MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Designing OpenSHMEM and Hybrid MPI+OpenSHMEM Libraries for Exascale Systems: MVAPICH2-X Experience

Talk at OpenSHMEM Workshop (August 2016)

by

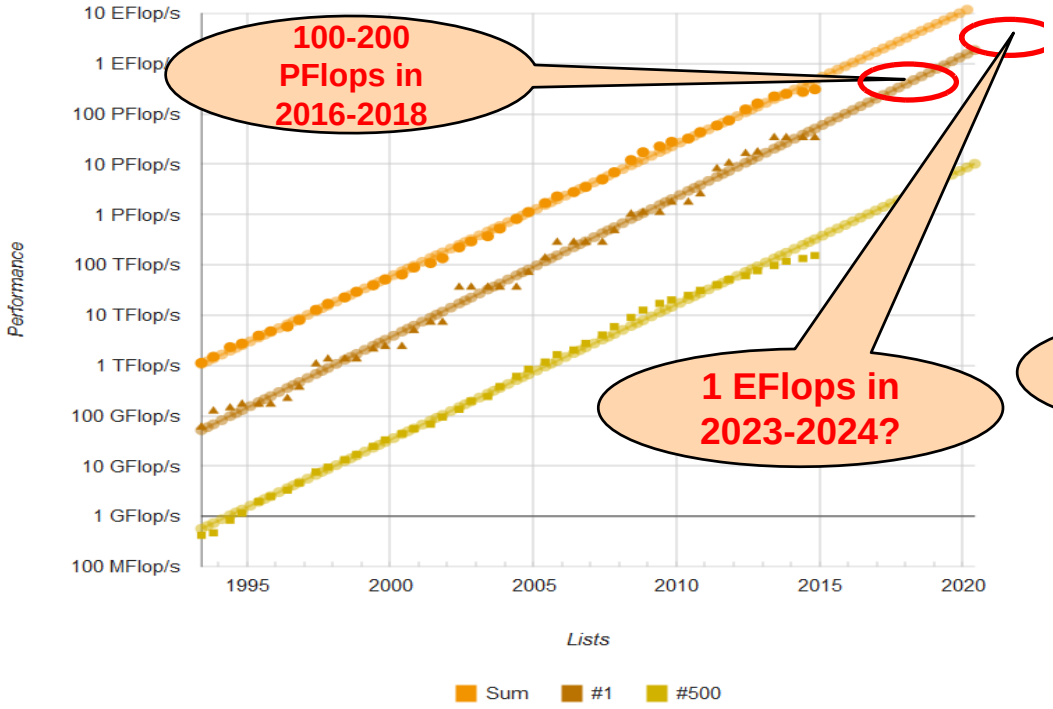
Dhabaleswar K. (DK) Panda

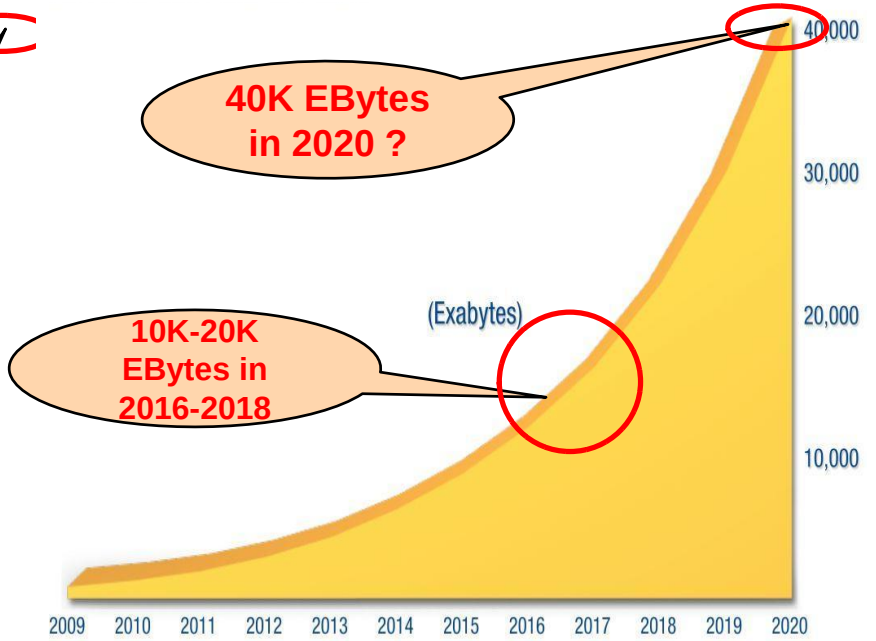
The Ohio State University

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

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

High-End Computing (HEC): ExaFlop & ExaByte



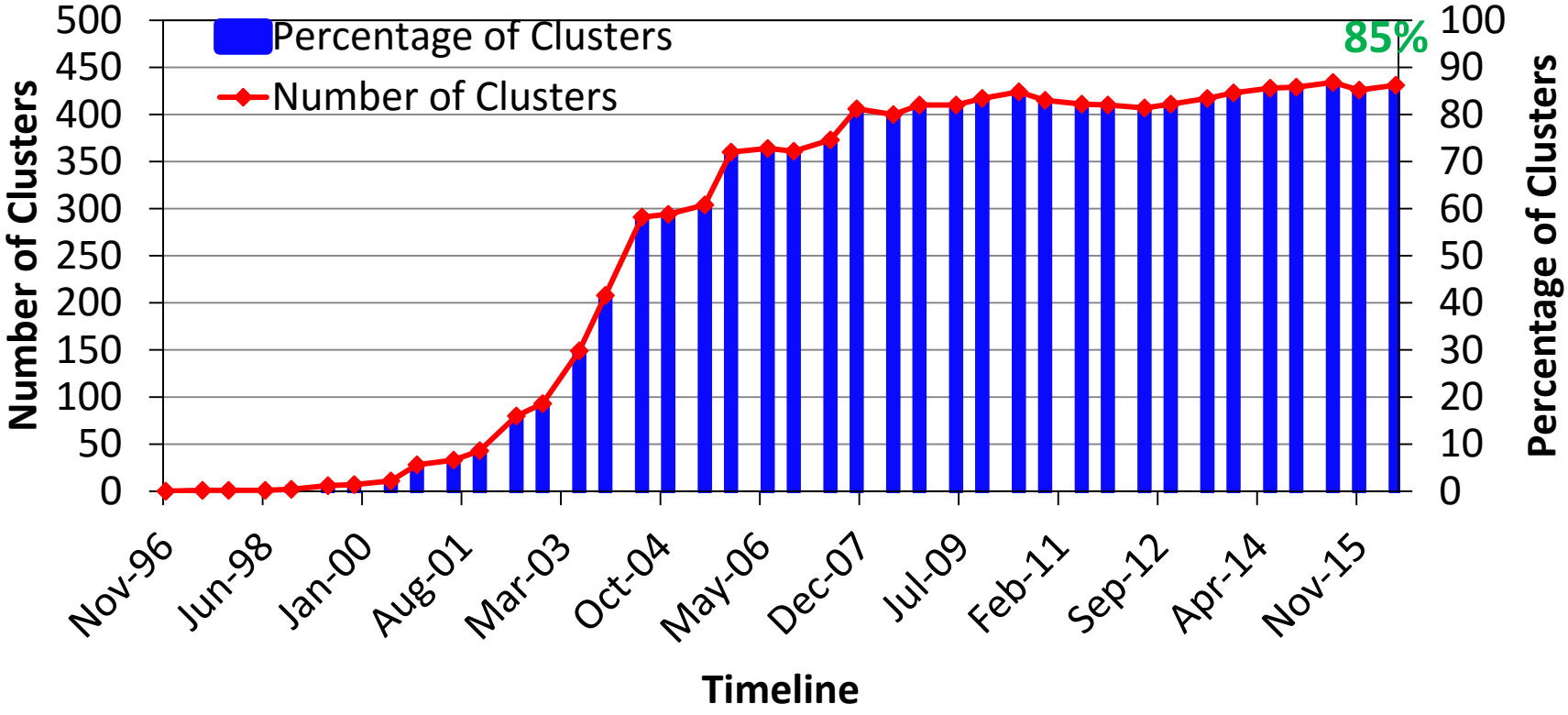


Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

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)

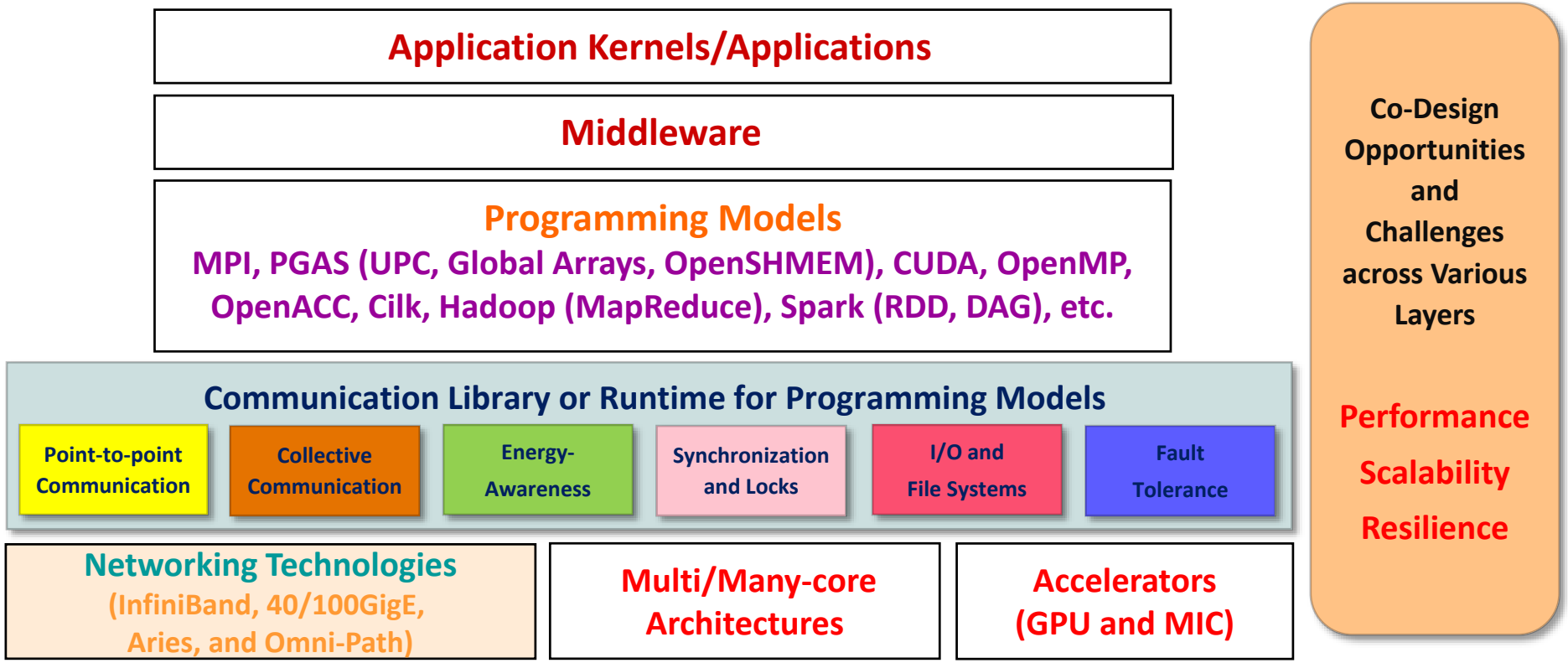
Tianhe – 2

Titan

Stampede

Tianhe – 1A

Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges



Broad Challenges in Designing Communication Libraries for (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)
  - 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
- Integrated Support for GPGPUs and Accelerators
- 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

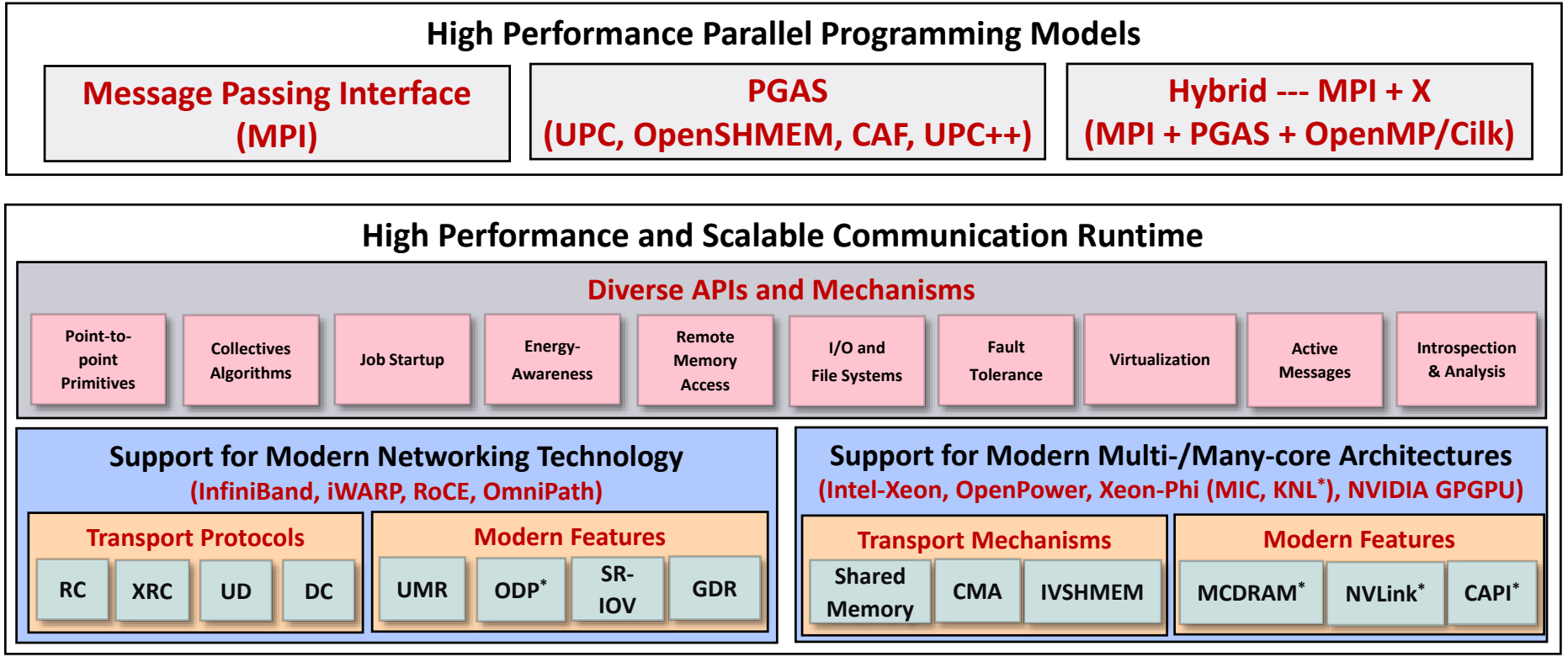
- **Extreme Low Memory Footprint**
  - Memory per core continues to decrease

- **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), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - 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 2,625 organizations in 81 countries**
  - **More than 382,000 (> 0.38 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Jun '16 ranking)
    - 12th ranked 519,640-core cluster (Stampede) at TACC
    - 15th ranked 185,344-core cluster (Pleiades) at NASA
    - 31st 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) ->
  - Stampede at TACC (12th in Jun'16, 462,462 cores, 5.168 Plops)

MVAPICH2 Overall Architecture



* Upcoming

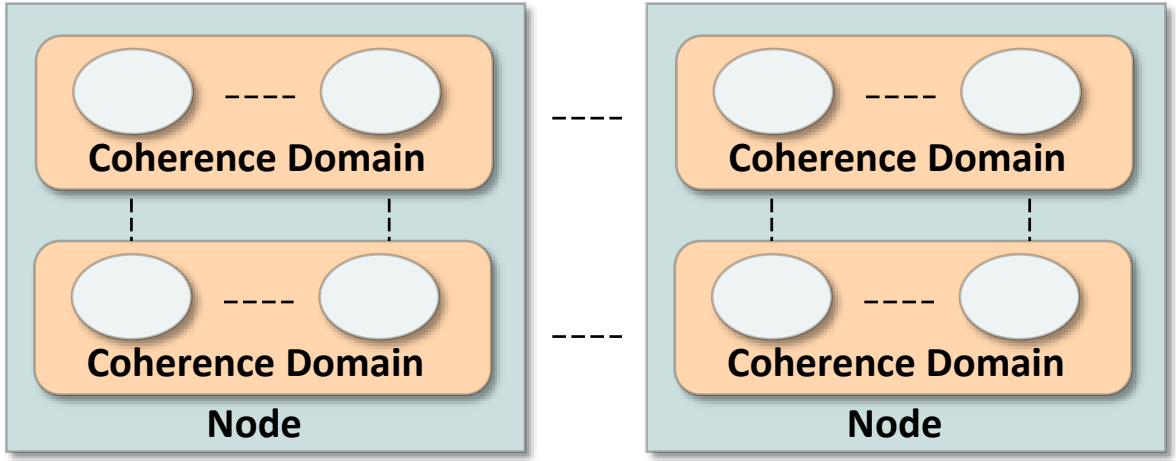
MVAPICH2 Software Family

Requirements	MVAPICH2 Library to use
MPI with IB, iWARP and RoCE	MVAPICH2
Advanced MPI, OSU INAM, PGAS and MPI+PGAS with IB and RoCE	MVAPICH2-X
MPI with IB & GPU	MVAPICH2-GDR
MPI with IB & MIC	MVAPICH2-MIC
HPC Cloud with MPI & IB	MVAPICH2-Virt
Energy-aware MPI with IB, iWARP and RoCE	MVAPICH2-EA

