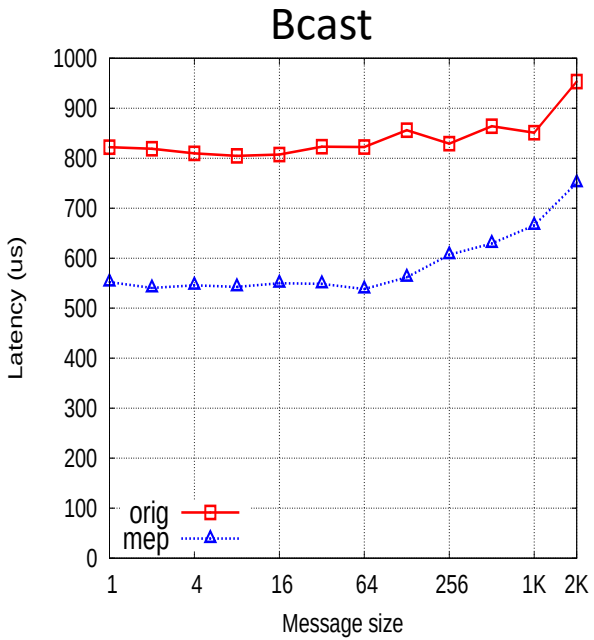
- Lock-free Communication
  - Threads have their own resources
- Dynamically adapt the number of threads
  - Avoid resource contention
  - Depends on application pattern and system performance
- Both intra- and inter-nodes communication
  - Threads boost both channels
- New MEP-Aware collectives
- Applicable to the endpoint proposal in MPI-4

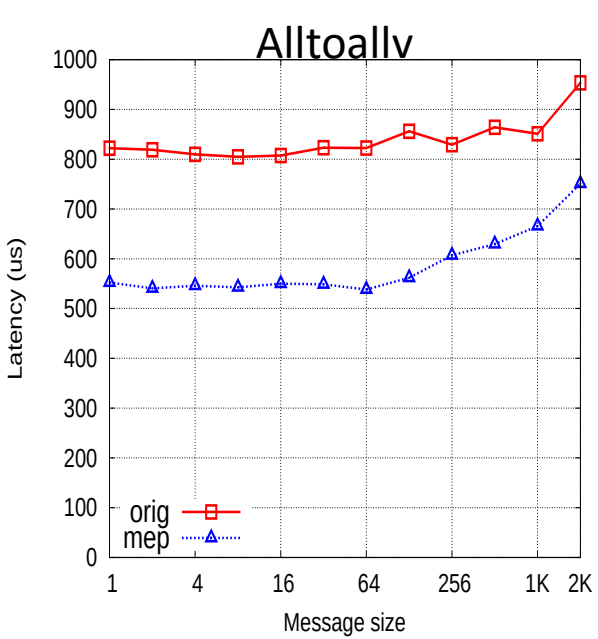


M. Luo, X. Lu, K. Hamidouche, K. Kandalla and D. K. Panda, Initial Study of Multi-Endpoint Runtime for MPI+OpenMP Hybrid Applications on Multi-Core Systems. International Symposium on Principles and Practice of Parallel Programming (PPoPP '14).

Performance Benefits: OSU Micro-Benchmarks (OMB) level

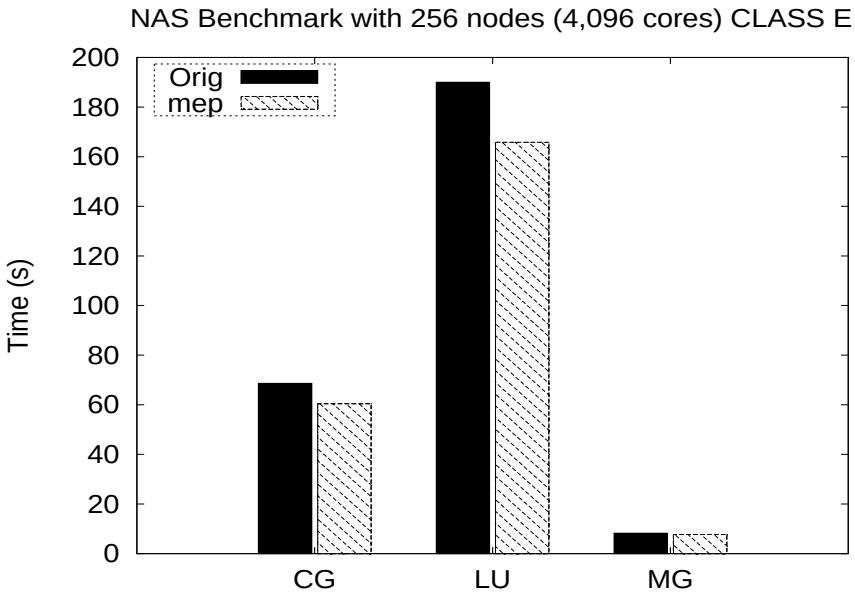


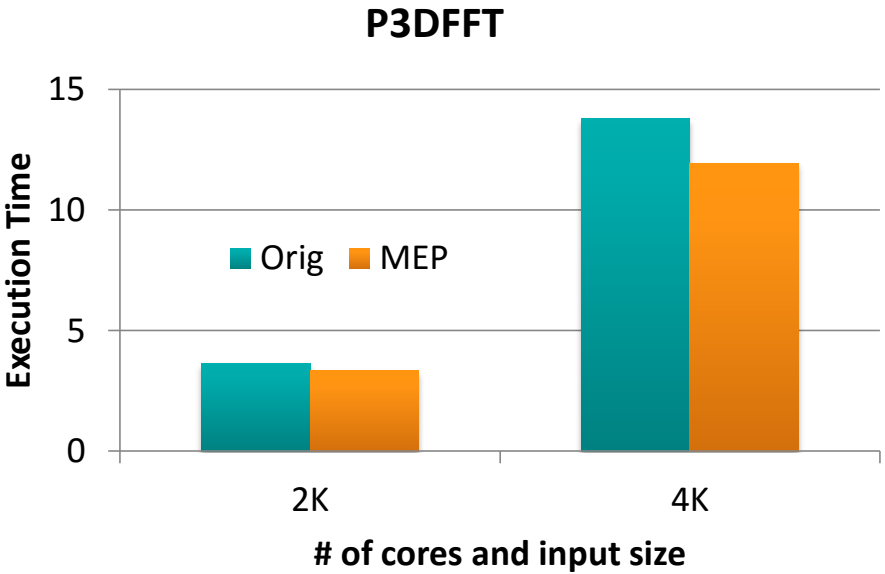




- Reduces the latency from 40us to 1.85 us (21X)
- Achieves the same as Processes
- 40% improvement on latency for Bcast on 4,096 cores
- 30% improvement on latency for Alltoall on 4,096 cores

Performance Benefits: Application Kernel level



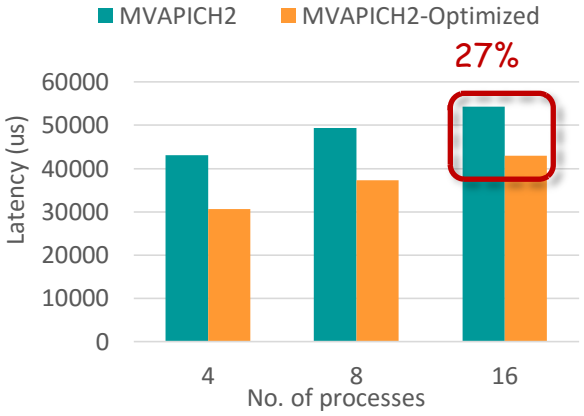


- 6.3% improvement for MG, 11.7% improvement for CG, and 12.6% improvement for LU on 4,096 cores.
- With P3DFFT, we are able to observe a 30% improvement in communication time and 13.5% improvement in the total execution time.

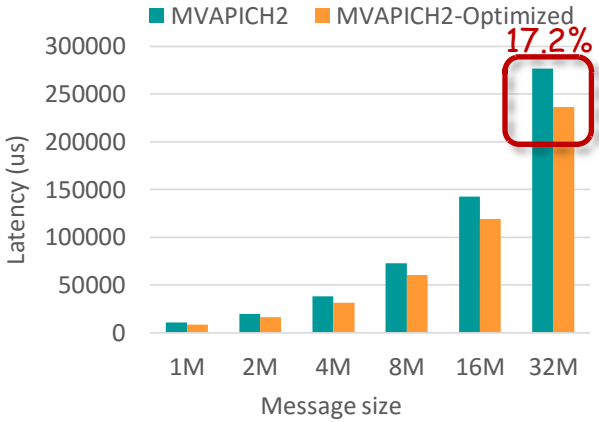
Enhanced Designs for KNL: MVAPICH2 Approach

- On-load approach
  - Takes advantage of the idle cores
  - Dynamically configurable
  - Takes advantage of highly multithreaded cores
  - Takes advantage of MCDRAM of KNL processors
- Applicable to other programming models such as PGAS, Task-based, etc.
- Provides portability, performance, and applicability to runtime as well as applications in a transparent manner

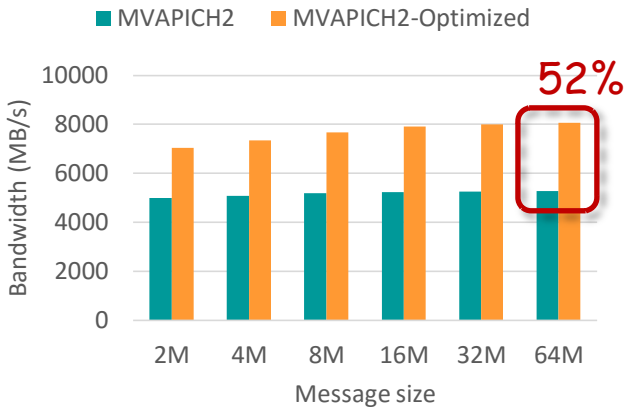
Performance Benefits of the Enhanced Designs