Outline

- Overview of MVAPICH2-X Architecture
  - Unified Runtime for Hybrid MPI+PGAS programming
  - OpenSHMEM Support
  - Other PGAS support (UPC, CAF and UPC++)
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
- Integrated Support for MICs

Architectures for Exascale Systems



Hypothetical Future Architecture*

- Modern architectures have increasing number of cores per node, but have limited memory per core
  - Memory bandwidth per core decreases
  - Network bandwidth per core decreases
  - Deeper memory hierarchy
  - More parallelism within the node

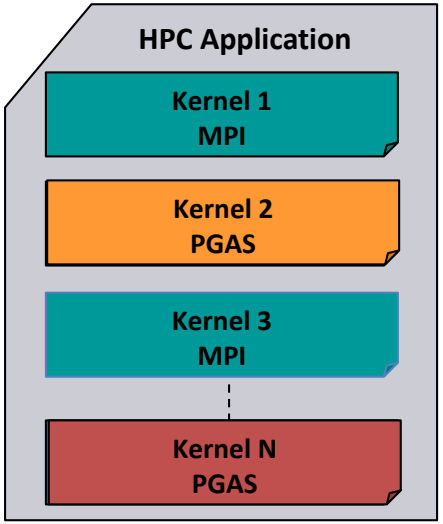
**Marc Snir, Keynote Talk – Programming Models for High Performance Computing, Cluster, Cloud and Grid Computing (CCGrid 2013)*

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

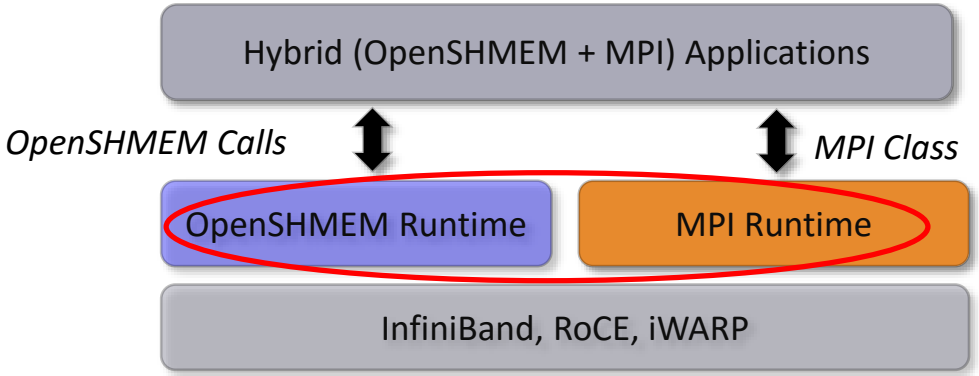
Hybrid (MPI+PGAS) Programming

- 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



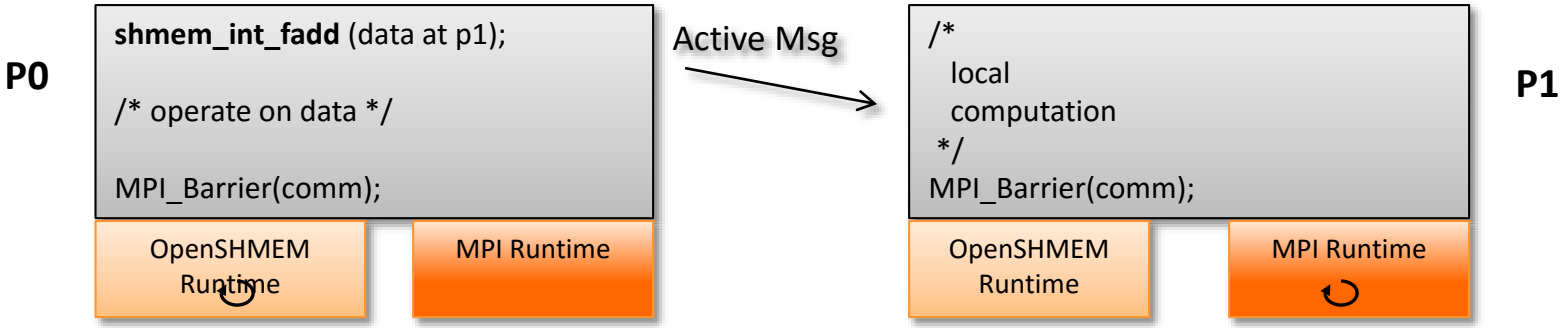
Current Approaches for Hybrid Programming

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



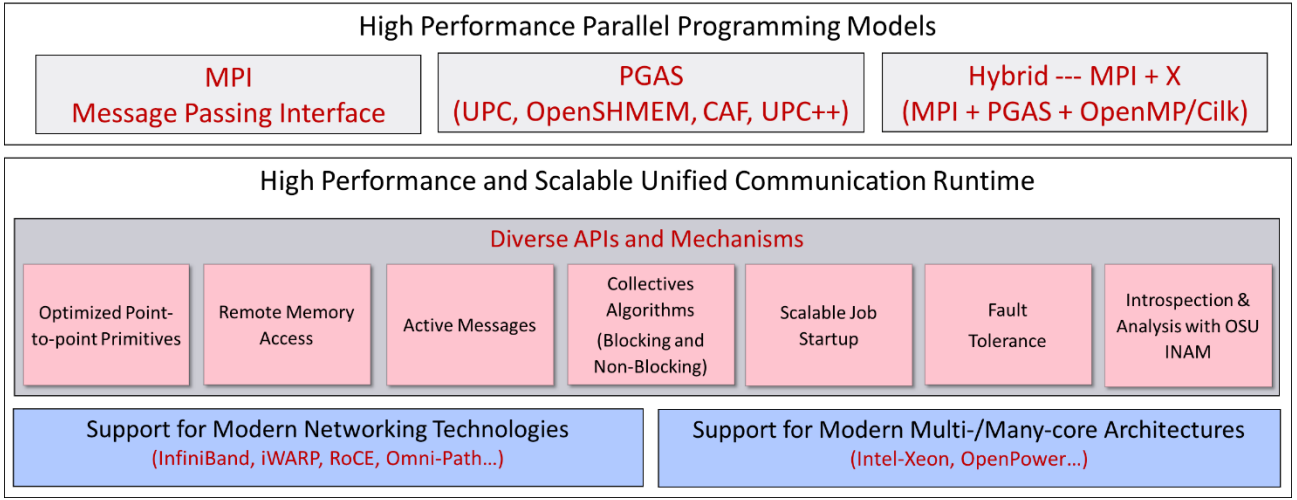
- Need more network and memory resources
- Might lead to deadlock!

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



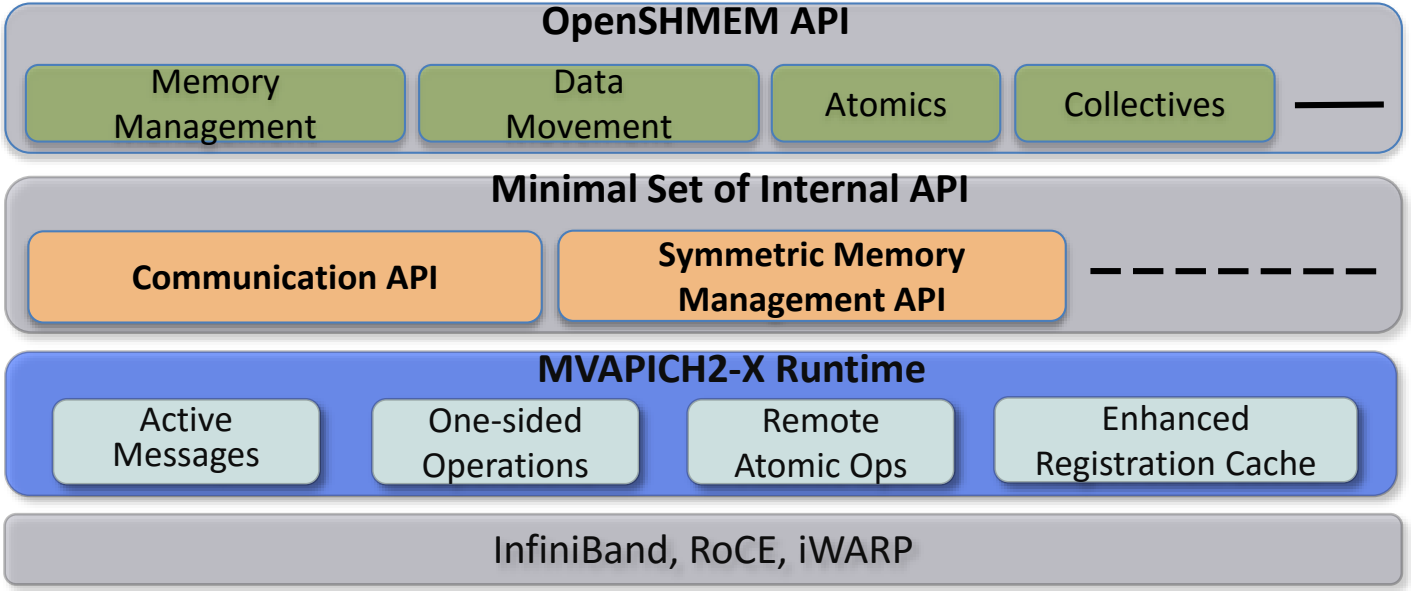
- Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF
  - Available since 2012 (starting with MVAPICH2-X 1.9)
  - http://mvapich.cse.ohio-state.edu

- 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

Outline

- Overview of MVAPICH2-X Architecture
  - Unified Runtime for Hybrid MPI+PGAS programming
  - OpenSHMEM Support
  - Other PGAS support (UPC, CAF and UPC++)
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
- Integrated Support for MICs

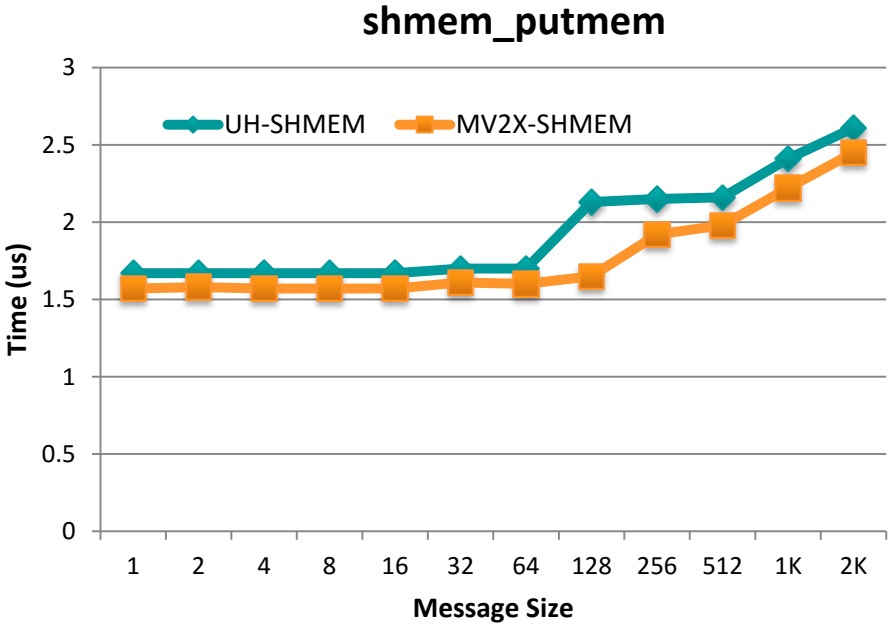
OpenSHMEM Design in MVAPICH2-X

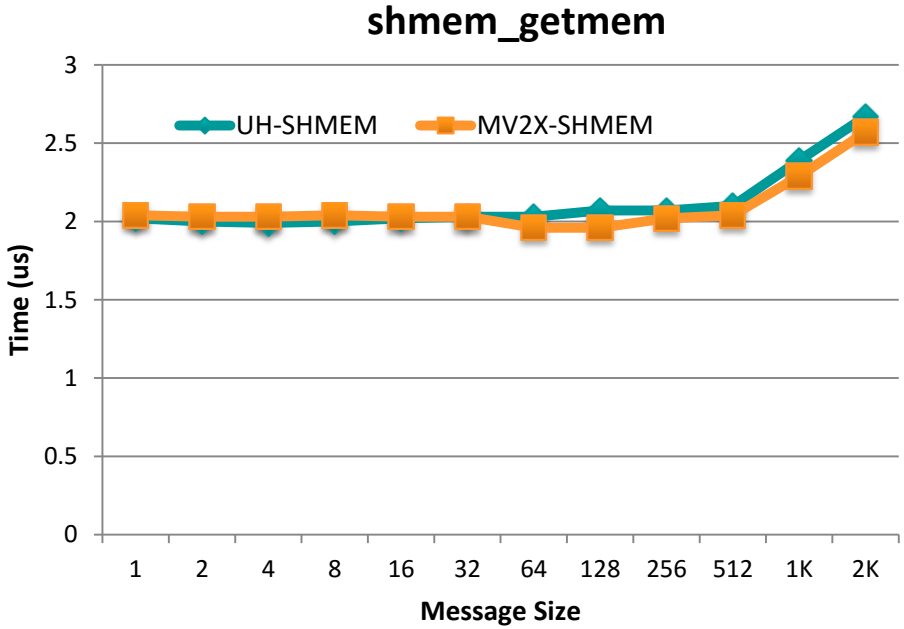


- OpenSHMEM Stack based on OpenSHMEM Reference Implementation
- OpenSHMEM Communication over MVAPICH2-X Runtime
  - Uses active messages, atomic and one-sided operations and remote registration cache