Intra-node Broadcast with 64MB Message



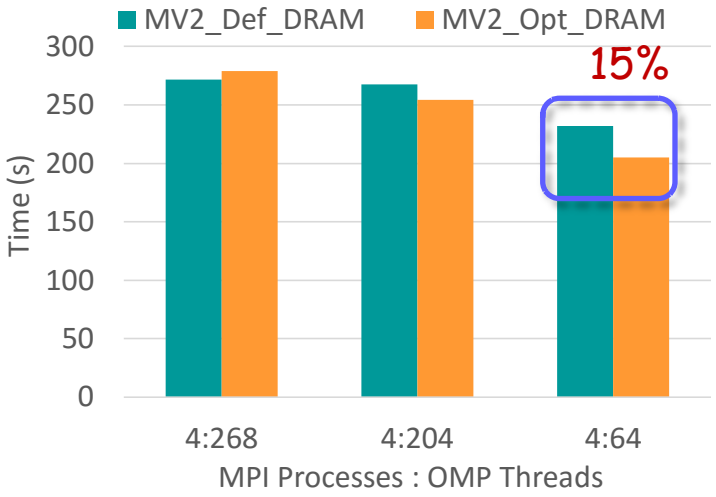
16-process Intra-node All-to-All



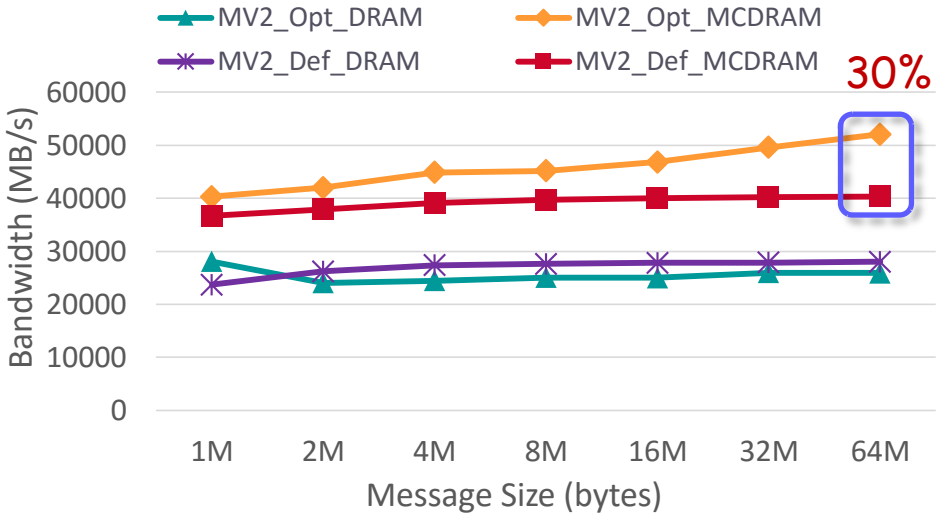
Very Large Message Bi-directional Bandwidth

- New designs to exploit high concurrency and MCDRAM of KNL
- Significant improvements for large message sizes
- Benefits seen in varying message size as well as varying MPI processes

Performance Benefits of the Enhanced Designs



CNTK: MLP Training Time using MNIST (BS:64)



Multi-Bandwidth using 32 MPI processes

- Benefits observed on training time of Multi-level Perceptron (MLP) model on MNIST dataset using CNTK Deep Learning Framework

Enhanced Designs will be available in upcoming MVAPICH2 releases

Outline

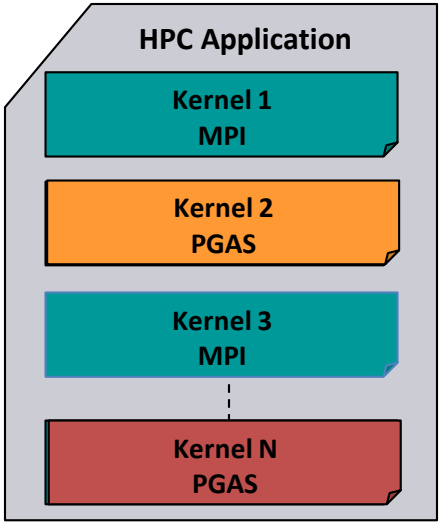
- Hybrid MPI+OpenMP Models for Highly-threaded Systems
- Hybrid MPI+PGAS Models for Irregular Applications
- Hybrid MPI+GPGPUs and OpenSHMEM for Heterogeneous Computing with Accelerators

Maturity of Runtimes and Application Requirements

- MPI has been the most popular model for a long time
  - Available on every major machine
  - Portability, performance and scaling
  - Most parallel HPC code is designed using MPI
  - Simplicity - structured and iterative communication patterns
- PGAS Models
  - Increasing interest in community
  - Simple shared memory abstractions and one-sided communication
  - Easier to express irregular communication
- Need for hybrid MPI + PGAS
  - Application can have kernels with different communication characteristics
  - Porting only part of the applications to reduce programming effort

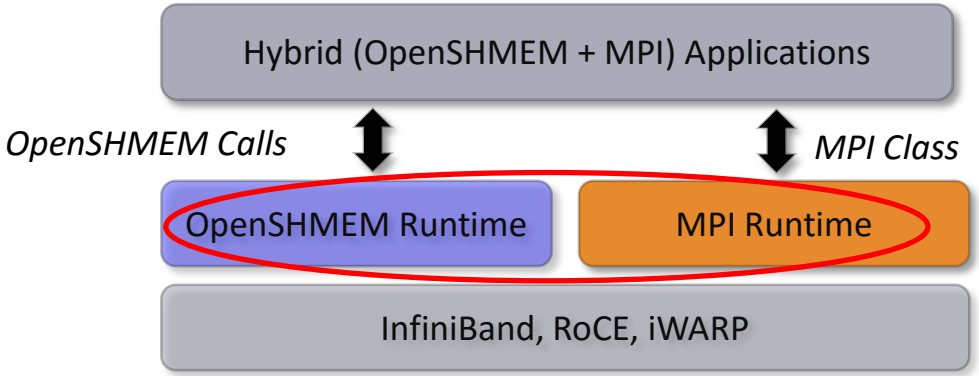
Hybrid (MPI+PGAS) Programming

- 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



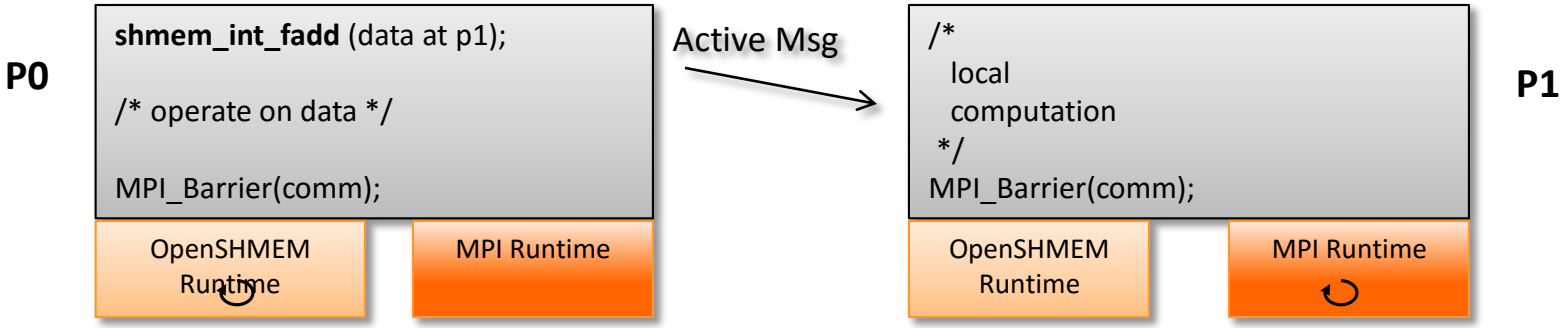
Current Approaches for Hybrid Programming

- Layering one programming model over another
  - Poor performance due to semantics mismatch
  - MPI-3 RMA tries to address
- Separate runtime for each programming model



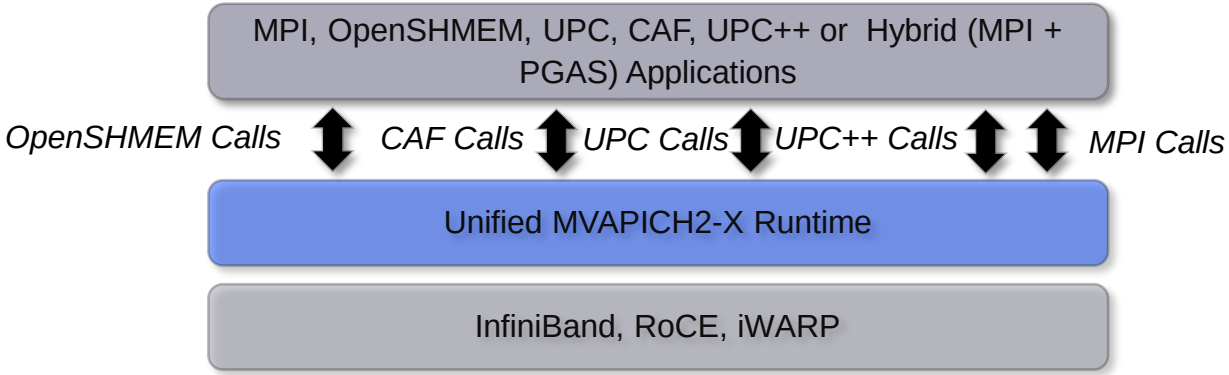
- Need more network and memory resources
- Might lead to deadlock!

The Need for a Unified Runtime



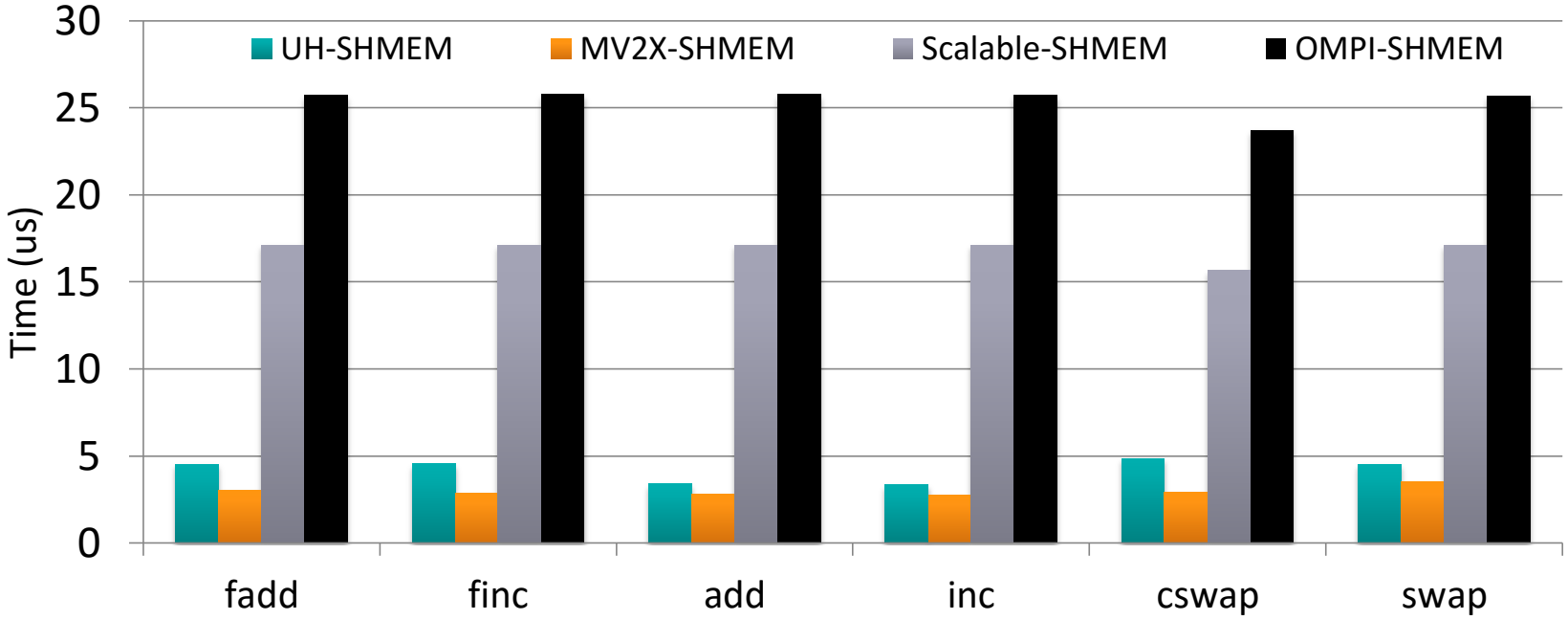
- Deadlock when a message is sitting in one runtime, but application calls the other runtime
- Prescription to avoid this is to barrier in one mode (either OpenSHMEM or MPI) before entering the other
- Or runtimes require dedicated progress threads
- Bad performance!!
- Similar issues for MPI + UPC applications over individual runtimes

MVAPICH2-X for Hybrid MPI + PGAS Applications



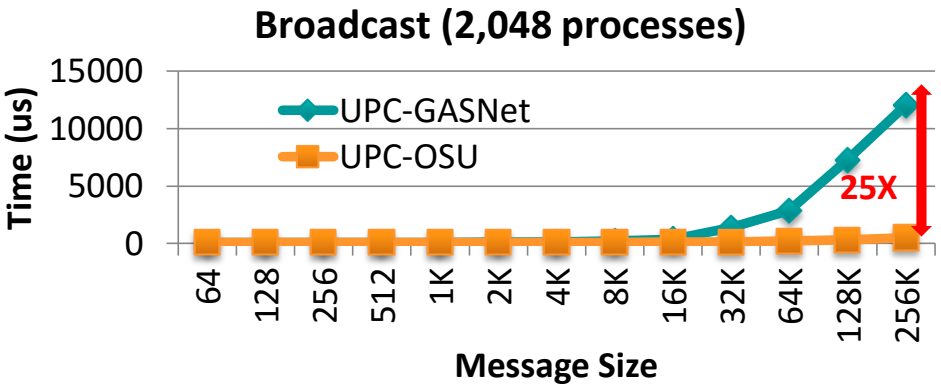
- Unified communication runtime for MPI, UPC, OpenSHMEM, CAF, UPC++ available with MVAPICH2-X 1.9 onwards! (since 2012)
  - http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, UPC++, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant (with initial support for UPC 1.3), CAF 2008 standard (OpenUH), UPC++
  - Scalable Inter-node and intra-node communication – point-to-point and collectives

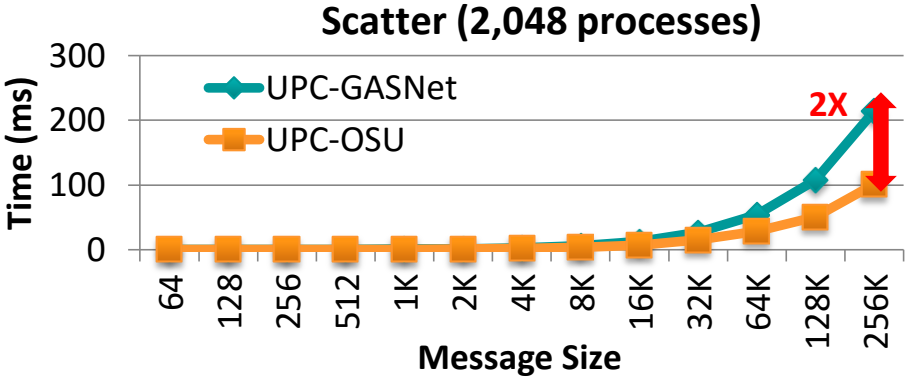
OpenSHMEM Atomic Operations: Performance

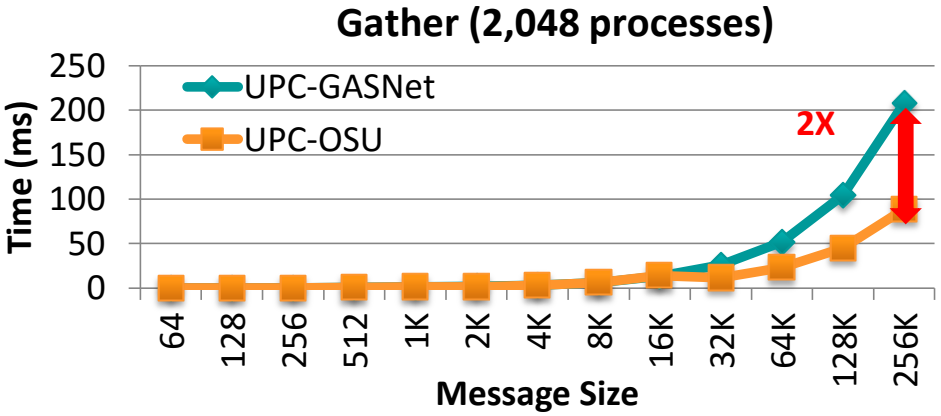


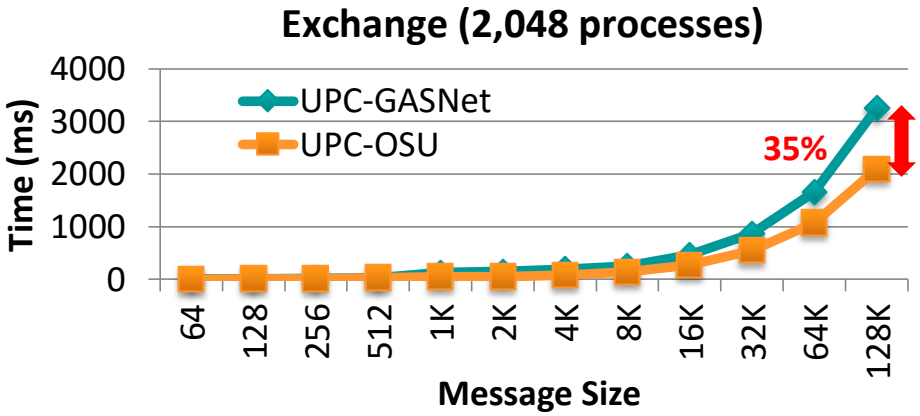
- OSU OpenSHMEM micro-benchmarks (OMB v4.1)
- MV2-X SHMEM performs up to **40%** better compared to UH-SHMEM