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

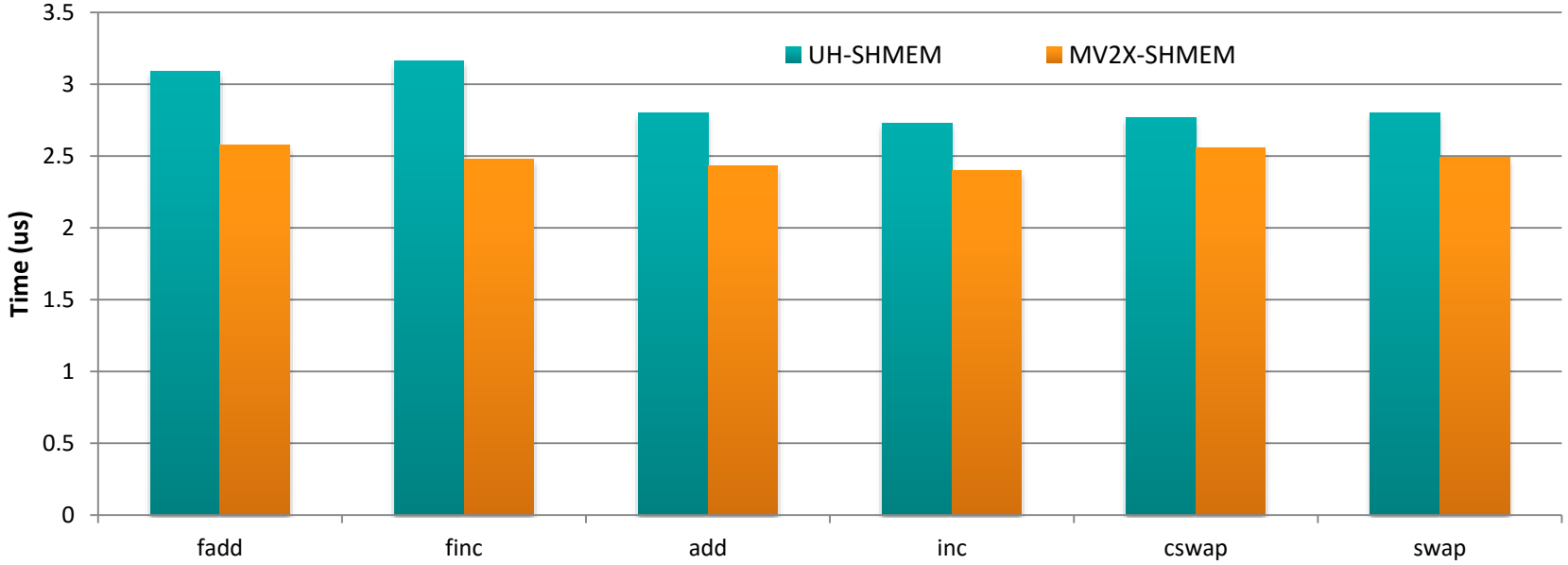
OpenSHMEM Data Movement: Performance





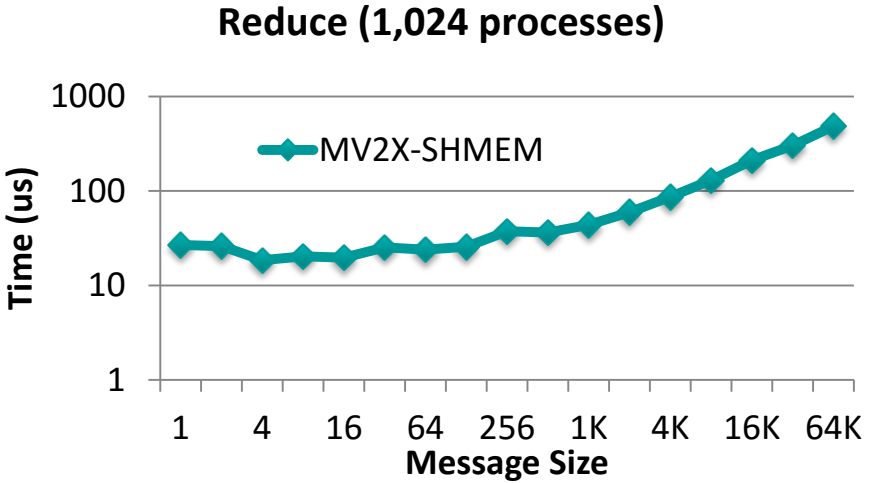
- OSU OpenSHMEM micro-benchmarks
  - http://mvapich.cse.ohio-state.edu/benchmarks/
- Slightly better performance for putmem and getmem with MVAPICH2-X
- MVAPICH2-X 2.2 RC1, Broadwell CPU, InfiniBand EDR Interconnect

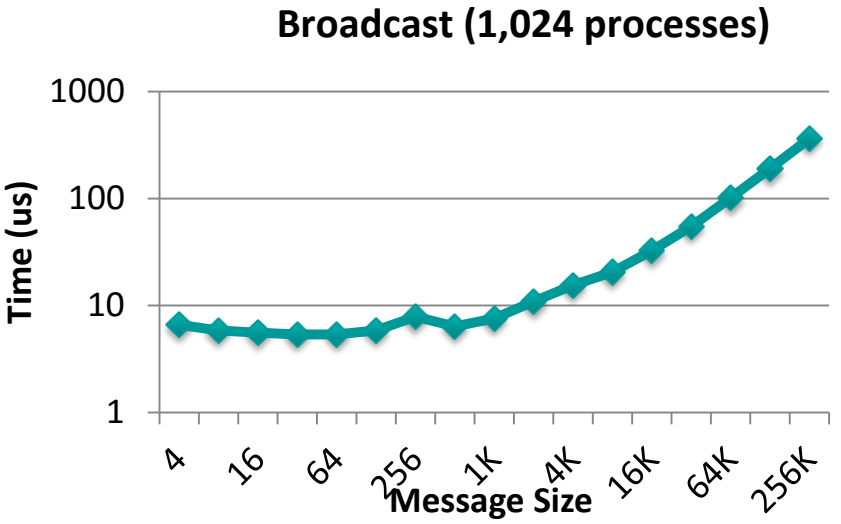
OpenSHMEM Atomic Operations: Performance

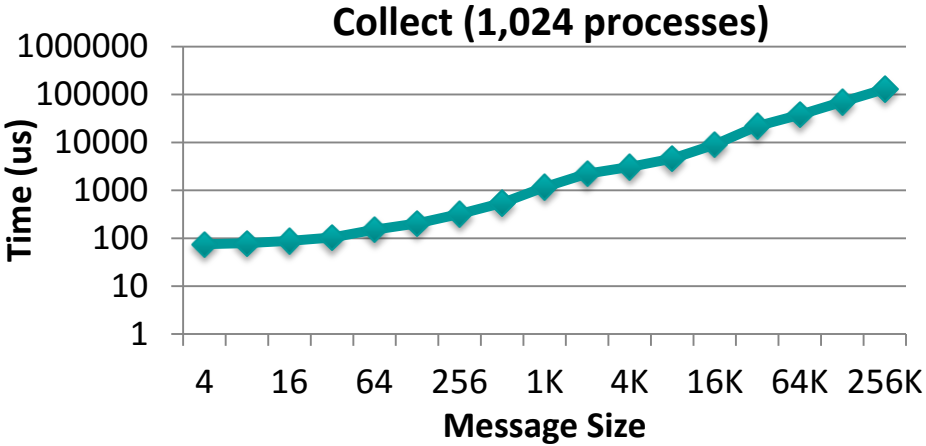


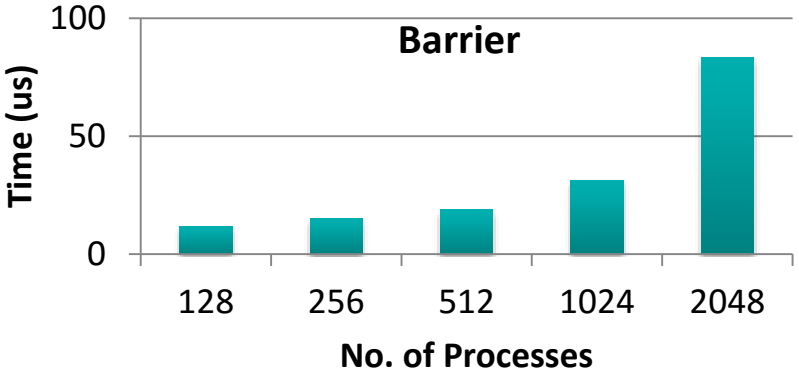
- OSU OpenSHMEM micro-benchmarks (OMB v5.3)
- MV2-X SHMEM performs up to **22%** better compared to UH-SHMEM

Collective Communication: Performance On Stampede

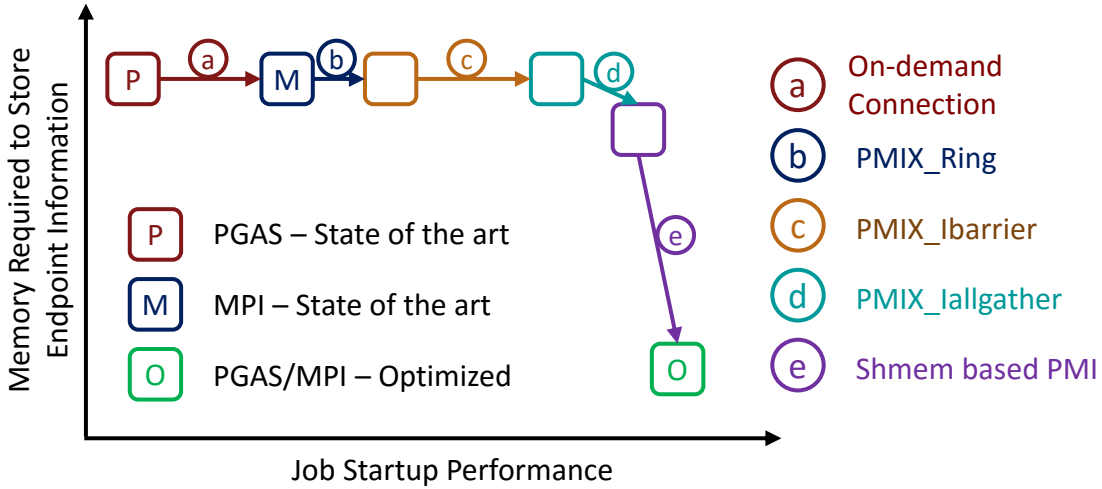








Towards High Performance and Scalable OpenSHMEM Startup at Exascale



- Near-constant MPI and OpenSHMEM initialization time at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM respectively at 16,384 processes
- Memory consumption reduced for remote endpoint information by O(processes per node)
- 1GB Memory saved per node with 1M processes and 16 processes per node

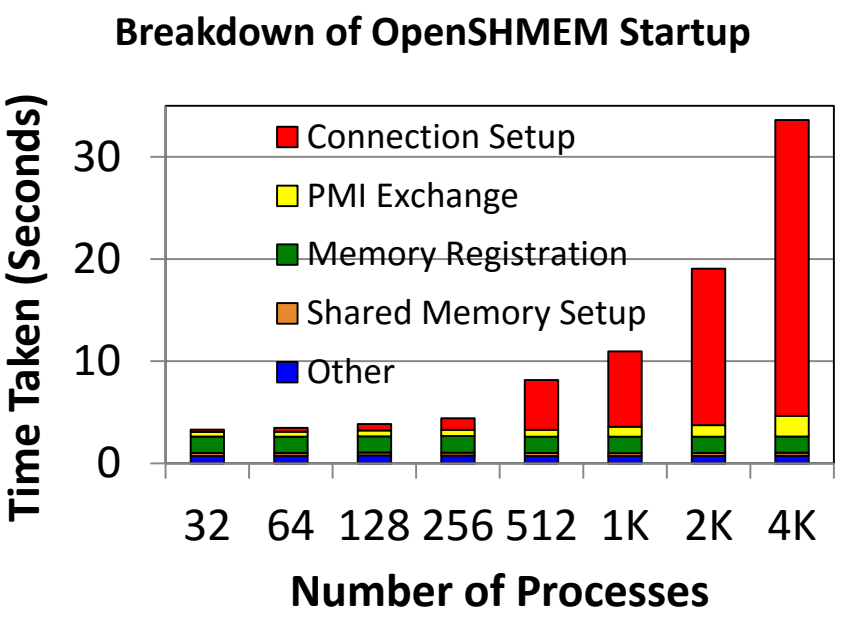
(a) **On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI.** S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS '15)

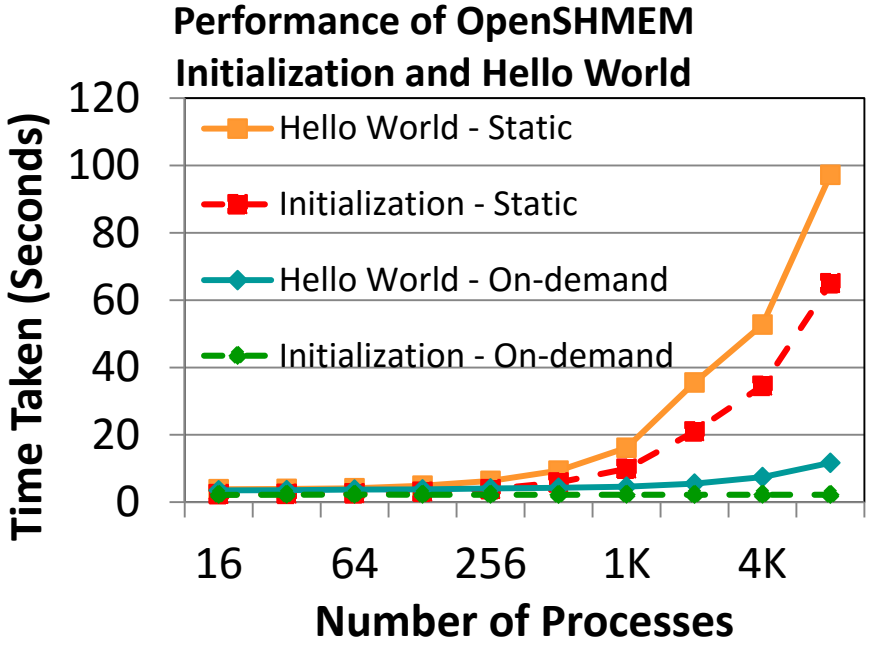
(b) **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia '14)

(c) (d) **Non-blocking PMI Extensions for Fast MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '15)

(e) **SHMEMPMI – Shared Memory based PMI for Improved Performance and Scalability.** S. Chakraborty, H. Subramoni, J. Perkins, and D K Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '16) , *Accepted for Publication*

On-demand Connection Management for OpenSHMEM+MPI

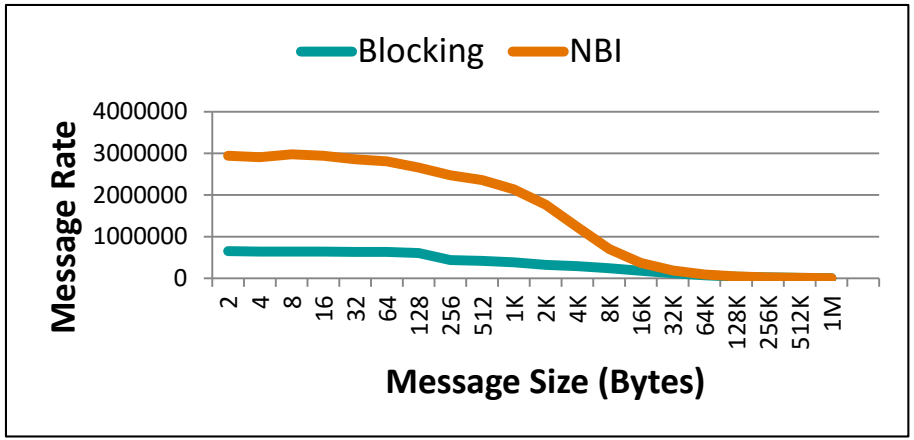




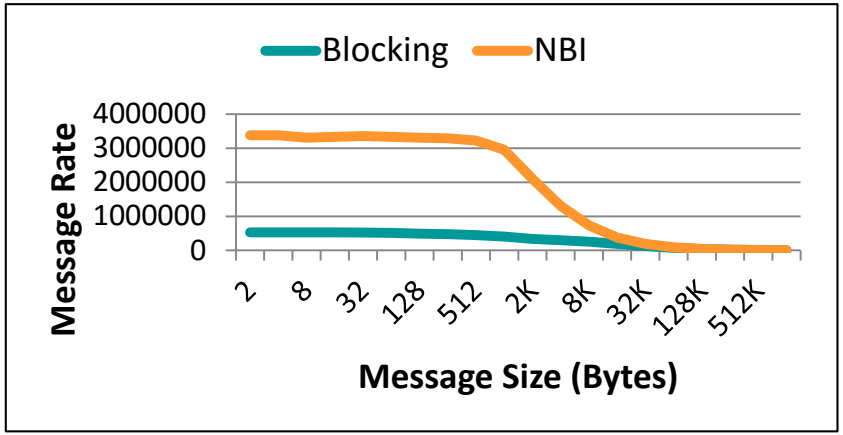
- Static connection establishment wastes memory and takes a lot of time
- On-demand connection management improves OpenSHMEM initialization time by 29.6 times
- Time taken for Hello World reduced by 8.31 times at 8,192 processes
- **Available since MVAPICH2-X 2.1rc1**