UPC Collectives Performance



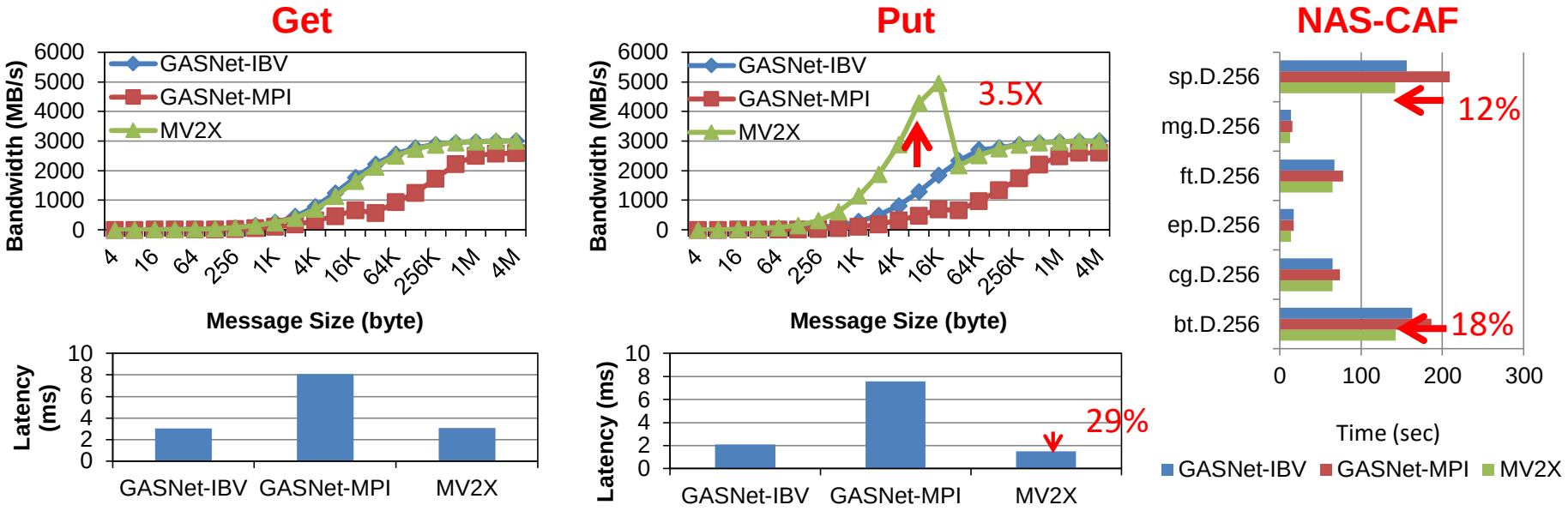






J. Jose, K. Hamidouche, J. Zhang, A. Venkatesh, and D. K. Panda, Optimizing Collective Communication in UPC (HiPS'14, in association with IPDPS'14)

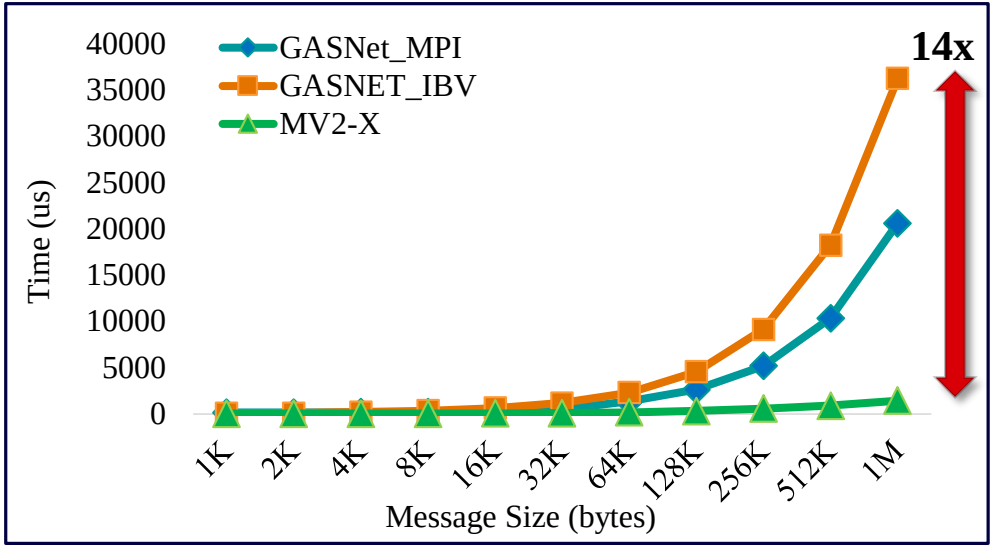
Performance Evaluations for CAF model



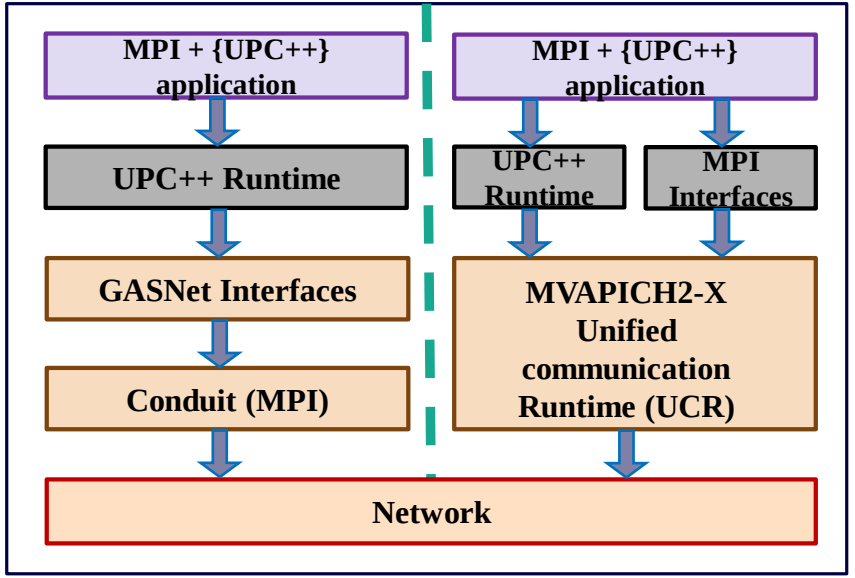
- Micro-benchmark improvement (MV2X vs. GASNet-IBV, UH CAF test-suite)
  - Put bandwidth: 3.5X improvement on 4KB; Put latency: reduce 29% on 4B
- Application performance improvement (NAS-CAF one-sided implementation)
  - Reduce the execution time by 12% (SP.D.256), 18% (BT.D.256)

J. Lin, K. Hamidouche, X. Lu, M. Li and D. K. Panda, High-performance Co-array Fortran support with MVAPICH2-X: Initial experience and evaluation, HIPS'15

UPC++ Collectives Performance



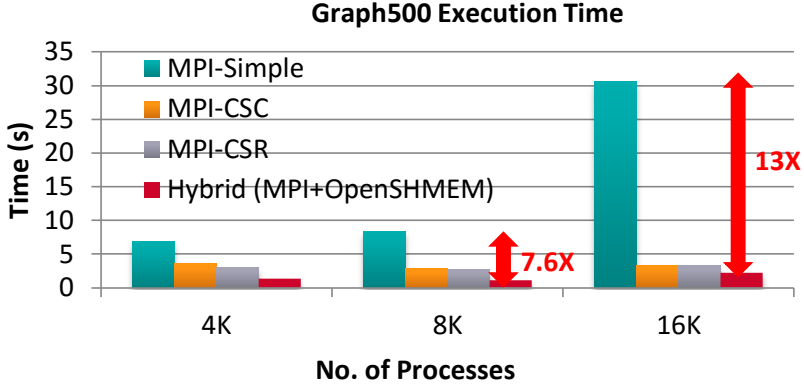
Inter-node Broadcast (64 nodes 1:ppn)

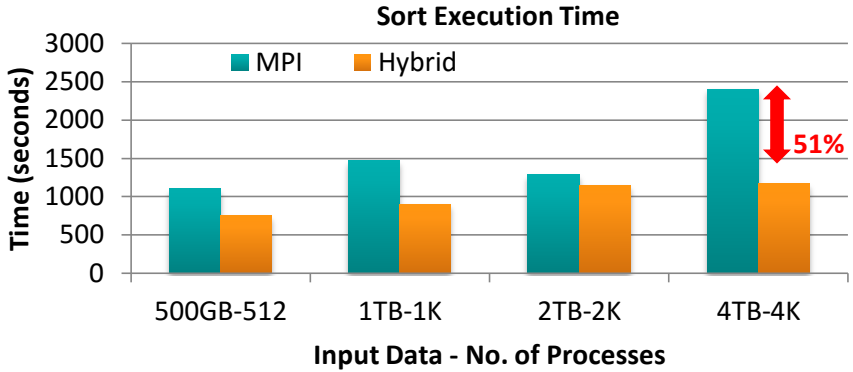


- Full and native support for hybrid MPI + UPC++ applications
- Better performance compared to IBV and MPI conduits
- OSU Micro-benchmarks (OMB) support for UPC++
- Available since MVAPICH2-X 2.2RC1

J. M. Hashmi, K. Hamidouche, and D. K. Panda, Enabling Performance Efficient Runtime Support for hybrid MPI+UPC++ Programming Models, IEEE International Conference on High Performance Computing and Communications (HPCC 2016)