OpenSHMEM 1.3 Support: NBI Operations

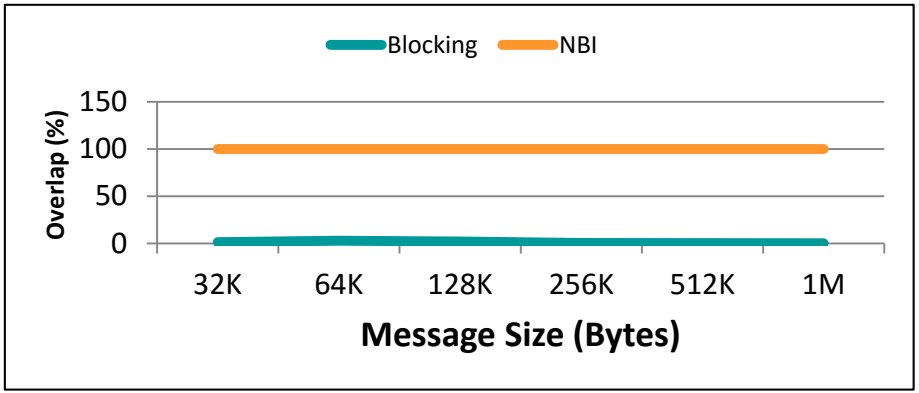
Inter-node Put



Inter-node Get



Inter-node Get

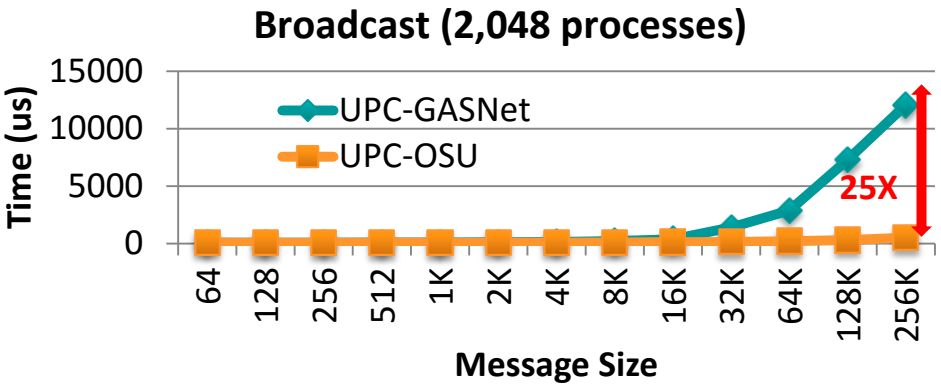


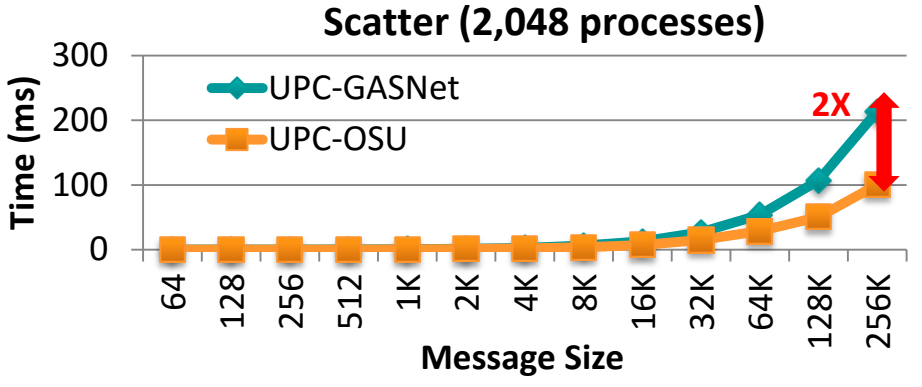
- Higher message rate
- Perfect overlap of computation/Communication
- Extension of OMB with NBI benchmarks
- Will be available with next releases of MVAPICH2-X

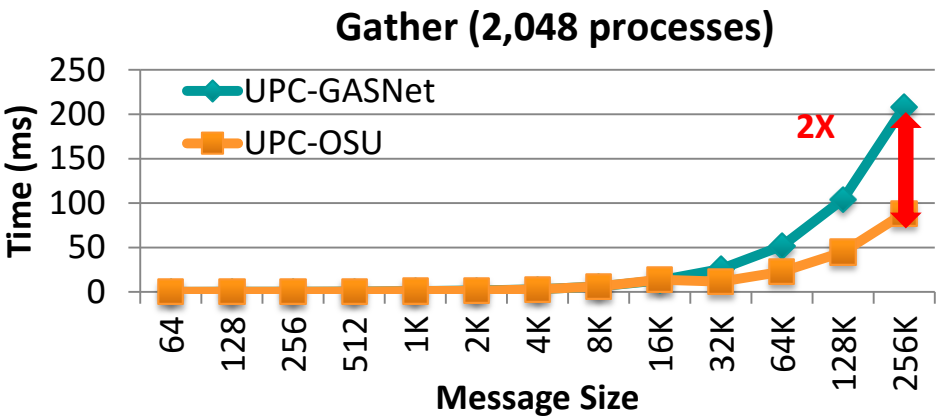
Outline

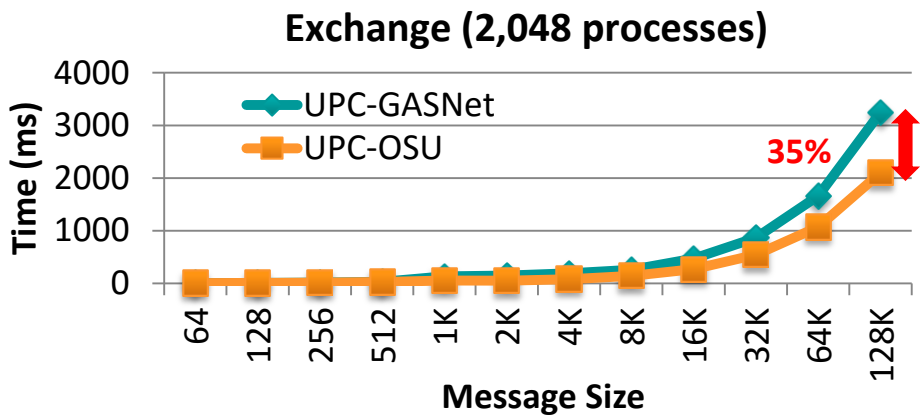
- Overview of MVAPICH2-X Architecture
  - Unified Runtime for Hybrid MPI+PGAS programming
  - OpenSHMEM Support
  - Other PGAS support (UPC, CAF and UPC++)
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
- Integrated Support for MICs

UPC Collectives Performance



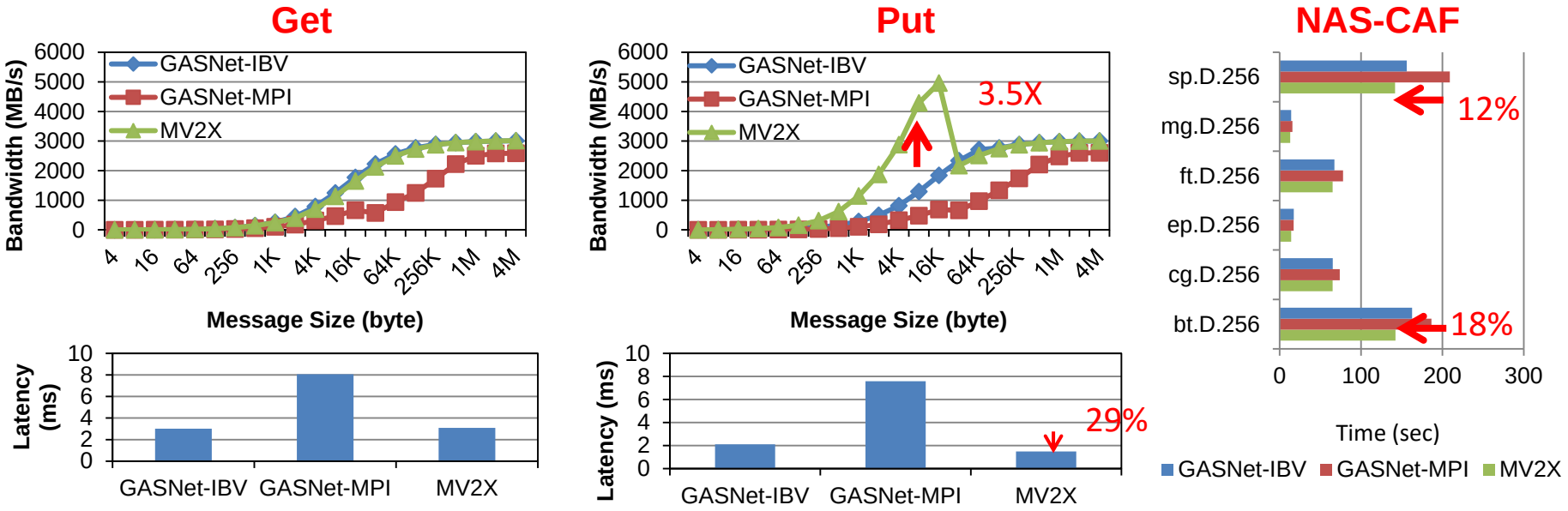






J. Jose, K. Hamidouche, J. Zhang, A. Venkatesh, and D. K. Panda, Optimizing Collective Communication in UPC (HiPS'14, in association with IPDPS'14)

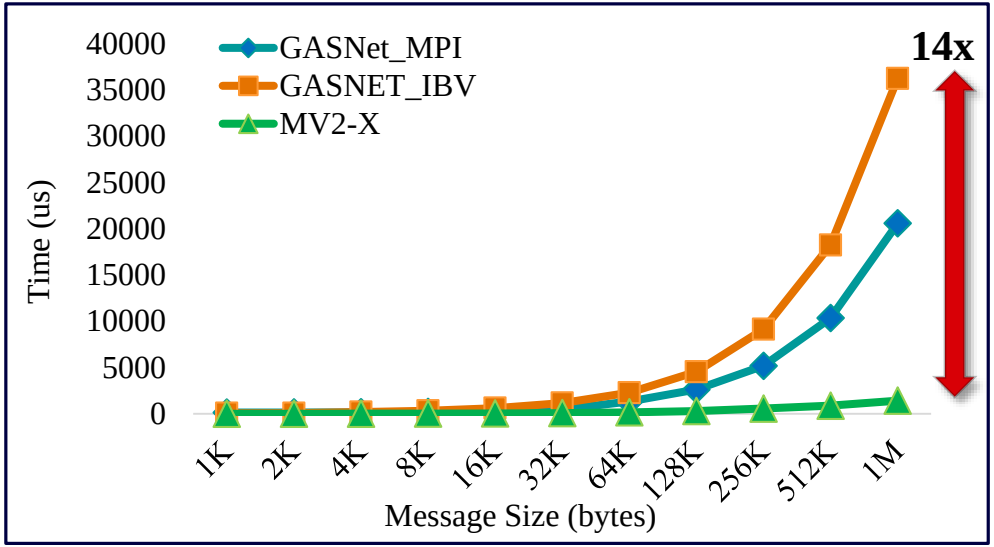
Performance Evaluations for CAF model



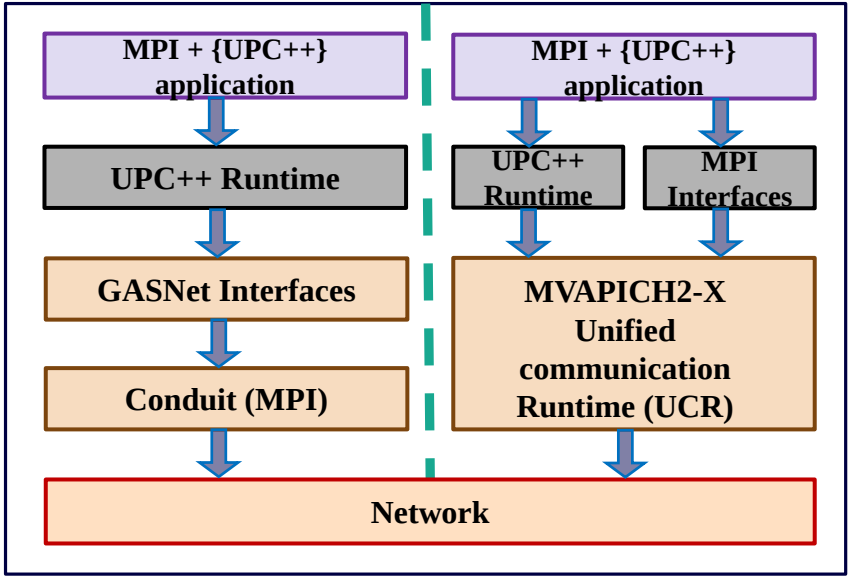
- Micro-benchmark improvement (MV2X vs. GASNet-IBV, UH CAF test-suite)
  - Put bandwidth: 3.5X improvement on 4KB; Put latency: reduce 29% on 4B
- Application performance improvement (NAS-CAF one-sided implementation)
  - Reduce the execution time by 12% (SP.D.256), 18% (BT.D.256)

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 with the latest release of MVAPICH2-X 2.2RC1

Outline

- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
  - Graph500
  - Out-of-Core Sort
  - MiniMD
  - MaTEx
- Integrated Support for GPGPUs
- Integrated Support for MICs

Incremental Approach to Exploit One-sided Operations

- Identify the communication critical section (mpiP, HPCToolkit)
- Allocate memory in shared address space
- Convert MPI Send/Recvs to assignment operations or one-sided operations
  - Non-blocking operations can be utilized
  - Coalescing for reducing the network operations
- Introduce synchronization operations for data consistency
  - After Put operations or before get operations
- Load balance through global view of data

Graph500 Benchmark – The Algorithm

- Breadth First Search (BFS) Traversal
- Uses ‘Level Synchronized BFS Traversal Algorithm
  - Each process maintains – ‘*CurrQueue*’ and ‘*NewQueue*’
  - Vertices in *CurrQueue* are traversed and newly discovered vertices are sent to their owner processes
  - Owner process receives edge information
    - If not visited; updates parent information and adds to *NewQueue*
  - Queues are swapped at end of each level
  - Initially the ‘root’ vertex is added to *currQueue*
  - Terminates when queues are empty

Hybrid Graph500 Design

- Communication and co-ordination using one-sided routines and fetch-add atomic operations
  - Every process keeps receive buffer
  - Synchronization using atomic fetch-add routines
- Level synchronization using non-blocking barrier
  - Enables more computation/communication overlap
- Load Balancing utilizing OpenSHMEM shmem_ptr
  - Adjacent processes can share work by reading shared memory

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

Pseudo Code For Both MPI and Hybrid Versions

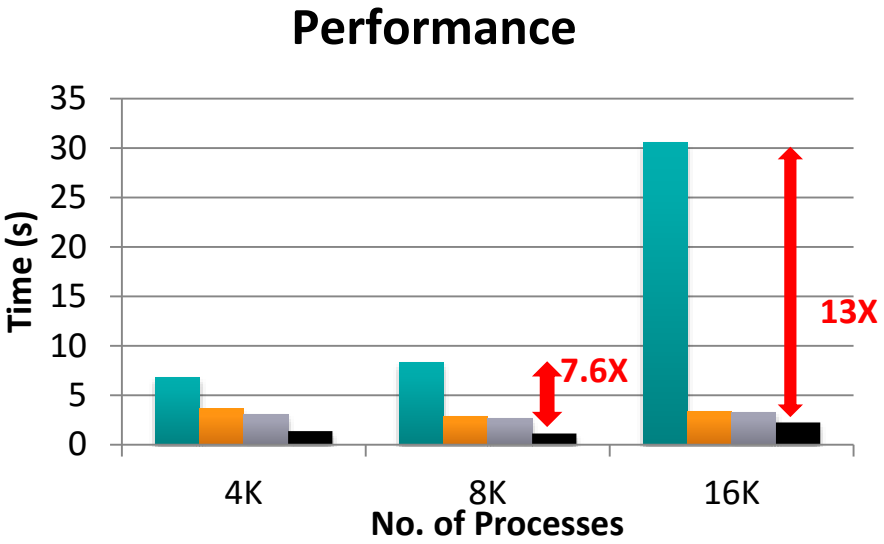
Algorithm 1: EXISTING MPI SEND/RECV

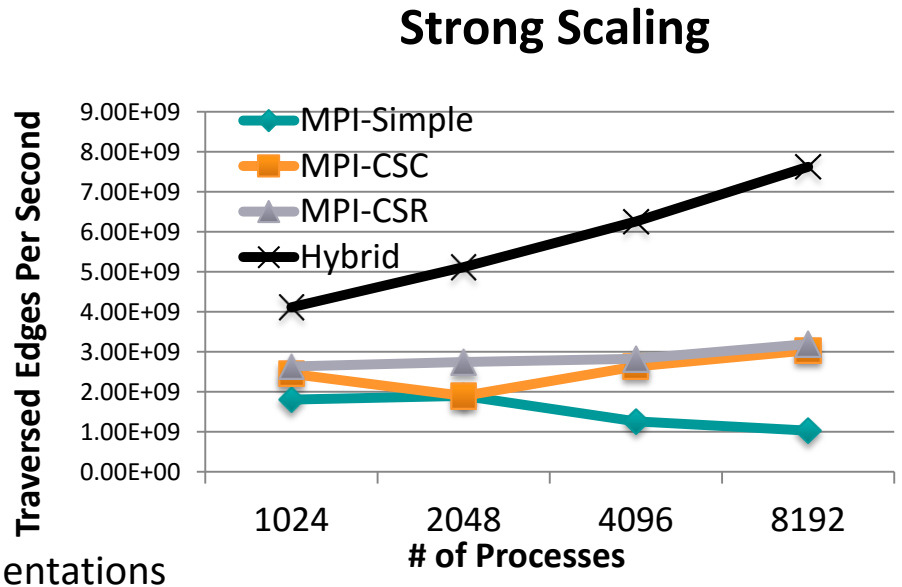
```
while true do
        while CurrQueue != NULL do
                for vertex u in CurrQueue do
                 HandleReceive()
                u ← Dequeue(CurrQueue)
                 Send(u, v) to owner
    end
  Send empty messages to all others
  while all_done != N – 1 do
     HandleReceive()
  end
       // Procedure: HandleReceive
 if rcv_count = 0 then
    all_ done ← all_done + 1
else
    update (NewQueue, v)
```

Algorithm 2: HYBRID VERSION

```
 while true do
  while CurrQueue 6= NULL do
    for vertex u in CurrQueue do
    u ← Dequeue(CurrQueue)
      to the adjacent points to u do
     Shmem_fadd(owner, size,recv_index)
      shmem_put(owner, size,recv_buf)
     end
    end
  end
   if recv_buf[size] = done then
    Set ← 1
end
```

Graph500 - BFS Traversal Time





- Hybrid design performs better than MPI implementations
- 16,384 processes
  - 1.5X improvement over MPI-CSR
  - 13X improvement over MPI-Simple (Same communication characteristics)
- Strong Scaling

  Graph500 Problem Scale = 29

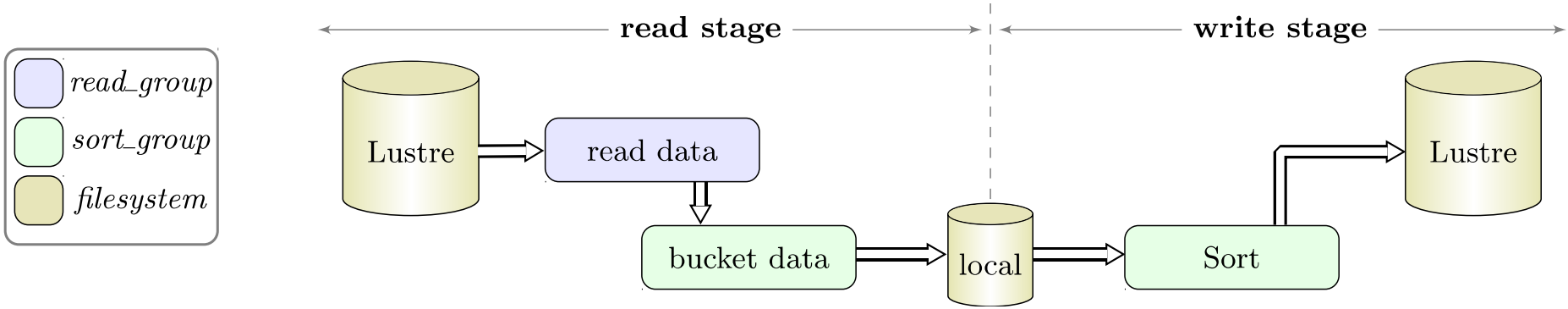
J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC '13), June 2013

Outline

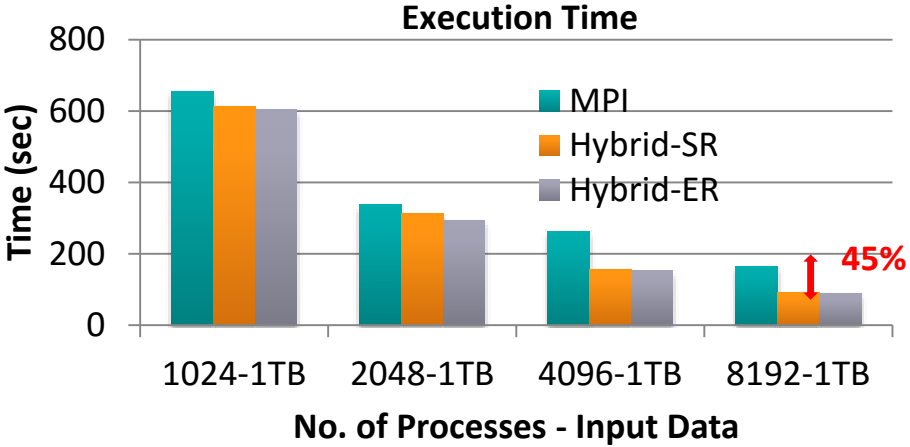
- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
  - Graph500
  - Out-of-Core Sort
  - MiniMD
  - MaTEx
- Integrated Support for GPGPUs
- Integrated Support for MICs

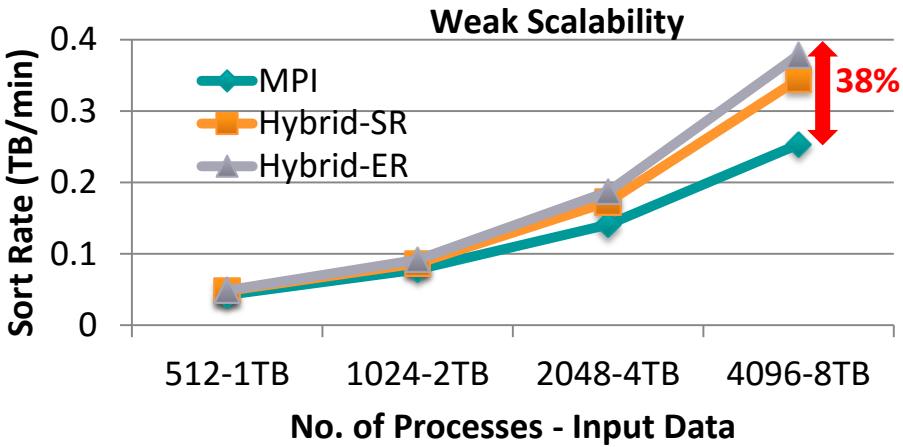
Out-of-Core Sorting

- Sorting: One of the most common algorithms in data analytics
- Sort Benchmark (sortbenchmark.org) ranks various frameworks available for large scale data analytics
- Read data from a global filesystem, sort it and write back to global filesystem



Hybrid MPI+OpenSHMEM Sort Application





- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - Execution Time (seconds)
    - 1TB Input size at 8,192 cores: MPI – 164, Hybrid-SR (Simple Read) – 92.5, Hybrid-ER (Eager Read) - 90.36
    - **45%** improvement over MPI-based design
  - Weak Scalability (configuration: input size of 1TB per 512 cores)
    - **At 4,096 cores:** MPI – 0.25 TB/min, Hybrid-SR – 0.34 TB/min, **Hybrid-SR – 0.38 TB/min**
    - **38%** improvement over MPI based 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

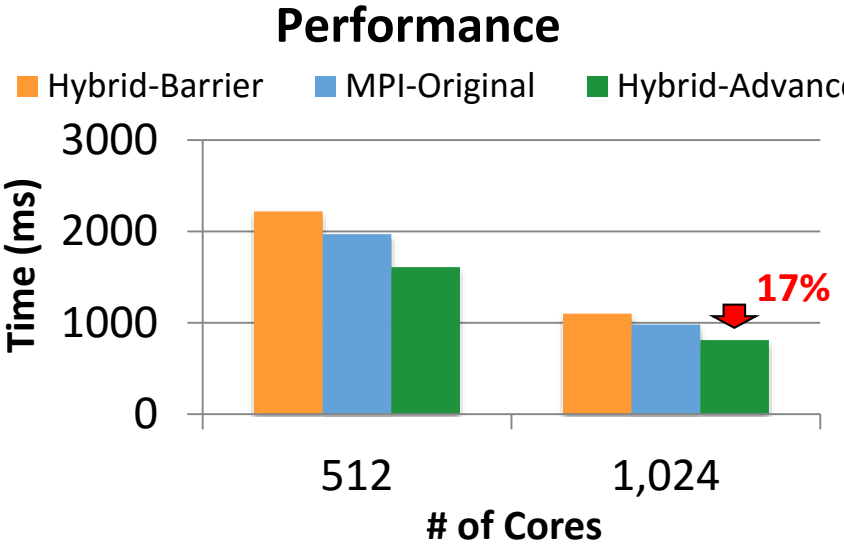
Outline

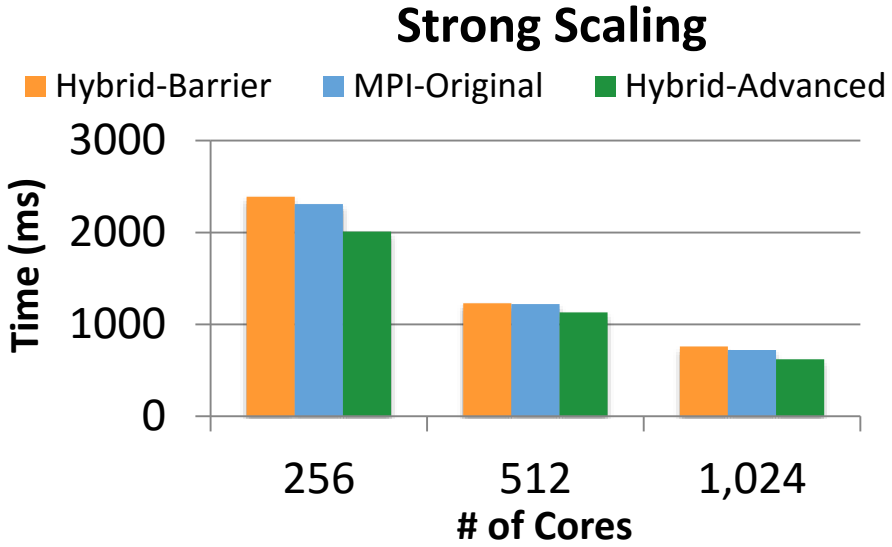
- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
  - Graph500
  - Out-of-Core Sort
  - MiniMD
  - MaTEx
- Integrated Support for GPGPUs
- Integrated Support for MICs

Overview of MiniMD

- MiniMD is a Molecular Dynamics (MD) mini-application in the Mantevo project at Sandia National Laboratories
- It has a stencil communication pattern which employs point-to-point message passing with irregular data
- Primary work loop inside MiniMD
  - Migrate the atoms to different ranks in every 20th iteration
  - Exchange position information of atoms in boundary regions
  - Compute forces based on local atoms and those in boundary region from neighboring ranks
  - Exchange force information of atoms in boundary regions
  - Update velocities and positions of local atoms

MiniMD – Total Execution Time





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

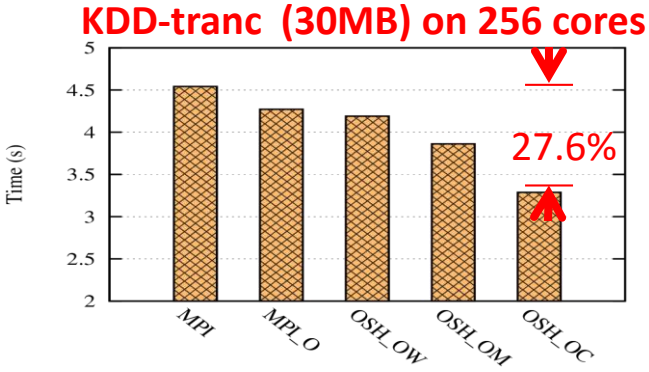
  Input size: 128 * 128 * 128

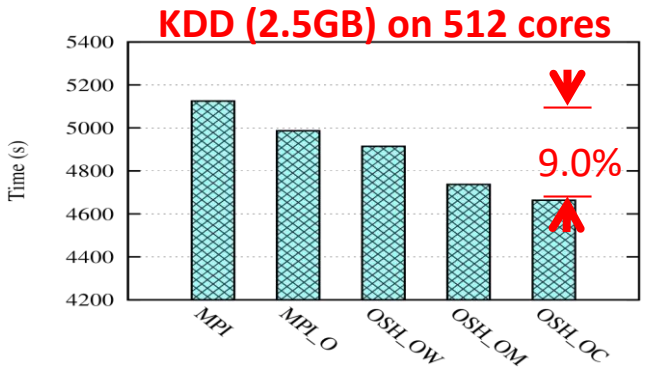
Outline

- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
  - Graph500
  - Out-of-Core Sort
  - MiniMD
  - MaTEx
- Integrated Support for GPGPUs
- Integrated Support for MICs

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

- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
  - Overview of CUDA-Aware Concepts
  - Designing Efficient MPI Runtime for GPU Clusters
  - Designing Efficient OpenSHMEM Runtime for GPU Clusters
- Integrated Support for MICs

MPI + CUDA - Naive

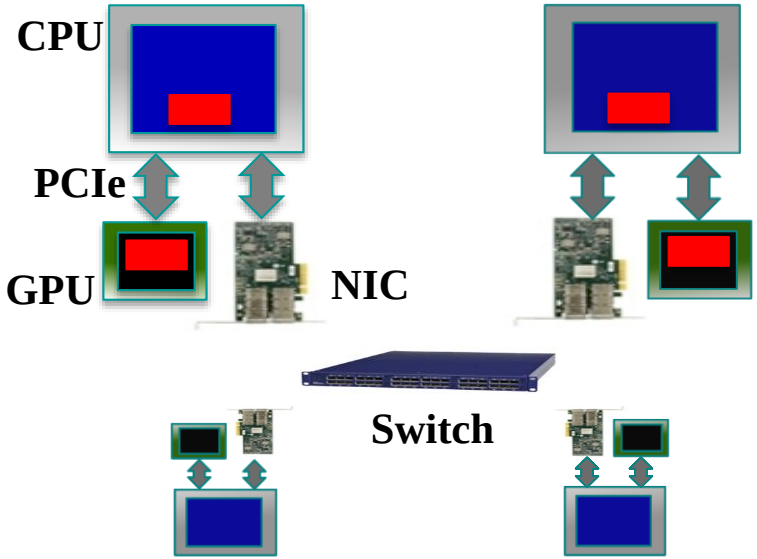
- Data movement in applications with standard MPI and CUDA interfaces

At Sender:

```
cudaMemcpy(s_hostbuf, s_devbuf, . . .);
MPI_Send(s_hostbuf, size, . . .);
```

At Receiver:

```
MPI_Recv(r_hostbuf, size, . . .);
cudaMemcpy(r_devbuf, r_hostbuf, . . .);
```



High Productivity and Low Performance

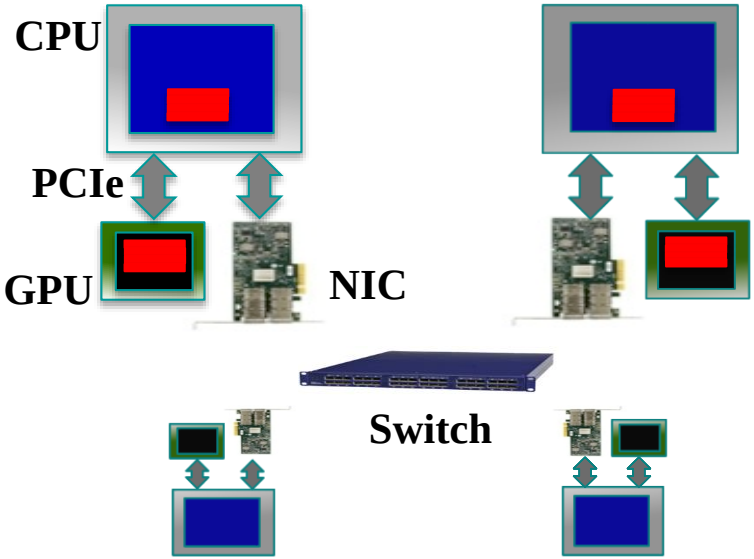
MPI + CUDA - Advanced

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

At Sender:

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

<<Similar at receiver>>



Low Productivity and High Performance

Can this be done within MPI Library?