Application Level Performance with Graph500 and Sort





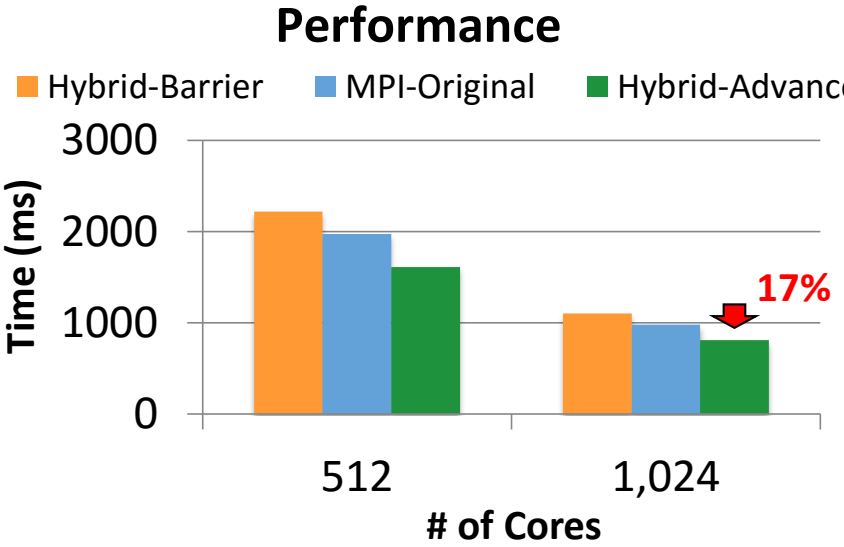
- Performance of Hybrid (MPI+ OpenSHMEM) Graph500 Design
  - 8,192 processes
    - **2.4X** improvement over MPI-CSR
    - **7.6X** improvement over MPI-Simple
  - 16,384 processes
    - **1.5X** improvement over MPI-CSR
    - **13X** improvement over MPI-Simple

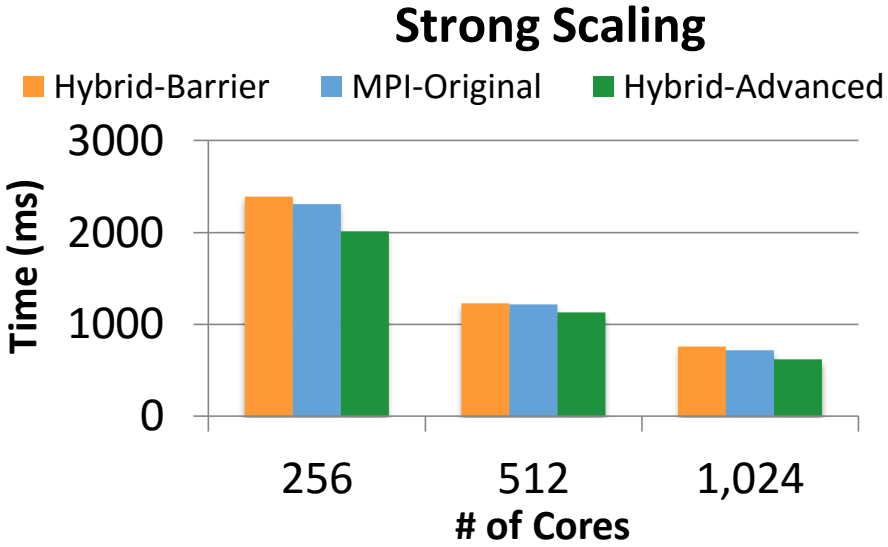
- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - 4,096 processes, 4 TB Input Size
    - MPI – 2408 sec; 0.16 TB/min
    - Hybrid – 1172 sec; 0.36 TB/min
    - **51%** improvement over MPI-design

J. Jose, S. Potluri, H. Subramoni, X. Lu, K. Hamidouche, K. Schulz, H. Sundar and D. Panda Designing Scalable Out-of-core Sorting with Hybrid MPI+PGAS Programming Models, PGAS'14

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

MiniMD – Total Execution Time



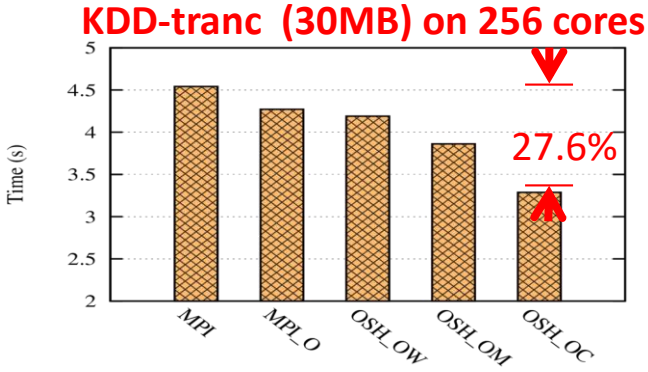


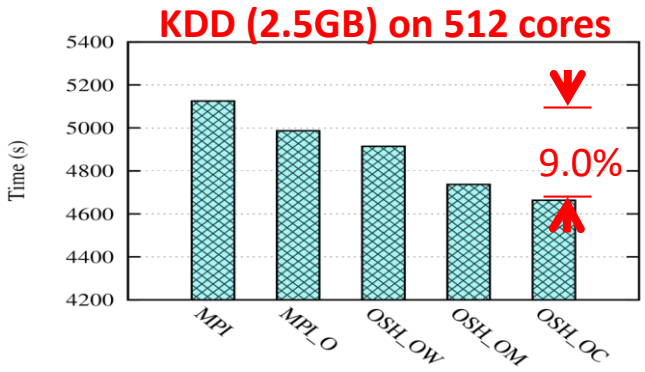
- Hybrid design performs better than MPI implementation
- 1,024 processes
  - 17% improvement over MPI version
- Strong Scaling

  Input size: 128 * 128 * 128

Accelerating MaTEx k-NN with Hybrid MPI and OpenSHMEM

- **MaTEx:** MPI-based Machine learning algorithm library
- **k-NN:** a popular supervised algorithm for classification
- **Hybrid designs:**
  - Overlapped Data Flow; One-sided Data Transfer; Circular-buffer Structure





- Benchmark: KDD Cup 2010 (8,407,752 records, 2 classes, k=5)
- For truncated KDD workload on 256 cores, reduce 27.6% execution time
- For full KDD workload on 512 cores, reduce 9.0% execution time

J. Lin, K. Hamidouche, J. Zhang, X. Lu, A. Vishnu, D. Panda. Accelerating k-NN Algorithm with Hybrid MPI and OpenSHMEM, OpenSHMEM 2015

Outline

- Hybrid MPI+OpenMP Models for Highly-threaded Systems
- Hybrid MPI+PGAS Models for Irregular Applications
- Hybrid MPI+GPGPUs and OpenSHMEM for Heterogeneous Computing with Accelerators

GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GPU

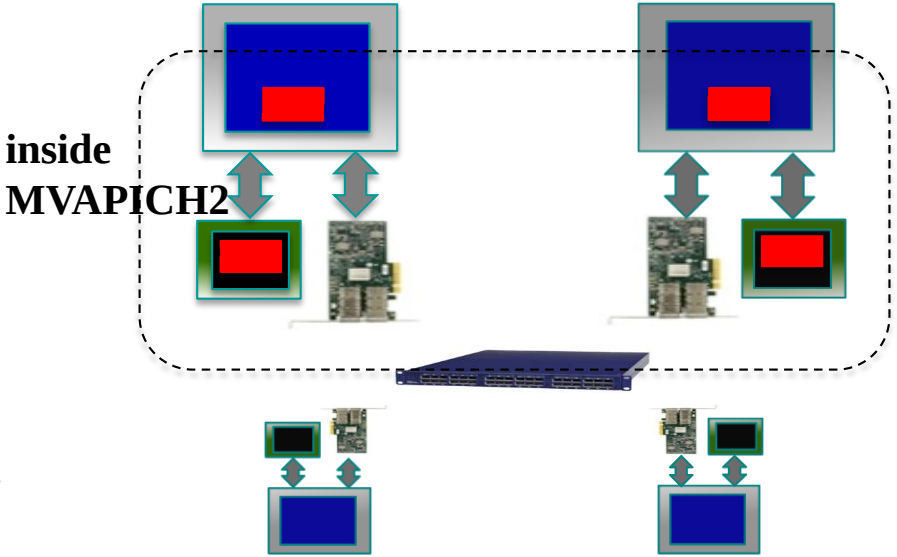
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, …);
```

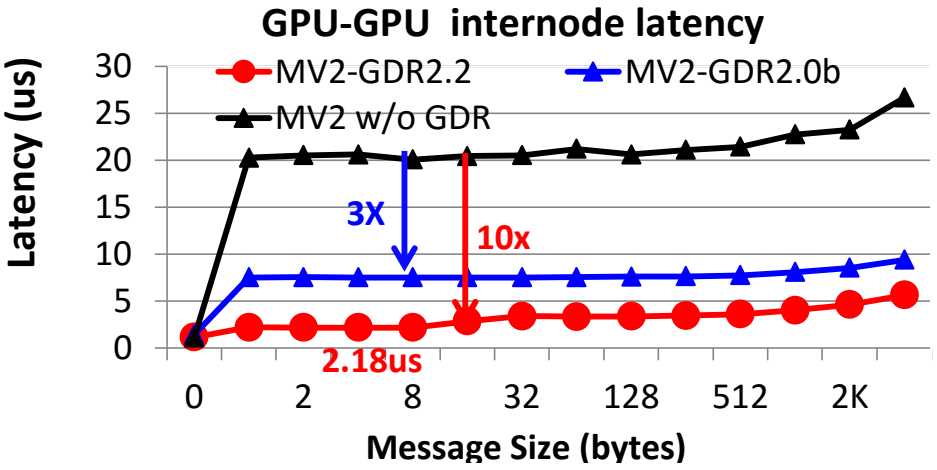


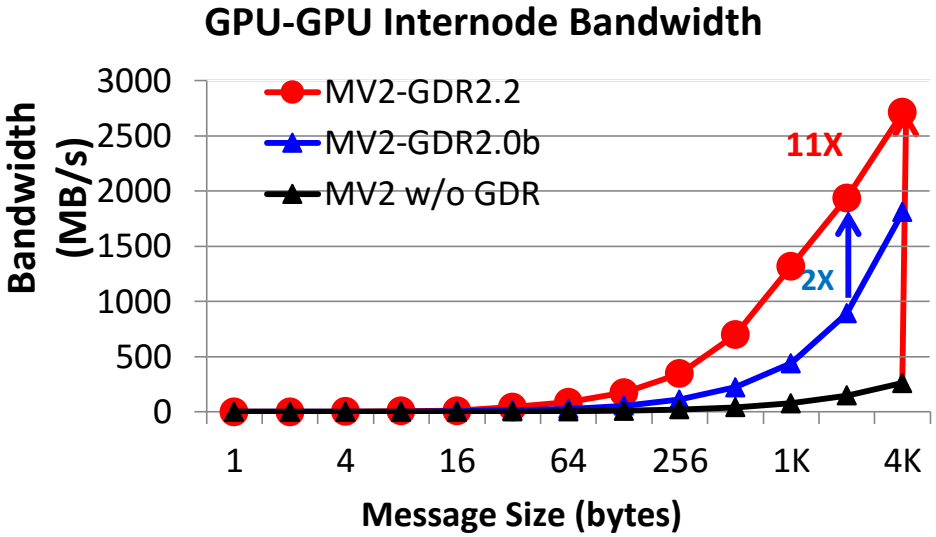
High Performance and High Productivity

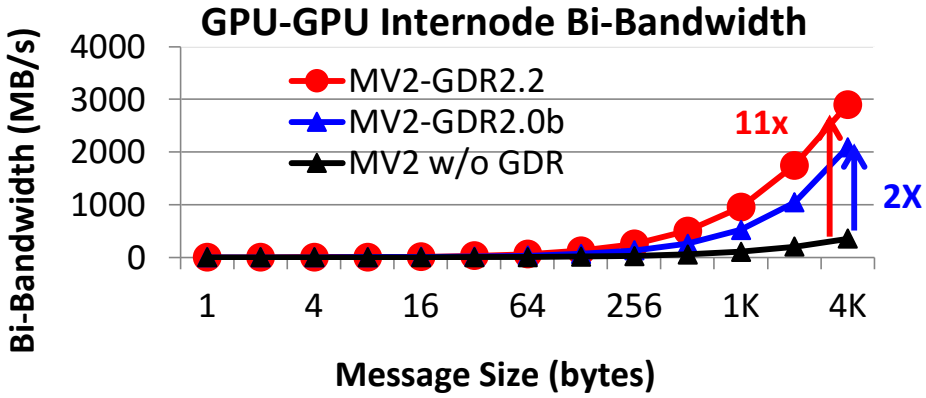
CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.2 Releases

- 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
- Unified memory

Performance of MVAPICH2-GPU with GPU-Direct RDMA (GDR)

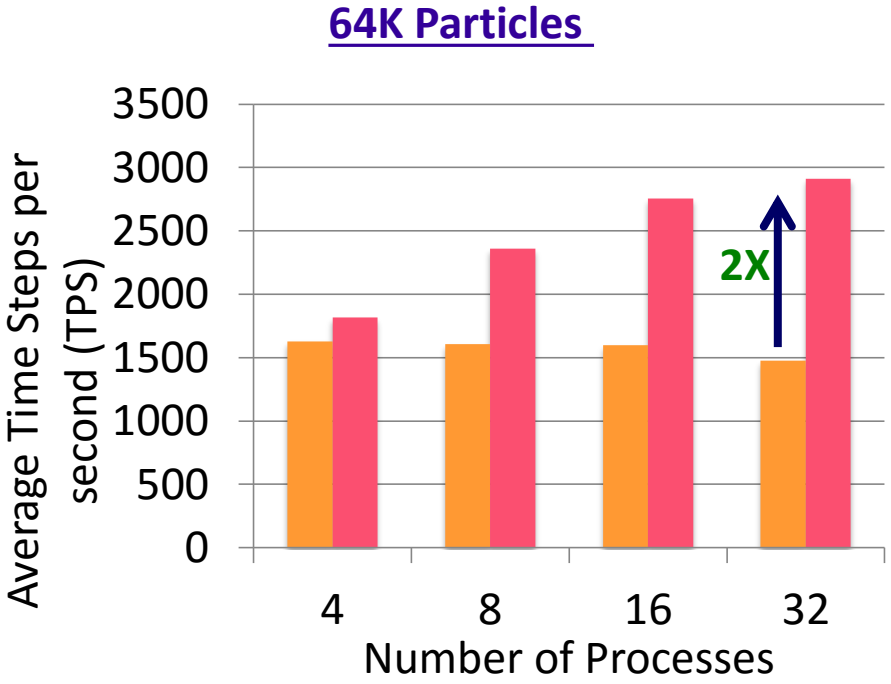


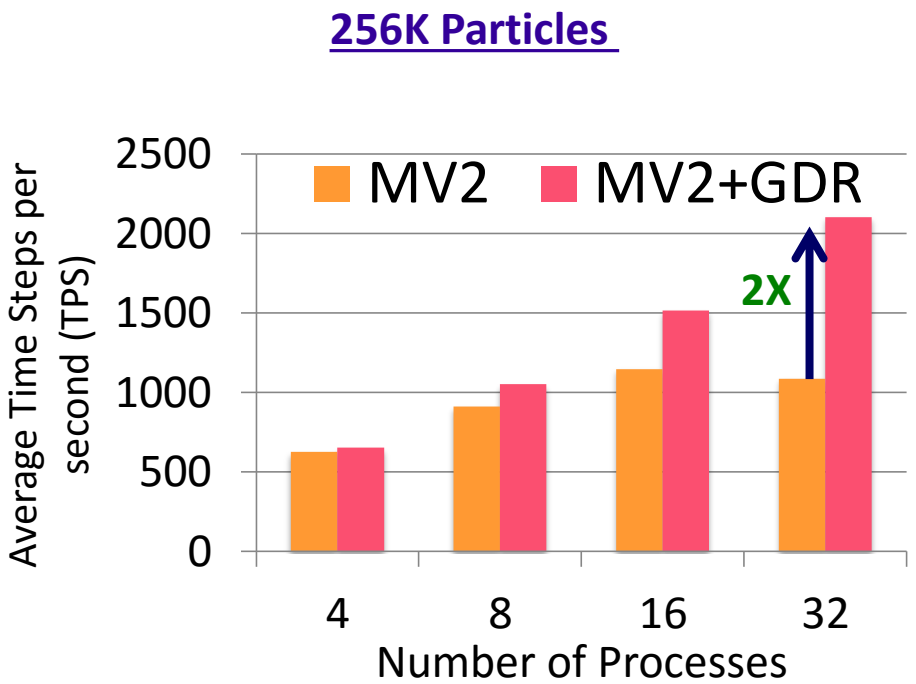




MVAPICH2-GDR-2.2
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 8.0
Mellanox OFED 3.0 with GPU-Direct-RDMA

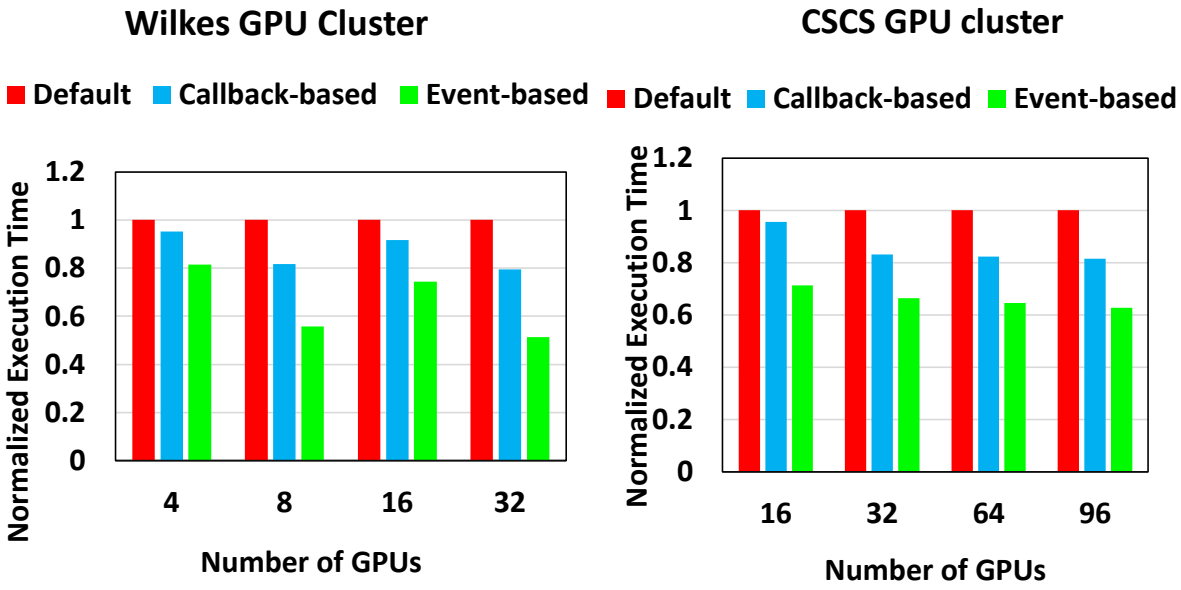
Application-Level Evaluation (HOOMD-blue)

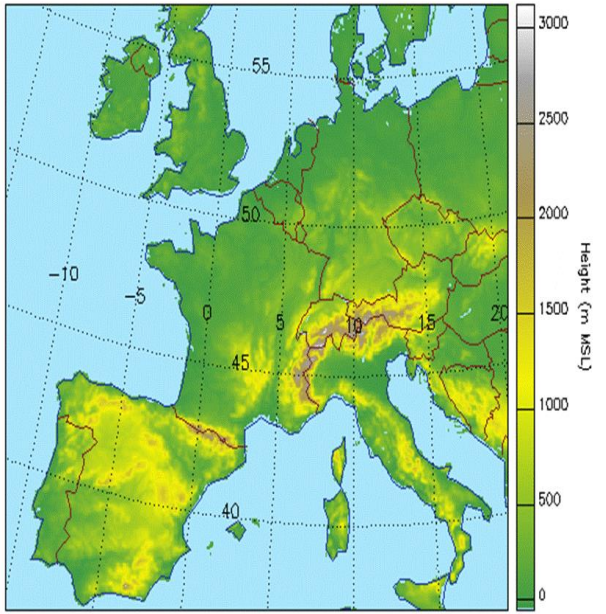




- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomdBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland





- 2X improvement on 32 GPUs nodes
- 30% improvement on 96 GPU nodes (8 GPUs/node)

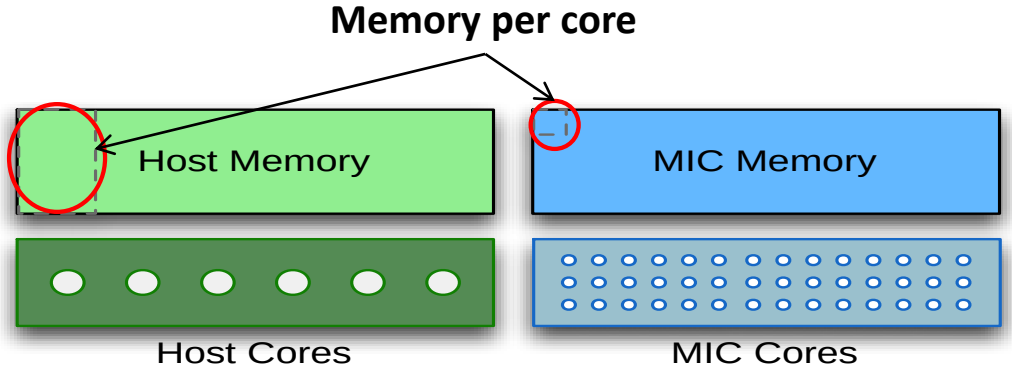