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

GPU-Aware MPI Library: MVAPICH2-GPU

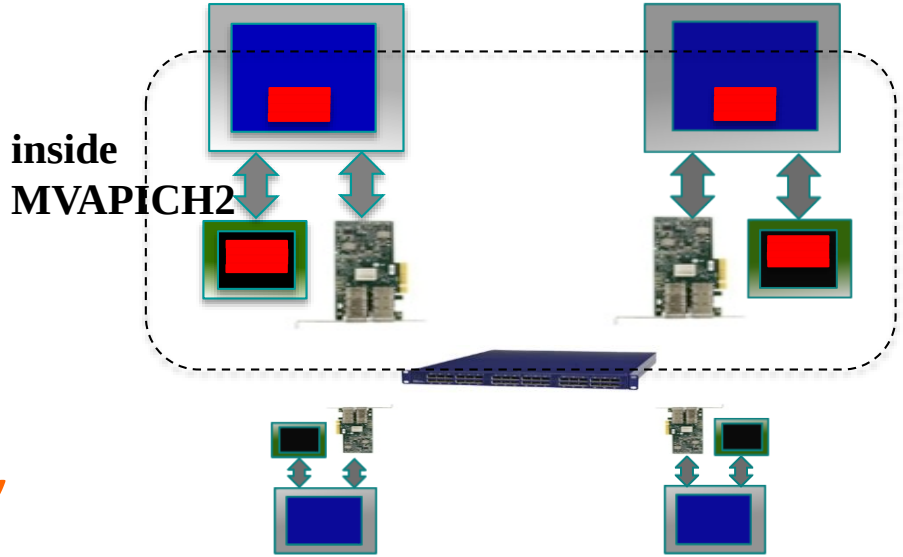
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, …);
```



High Performance and High Productivity

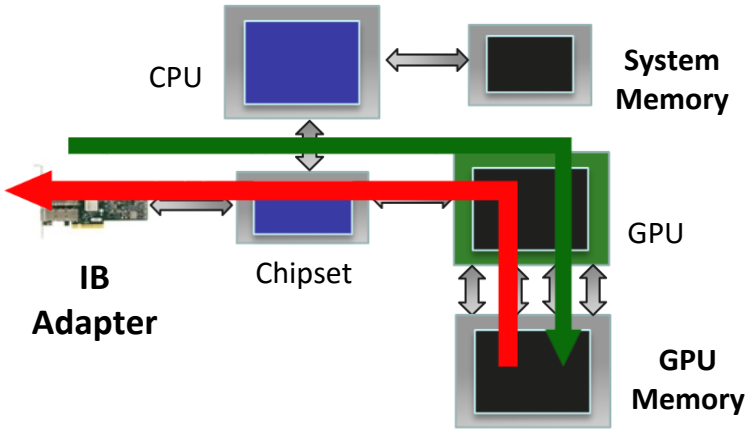
Outline

- Overview of MVAPICH2-X Architecture
- Case Study of Applications Redesign with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
  - Overview of CUDA-Aware Concepts
  - Designing Efficient MPI Runtime for GPU Clusters
  - Designing Efficient OpenSHMEM Runtime for GPU Clusters
- Integrated Support for MICs

GPU-Direct RDMA (GDR) with CUDA

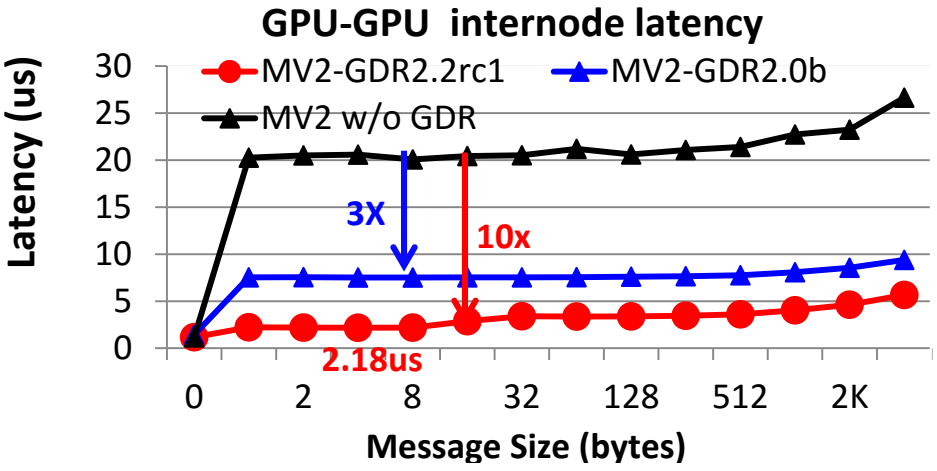
- OFED with support for GPUDirect RDMA is developed by NVIDIA and Mellanox
- OSU has a design of MVAPICH2 using GPUDirect RDMA
  - Hybrid design using GPU-Direct RDMA
    - GPUDirect RDMA and Host-based pipelining
    - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
    - Similar bottlenecks on Haswell
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters

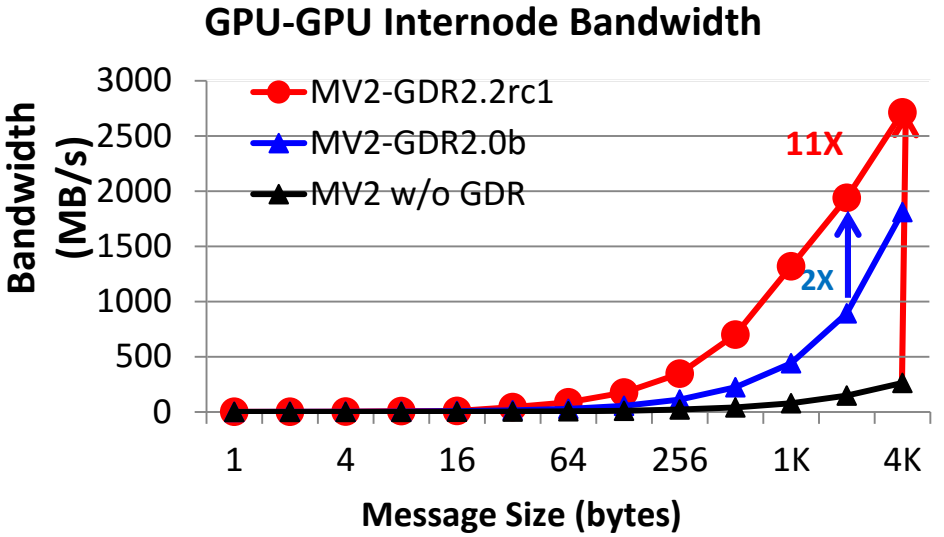
SNB E5-2670 / IVB E5-2680V2

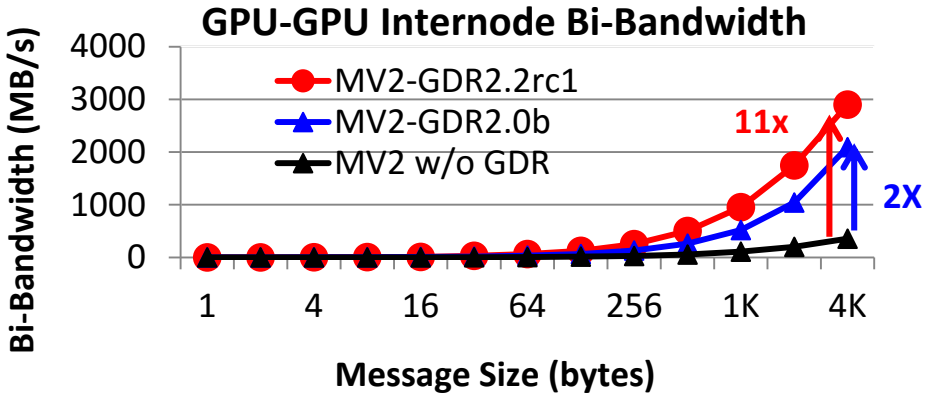


	SNB E5-2670		IVB E5-2680V2	
	Intra-socket	**Inter-sockets**	**Intra-socket**	**Inter-sockets**
P2P read	<1.0 GBs	<300 MBs	3.5 GBs	<300 MBs
P2P write	5.2 GBs	<300 MBs	6.4 GBs	<300 MBs

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

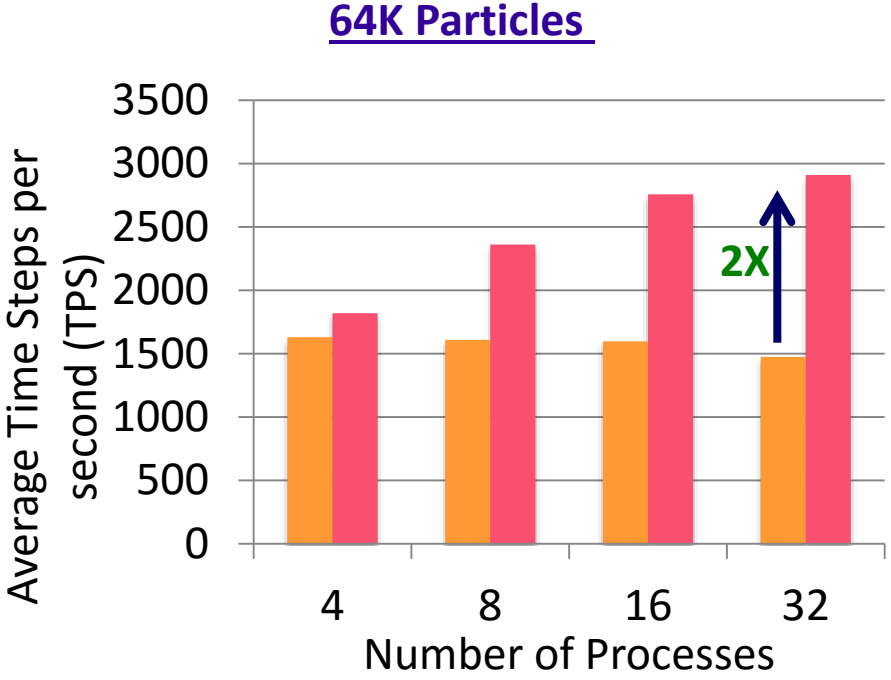


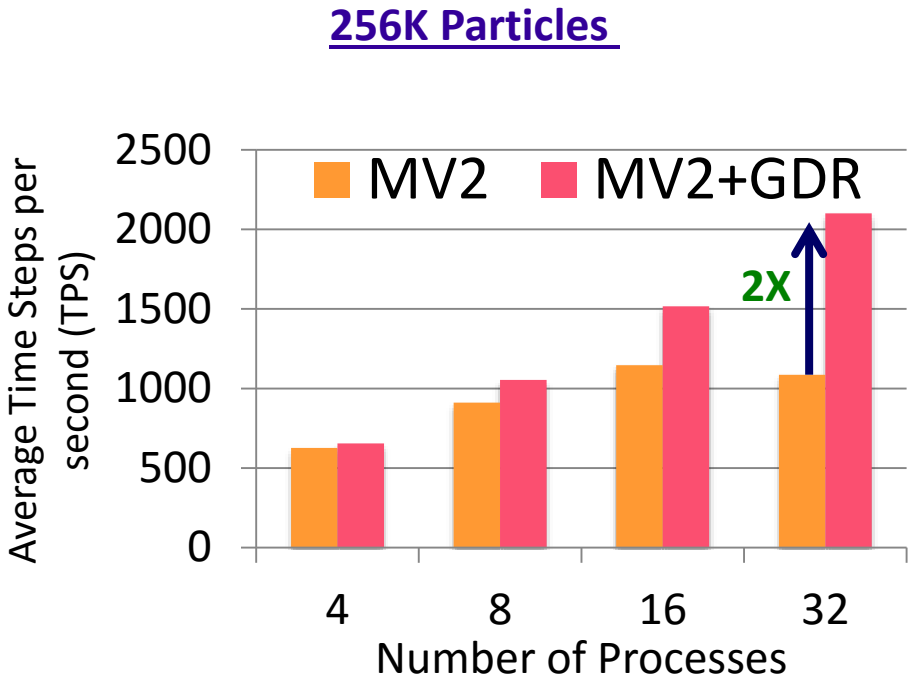




MVAPICH2-GDR-2.2rc1
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 7.5
Mellanox OFED 3.0 with GPU-Direct-RDMA

Application-Level Evaluation (HOOMD-blue)





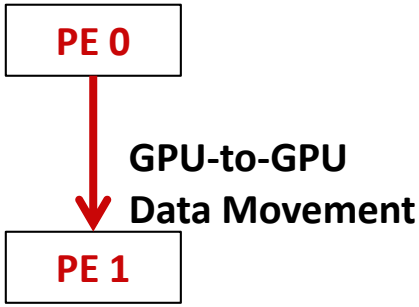
- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomdBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

Outline

- Overview of MVAPICH2-X PGAS Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
  - Overview of CUDA-Aware Concepts
  - Designing Efficient MPI Runtime for GPU Clusters
  - Designing Efficient OpenSHMEM Runtime for GPU Clusters
- Integrated Support for MICs

Limitations of OpenSHMEM for GPU Computing

- OpenSHMEM memory model does not support disjoint memory address spaces - case with GPU clusters



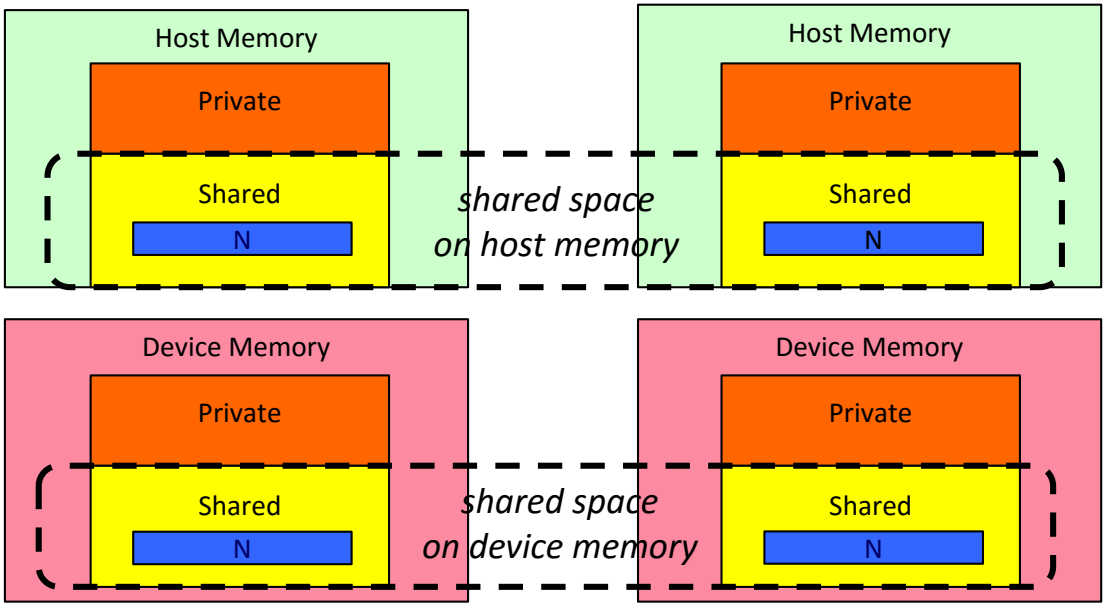
Existing OpenSHMEM Model with CUDA

```
PE 0
host_buf = shmem_malloc (…)
cudaMemcpy (host_buf, dev_buf,  . . . )
shmem_putmem (host_buf, host_buf, size, pe)
shmem_barrier (…)

PE 1
host_buf = shmem_malloc (…)
shmem_barrier (…) . )
cudaMemcpy (dev_buf, host_buf, size, . . . )
```

- Copies severely limit the performance
- Synchronization negates the benefits of one-sided communication
- Similar issues with UPC

Global Address Space with Host and Device Memory

- Extended APIs:
- heap_on_device/heap_on_host
- a way to indicate location of heap
- *host_buf = shmalloc (sizeof(int), 0);*
- *dev_buf = shmalloc (sizeof(int), 1);*



CUDA-Aware OpenSHMEM
Same design for UPC

PE 0

```
dev_buf = shmalloc (size, 1);
shmem_putmem (dev_buf, dev_buf, size, pe)
```

PE 1

```
dev_buf = shmalloc (size, 1);
```

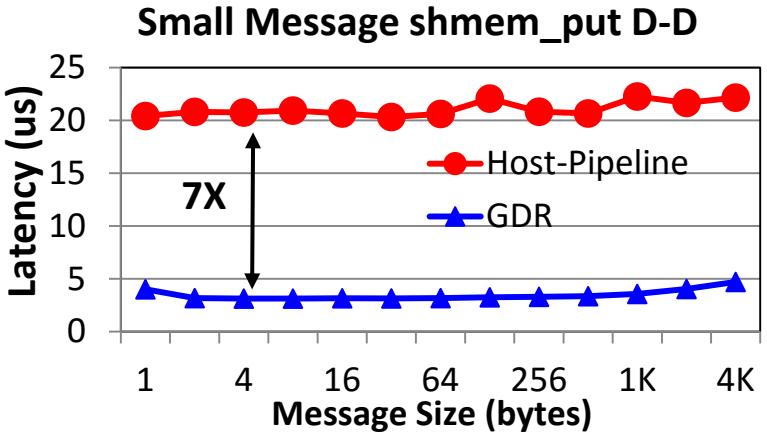
S. Potluri, D. Bureddy, H. Wang, H. Subramoni and D. K. Panda, Extending OpenSHMEM for GPU Computing, IPDPS'13

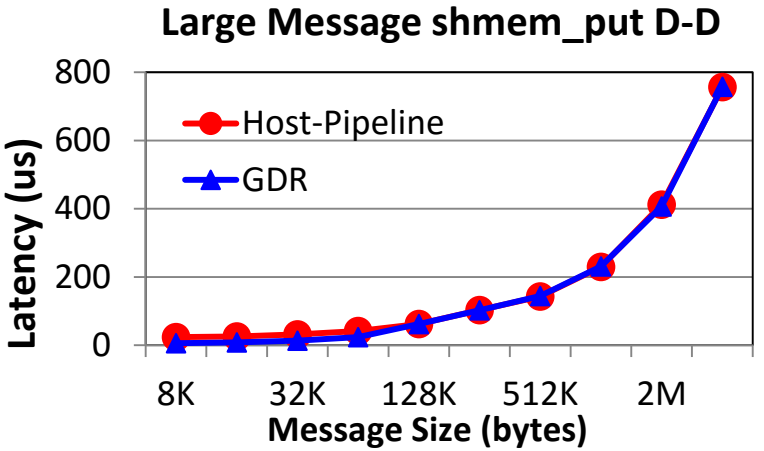
CUDA-aware OpenSHMEM Runtime

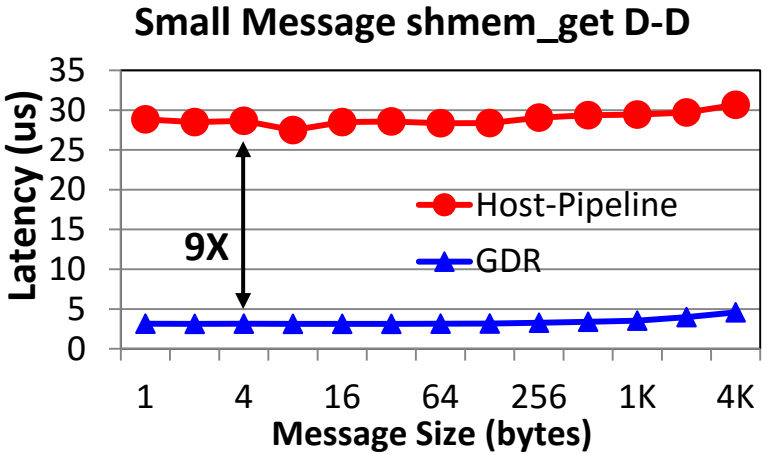
- After device memory becomes part of the global shared space:
  - Accessible through standard OpenSHMEM communication APIs
  - Data movement transparently handled by the runtime
  - Preserves one-sided semantics at the application level
- Efficient designs to handle communication
  - Inter-node transfers use host-staged transfers with pipelining
  - Intra-node transfers use CUDA IPC
  - Possibility to take advantage of GPUDirect RDMA (GDR)

- Goal: Enabling High performance one-sided communications semantics with GPU devices

Exploiting GDR: OpenSHMEM: Inter-node Evaluation

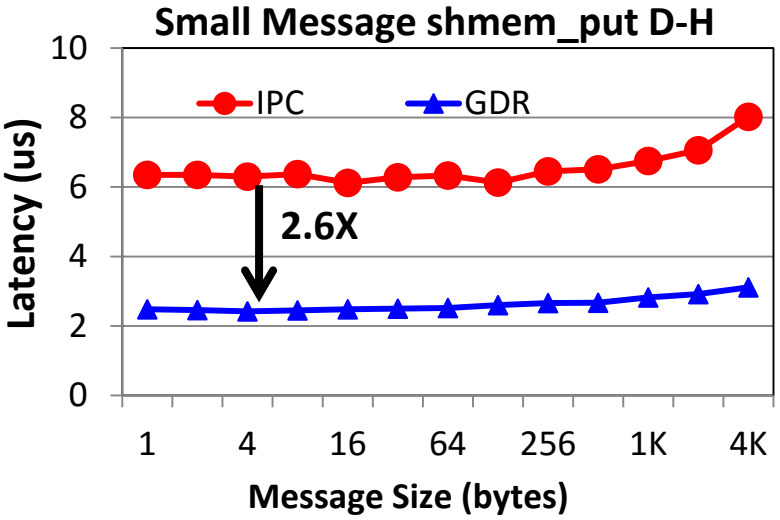


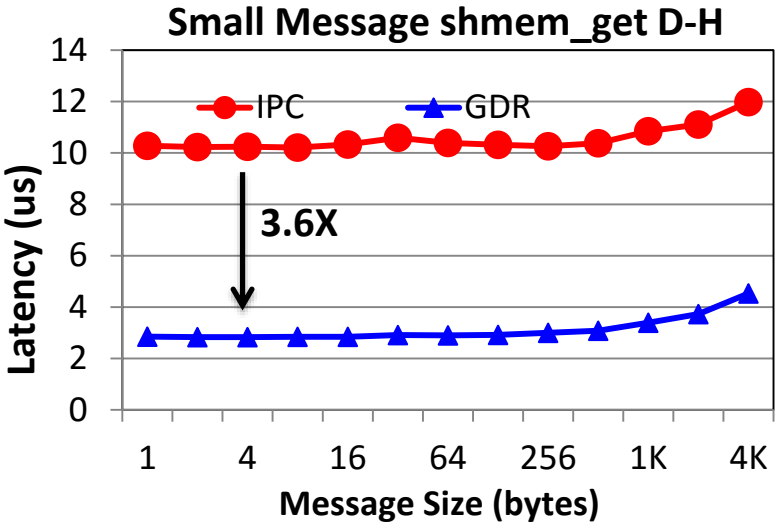




- GDR for small/medium message sizes
- Host-staging for large message to avoid PCIe bottlenecks
- Hybrid design brings best of both
- 3.13 us Put latency for 4B (7X improvement) and 4.7 us latency for 4KB
- 9X improvement for Get latency of 4B

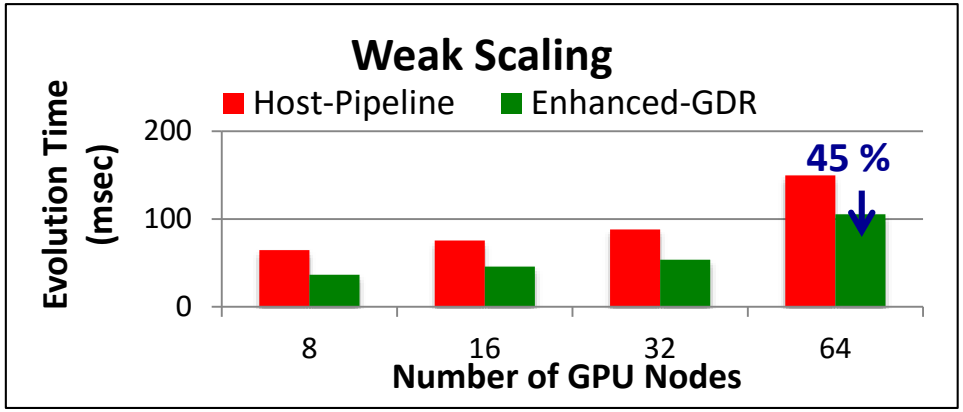
OpenSHMEM: Intra-node Evaluation



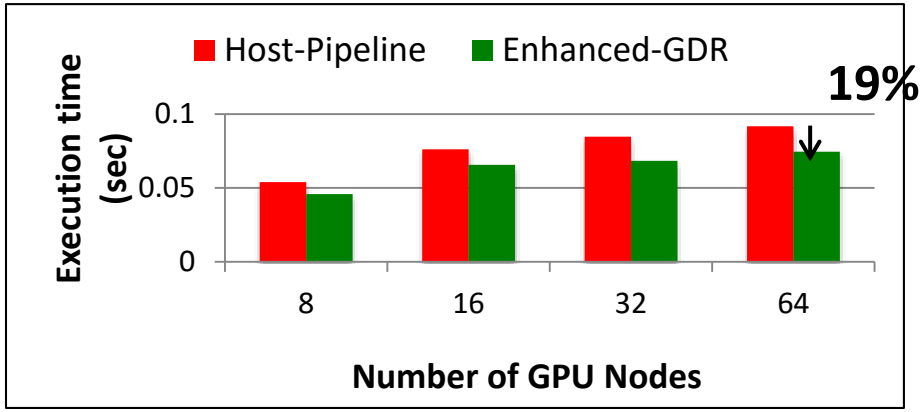


- GDR for small and medium message sizes
- IPC for large message to avoid PCIe bottlenecks
- Hybrid design brings best of both
- 2.42 us Put D-H latency for 4 Bytes (2.6X improvement) and 3.92 us latency for 4 KBytes
- 3.6X improvement for Get operation
- Similar results with other configurations (D-D, H-D and D-H)

Application Evaluation: GPULBM and 2DStencil



GPULBM: 64x64x64



2DStencil 2Kx2K

- Redesign the application
  - CUDA-Aware MPI : Send/Recv=> hybrid CUDA-Aware MPI+OpenSHMEM
  - cudaMalloc =>shmalloc(size,1);
  - MPI_Send/recv => shmem_put + fence
  - 53% and 45%
  - Degradation is due to small Input size
  - Will be available in future MVAPICH2-GDR

- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- New designs achieve 20% and 19% improvements on 32 and 64 GPU nodes

1. K. Hamidouche, A. Venkatesh, A. Awan, H. Subramoni, C. Ching and D. K. Panda, Exploiting GPUDirect RDMA in Designing High Performance OpenSHMEM for GPU Clusters. IEEE Cluster 2015.
2. K. Hamidouche, A. Venkatesh, A. Awan, H. Subramoni, C. Ching and D. K. Panda, CUDA-Aware OpenSHMEM: Extensions and Designs for High Performance OpenSHMEM on GPU Clusters. To appear in PARCO.

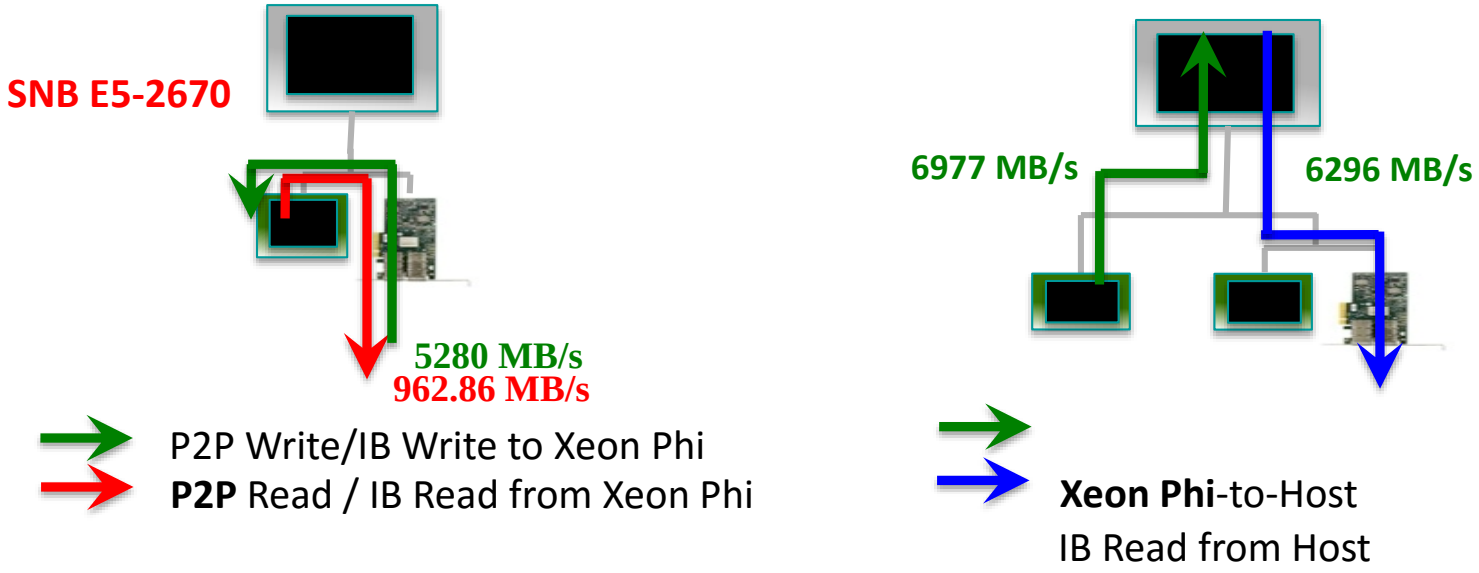
Outline

- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
- Integrated Support for MICs
  - Designing Efficient MPI Runtime for Intel MIC
  - Designing Efficient OpenSHMEM Runtime for Intel MIC

Designs 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

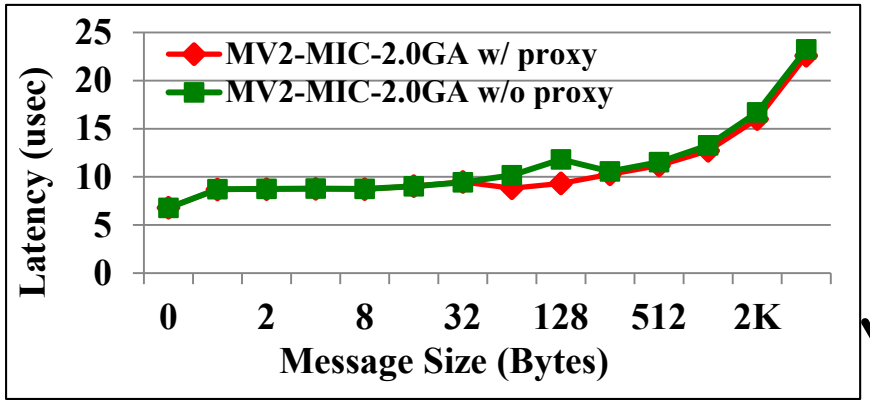
Host Proxy-based Designs in MVAPICH2-MIC



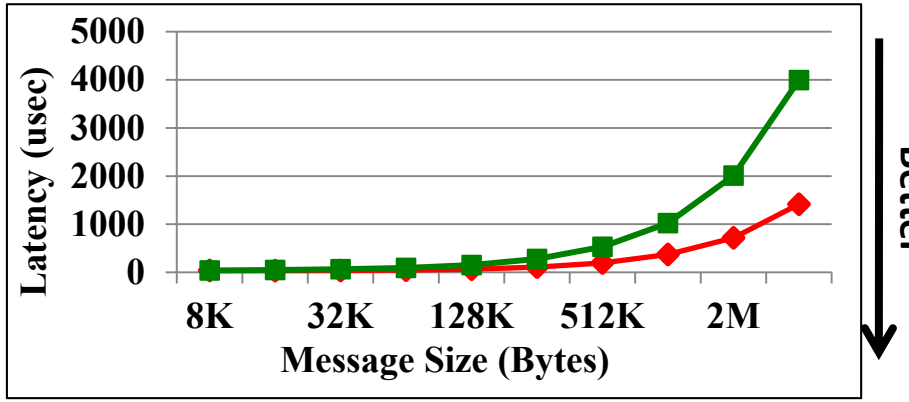
- Direct IB channels is limited by P2P read bandwidth
- MVAPICH2-MIC uses a hybrid DirectIB + host proxy-based approach to work around this

S. Potluri, D. Bureddy, K. Hamidouche, A. Venkatesh, K. Kandalla, H. Subramoni and D. K. Panda, MVAPICH-PRISM: A Proxy-based Communication Framework using InfiniBand and SCIF for Intel MIC Clusters Int'l Conference on Supercomputing (SC '13), November 2013

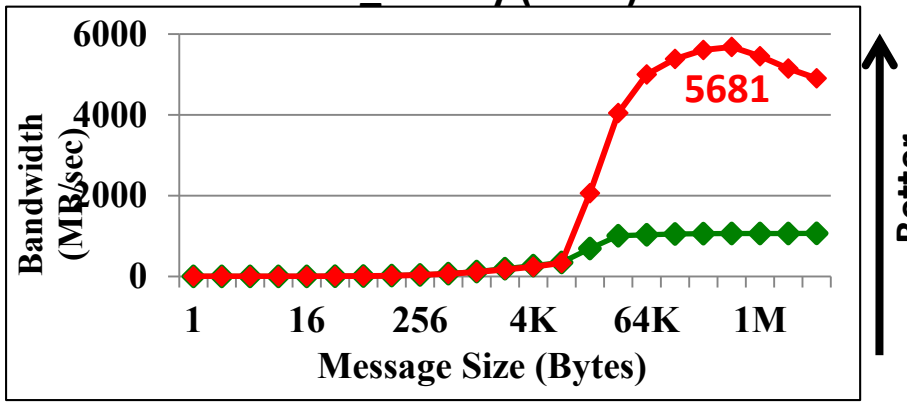
MIC-RemoteMIC Point-to-Point Communication (Active Proxy)



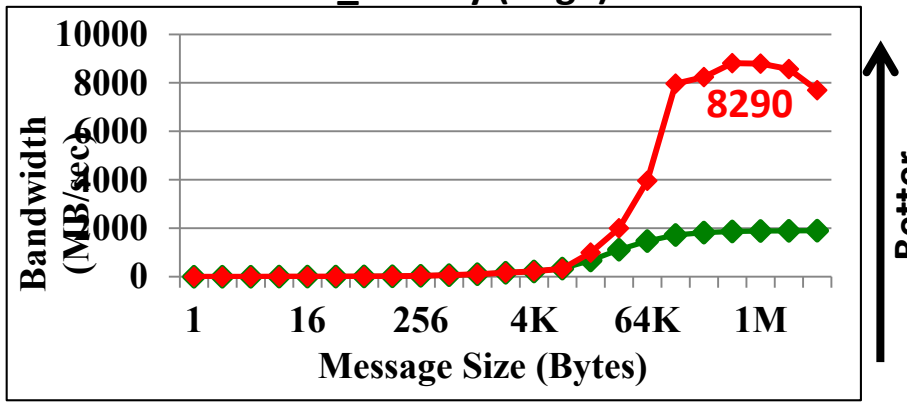
osu_latency (small)



osu_latency (large)



osu_bw



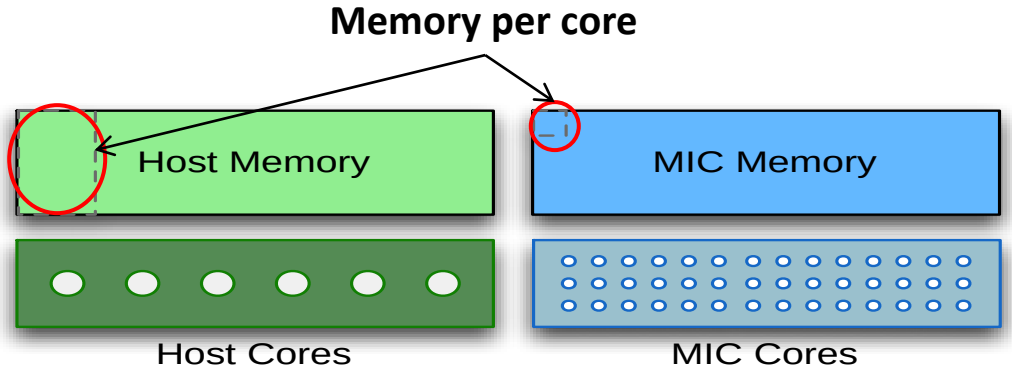
osu_bibw

Outline

- Overview of MVAPICH2-X Architecture
- Case Study of Applications Re-design with Hybrid MPI+OpenSHMEM
- Integrated Support for GPGPUs
- Integrated Support for MICs
  - Designing Efficient MPI Runtime for Intel MIC
  - Designing Efficient OpenSHMEM Runtime for Intel MIC

Need for Non-Uniform Memory Allocation in OpenSHMEM

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

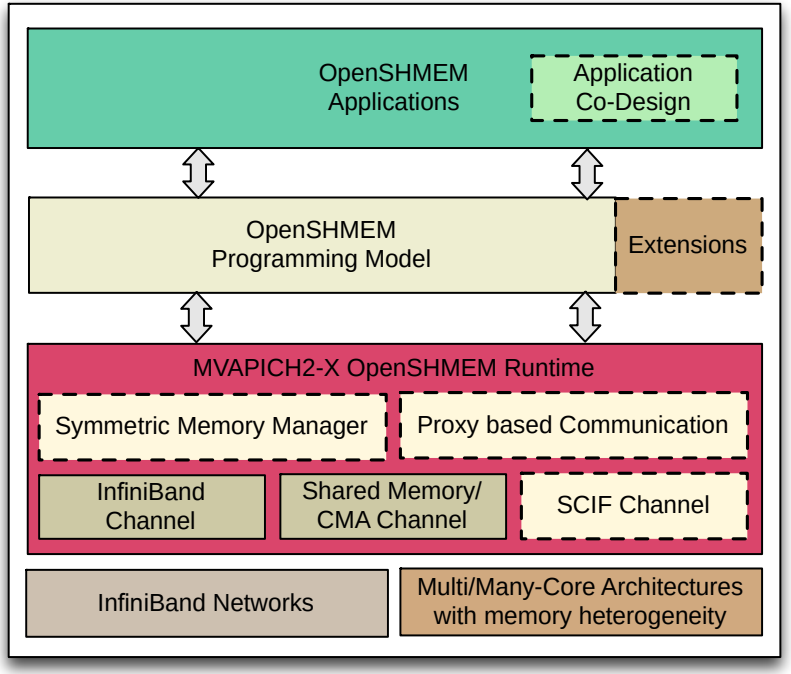


- shmalloc() allocates same amount of memory on all PEs
- For applications running in symmetric mode, this limits the total heap size
- Similar issues for applications (even host-only) with memory load imbalance (Graph500, Out-of-Core Sort, etc.)
- How to allocate different amounts of memory on host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

OpenSHMEM Design for MIC Clusters

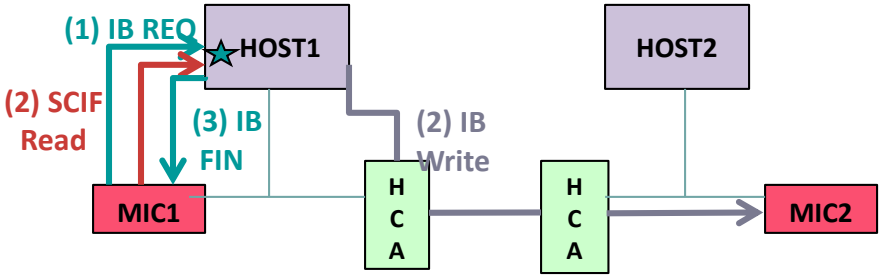
- Non-Uniform Memory Allocation:
  - Team-based Memory Allocation (Proposed Extensions)