Cosmo model: http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/

On-going collaboration with CSCS and MeteoSwiss (Switzerland) in co-designing MV2-GDR and Cosmo Application

C. Chu, K. Hamidouche, A. Venkatesh, D. Banerjee , H. Subramoni, and D. K. Panda, Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-enabled Systems, IPDPS'16

Need for Non-Uniform Memory Allocation in OpenSHMEM for Heterogeneous Architectures

- MIC cores have limited memory per core
- OpenSHMEM relies on symmetric memory, allocated using shmalloc()

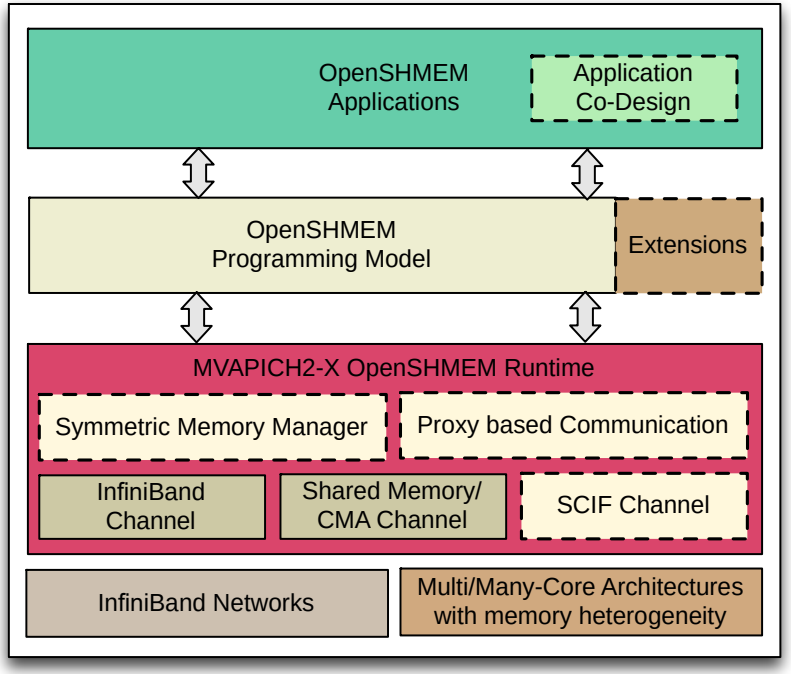


- shmalloc() allocates same amount of memory on all PEs
- For applications running in symmetric mode, this limits the total heap size
- Similar issues for applications (even host-only) with memory load imbalance (Graph500, Out-of-Core Sort, etc.)
- How to allocate different amounts of memory on host and MIC cores, and still be able to communicate?

OpenSHMEM Design for MIC Clusters

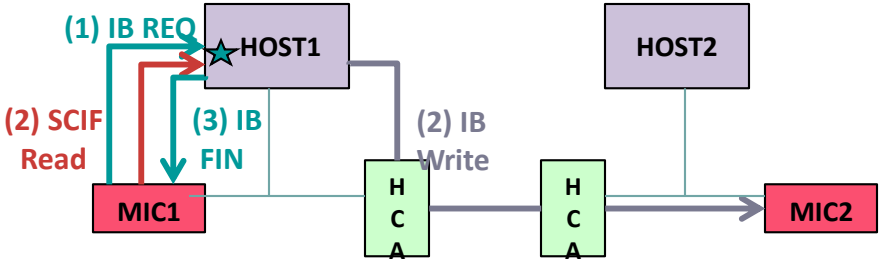
- Non-Uniform Memory Allocation:
  - Team-based Memory Allocation (Proposed Extensions)