```
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);
```

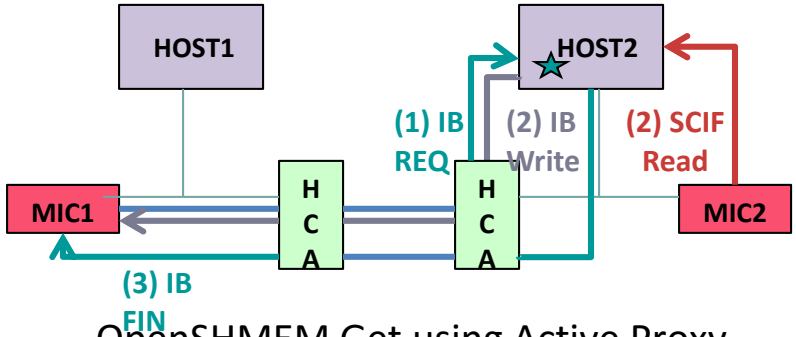


  - Address Structure for non-uniform memory allocations

Proxy-based Designs for OpenSHMEM



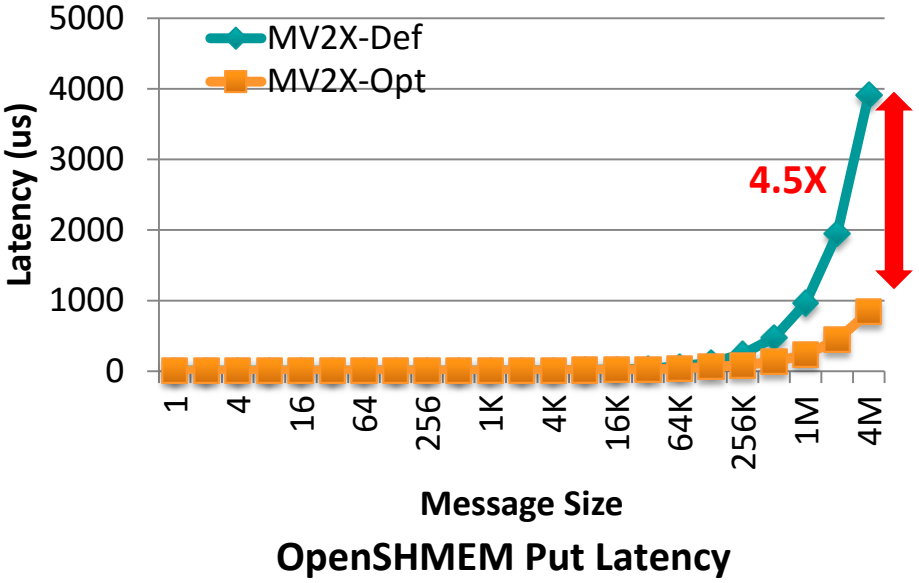
OpenSHMEM Put using Active Proxy



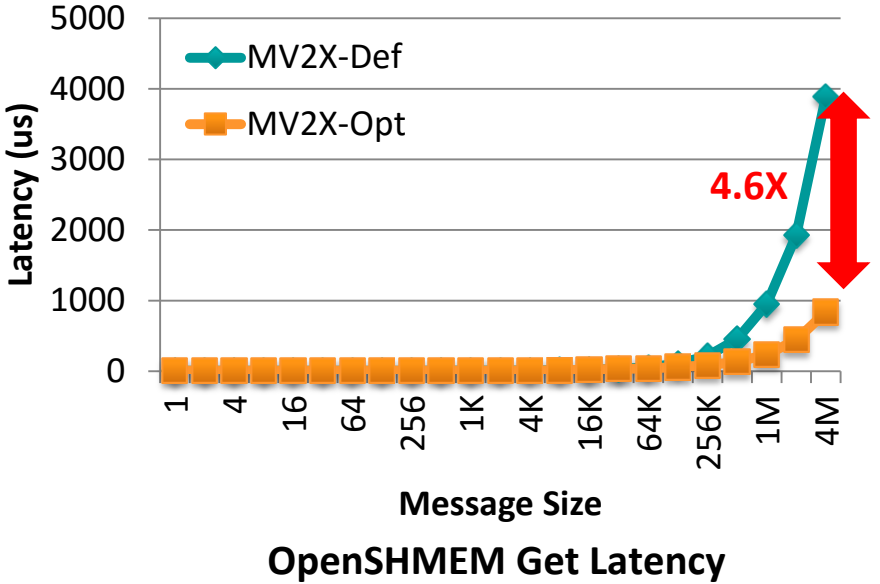
OpenSHMEM Get using Active Proxy

- MIC 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



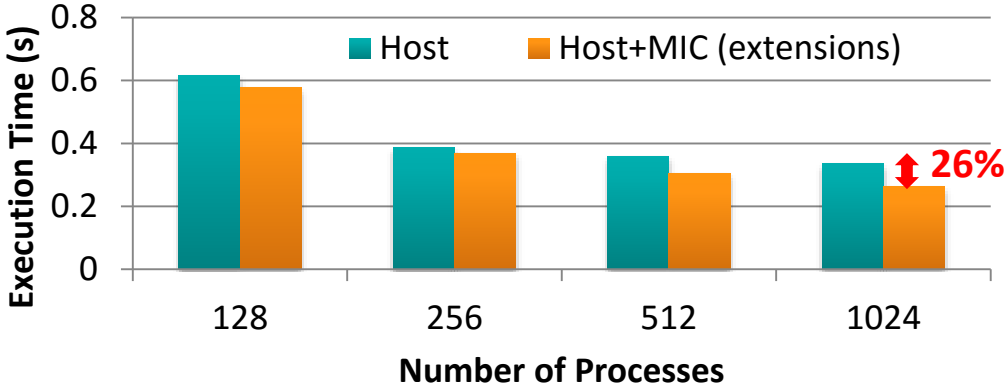
OpenSHMEM Put Latency



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

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 ….

- GPU-initiated communication with GDS technology for OpenSHMEM
  - Similar to NVSHMEM but for inter-node communication
  - Hybrid GDS-NVSHMEM
- Heterogeneous Memory support for OpenSHMEM
  - NVRAM-/NVMe- aware protocols
- Energy-Aware OpenSHMEM runtime
  - Energy-Performance tradeoffs
  - Model extensions for energy-awareness
- Co-design approach at different level
  - Programming Model and Runtime
  - Hardware Support
  - Application

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Gugnani (Ph.D.)
- J. Hashimi (Ph.D.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- K. Kulkarni (M.S.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Past Students

- 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.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)

Past Post-Docs

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

Current Research Scientists

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

Current Research Specialist

- M. Arnold
- J. Perkins

- 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

Upcoming 4th Annual MVAPICH User Group (MUG) Meeting

- August 15-17, 2016; Columbus, Ohio, USA
- Keynote Talks, Invited Talks, Invited Tutorials by Intel, NVIDIA, Contributed Presentations, Tutorial on MVAPICH2, MVAPICH2-X, MVAPICH2-GDR, MVAPICH2-MIC,MVAPICH2-Virt, MVAPICH2-EA, OSU INAM as well as other optimization and tuning hints
- Tutorials
  - Recent Advances in CUDA for GPU Cluster Computing
    - Davide Rossetti, Sreeram Potluri (NVIDIA)
  - Designing High-Performance Software on Intel Xeon Phi and Omni-Path Architecture
    - Ravindra Babu Ganapathi, Sayantan Sur (Intel)
  - Enabling Exascale Co-Design Architecture
    - Devendar Bureddy (Mellanox)
  - How to Boost the Performance of Your MPI and PGAS Applications with MVAPICH2 Libraries
    - The MVAPICH Team
- Demo and Hands-On Session
  - Performance Engineering of MPI Applications with MVAPICH2 and TAU
    - Sameer Shende (University of Oregon, Eugene) with Hari Subramoni, and Khaled Hamidouche (The Ohio State University)
  - Visualize and Analyze your Network Activities using INAM (InfiniBand Networking and Monitoring tool)
    - MVAPICH Group, The Ohio State University(The Ohio State University)
- Keynote Speakers
  - Thomas Schulthess (CSCS, Switzerland)
  - Gilad Shainer (Mellanox)
- Invited Speakers (Confirmed so far)
  - Kapil Arya (Mesosphere, Inc. and Northeastern University)
  - Jens Glaser (University of Michigan)
  - Darren Kerbyson (Pacific Northwest National Laboratory)
  - Ignacio Laguna (Lawrence Livermore National Laboratory)
  - Adam Moody (Lawrence Livermore National Laboratory)
  - Takeshi Nanri (University of Kyushu, Japan)
  - Davide Rossetti (NVIDIA)
  - Sameer Shende (University of Oregon)
  - Karl Schulz (Intel)
  - Filippo Spiga (University of Cambridge, UK)
  - Sayantan Sur (Intel)
  - Rick Wagner (San Diego Supercomputer Center)
  - Yajuan Wang (Inspur, China)
- **Student Travel Support available through NSF**
- **More details at: http://mug.mvapich.cse.ohio-state.edu**

International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)

ESPM2 2016 will be held with the Supercomputing Conference (SC '16), at Salt Lake City, Utah, on Friday, November 18th, 2016

http://web.cse.ohio-state.edu/~hamidouc/ESPM2/espm2_16.html#program

In Cooperation with ACM SIGHPC

Paper Submission Deadline: August 26th, 2016

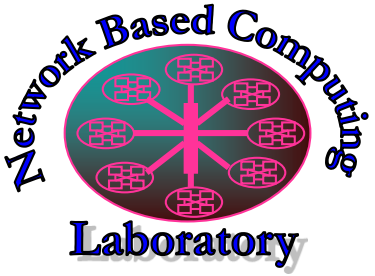
Author Notification: September 30th, 2016

Camera Ready: October 7th, 2016

ESPM2 2015 was held with the Supercomputing Conference (SC '15), at Austin, Texas, on Sunday, November 15th, 2015

http://web.cse.ohio-state.edu/~hamidouc/ESPM2/espm2.html#program

Thank You!



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

MVAPICH

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

panda@cse.ohio-state.edu