```
void shmem_team_create(shmem_team_t team, int *ranks,
int size, shmem_team_t *newteam);
void shmem_team_destroy(shmem_team_t *team);

void shmem_team_split(shmem_team_t team, int color,
int key, shmem_team_t *newteam);

int shmem_team_rank(shmem_team_t team);
int shmem_team_size(shmem_team_t team);

void *shmalloc_team (shmem_team_t team, size_t size);
void shfree_team(shmem_team_t team, void *addr);
```

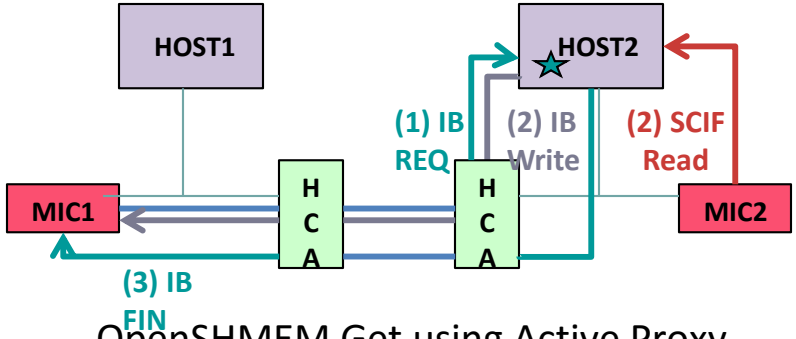


  - Address Structure for non-uniform memory allocations

Proxy-based Designs for OpenSHMEM



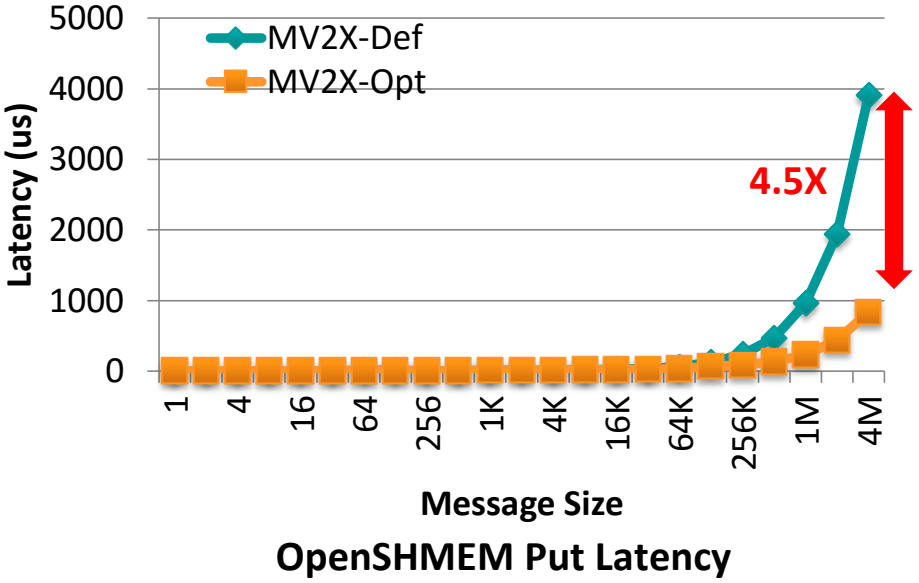
OpenSHMEM Put using Active Proxy



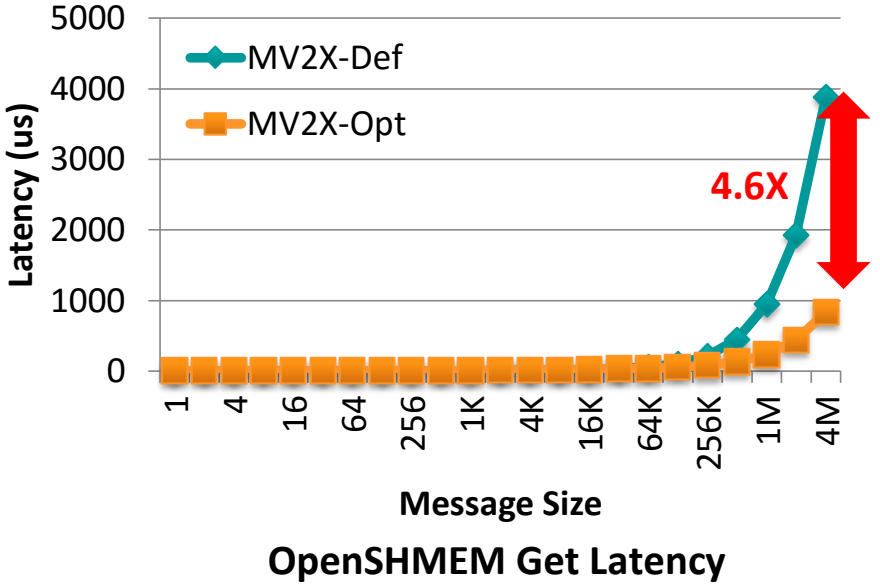
OpenSHMEM Get using Active Proxy

- Current generation architectures impose limitations on read bandwidth when HCA reads from MIC memory
  - Impacts both put and get operation performance
- Solution: Pipelined data transfer by proxy running on host using IB and SCIF channels
- Improves latency and bandwidth!

OpenSHMEM Put/Get Performance



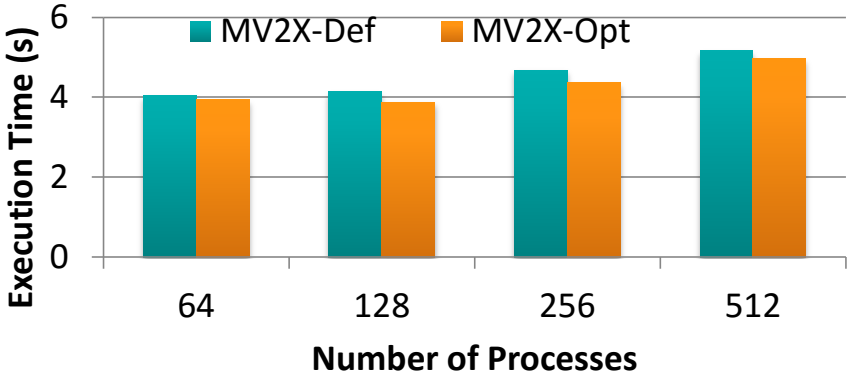
OpenSHMEM Put Latency



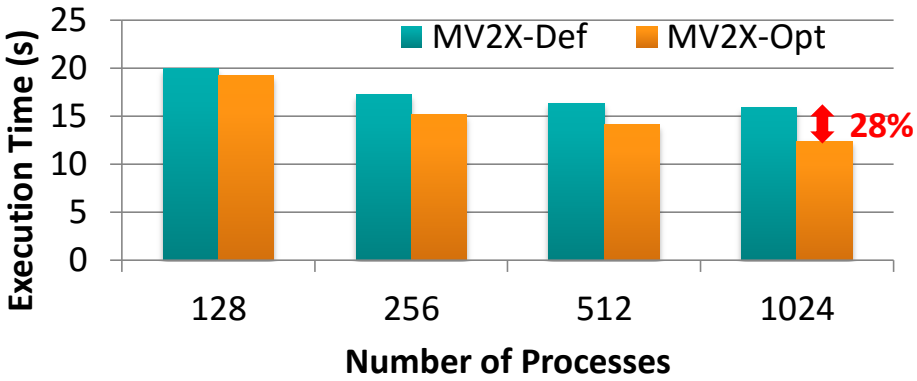
OpenSHMEM Get Latency

- Proxy-based designs alleviate hardware limitations
- Put Latency of 4M message: Default: 3911us, Optimized: 838us
- Get Latency of 4M message: Default: 3889us, Optimized: 837us

Performance Evaluations using Graph500



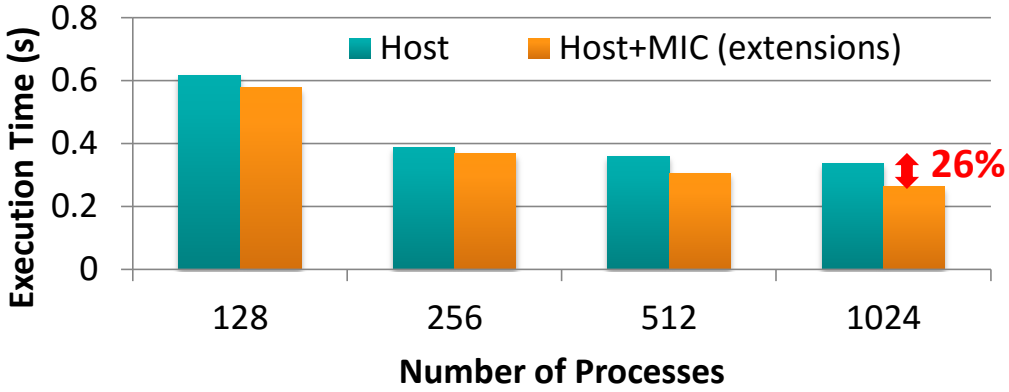
Native Mode (8 procs/MIC)



Symmetric Mode (16 Host+16MIC)

- Graph500 Execution Time (Native Mode):
  - **8 processes per MIC node**
  - At 512 processes , Default: 5.17s, Optimized: 4.96s
  - Performance Improvement from MIC-aware collectives design
- Graph500 Execution Time (**Symmetric** Mode):
  - **16 processes on each Host and MIC node**
  - At 1,024 processes, Default: 15.91s, Optimized: 12.41s
  - Performance Improvement from MIC-aware collectives and proxy-based designs

Graph500 Evaluations with Extensions



- Redesigned Graph500 using MIC to overlap computation/communication
  - Data Transfer to MIC memory; MIC cores pre-processes received data
  - Host processes traverses vertices, and sends out new vertices
- Graph500 Execution time at 1,024 processes:
  - **16 processes on each Host and MIC node**
  - Host-Only: .33s, Host+MIC with Extensions: .26s
- Magnitudes of improvement compared to default symmetric mode
  - Default Symmetric Mode: 12.1s, Host+MIC Extensions: 0.16s

J. Jose, K. Hamidouche, X. Lu, S. Potluri, J. Zhang, K. Tomko and D. K. Panda, High Performance OpenSHMEM for Intel MIC Clusters: Extensions, Runtime Designs and Application Co-Design, IEEE International Conference on Cluster Computing (CLUSTER '14) (Best Paper Finalist)

Looking into the Future ….

- Architectures for Exascale systems are evolving
- Exascale systems will be constrained by
  - Power
  - Memory per core
  - Data movement cost
  - Faults
- Programming Models, Runtimes and Middleware need to be designed for
  - Scalability
  - Performance
  - Fault-resilience
  - Energy-awareness
  - Programmability
  - Productivity
- High Performance and Scalable MPI+X libraries are needed
- Highlighted some of the approaches taken by the MVAPICH2 project
- Need continuous innovation to have the right MPI+X libraries for Exascale systems

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)
- J. Hashmi (Ph.D.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Current Research Scientists

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

Current Research Specialist

- J. Smith

Past Students

- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- 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.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- 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.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Research Scientist

- S. Sur

Past Programmers

- D. Bureddy
- M. Arnold
- J. Perkins

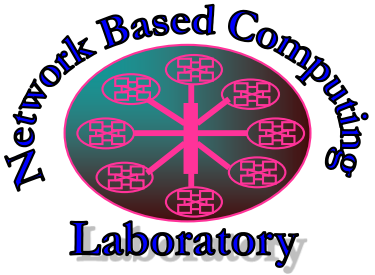
Past Post-Docs

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

OSU Team Will be Participating in Multiple Events at SC '16

- Three Conference Tutorials (IB+HSE, IB+HSE Advanced, Big Data)
- HP-CAST
- Technical Papers (SC main conference; Doctoral Showcase; Poster; PDSW-DISC, PAW, COMHPC, and ESPM2 Workshops)
- Booth Presentations (Mellanox, NVIDIA, NRL, PGAS)
- HPC Connection Workshop
- Will be stationed at Ohio Supercomputer Center/OH-TECH Booth (#1107)
  - Multiple presentations and demos
- More Details from http://mvapich.cse.ohio-state.edu/talks/

Thank You!



Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH Project
http://mvapich.cse.ohio-state.edu/

panda@cse.ohio-state.edu, hamidouch@cse.ohio-state.edu