Supporting PGAS Models (UPC and OpenSHMEM) on MIC Clusters

Presentation at IXPUG Meeting, July 2014

by

Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

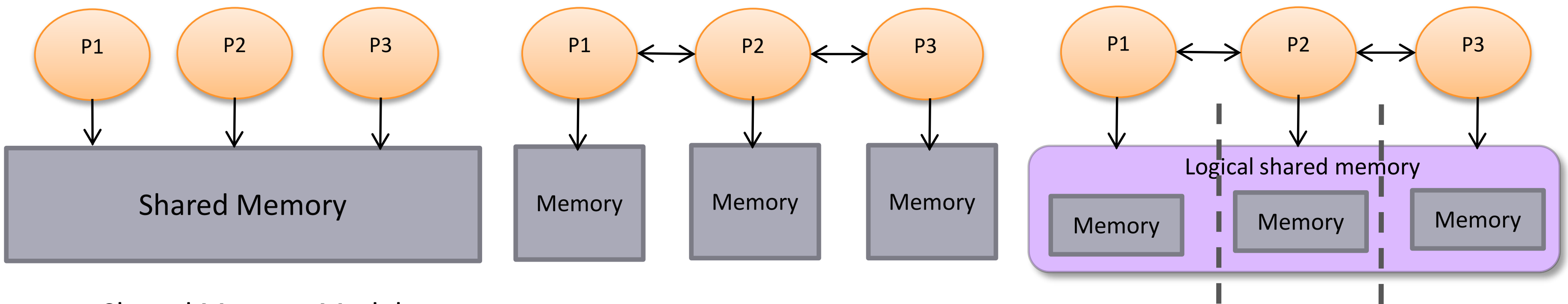
Khaled Hamidouche

The Ohio State University

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

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

Parallel Programming Models Overview



Shared Memory Model

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.
- Additionally, OpenMP can be used to parallelize computation within the node
- Each model has strengths and drawbacks - suite different problems or applications

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
    - Chapel
  - Libraries
    - OpenSHMEM
    - Global Arrays

SHMEM

- SHMEM: Symmetric Hierarchical MEMory library
- One-sided communications library – had been around for a while
- Similar to MPI, processes are called PEs, data movement is explicit through library calls
- Provides globally addressable memory using symmetric memory objects (more in later slides)
- Library routines for
  - Symmetric object creation and management
  - One-sided data movement
  - Atomics
  - Collectives
  - Synchronization

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 application 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

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, …

MPI+PGAS for Exascale Architectures and Applications

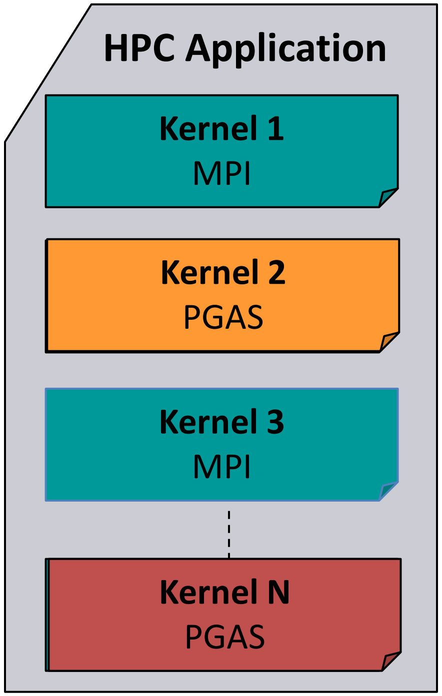
- Hierarchical architectures with multiple address spaces
- (MPI + PGAS) Model
  - MPI across address spaces
  - PGAS within an address space
- MPI is good at moving data between address spaces
- Within an address space, MPI can interoperate with other shared memory programming models

- 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"

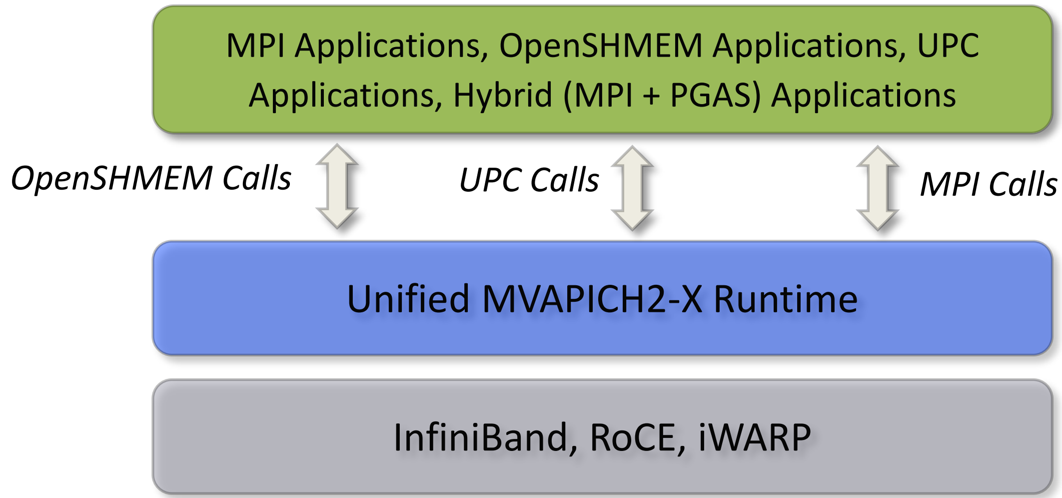


** The International Exascale Software Roadmap, Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420*

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 (HPC Centers, Industry and Universities) in 72 countries**
  - **More than 218,000 downloads from OSU site directly**
  - Empowering many TOP500 clusters
    - 7th ranked 519,640-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 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-X for Hybrid MPI + PGAS Applications

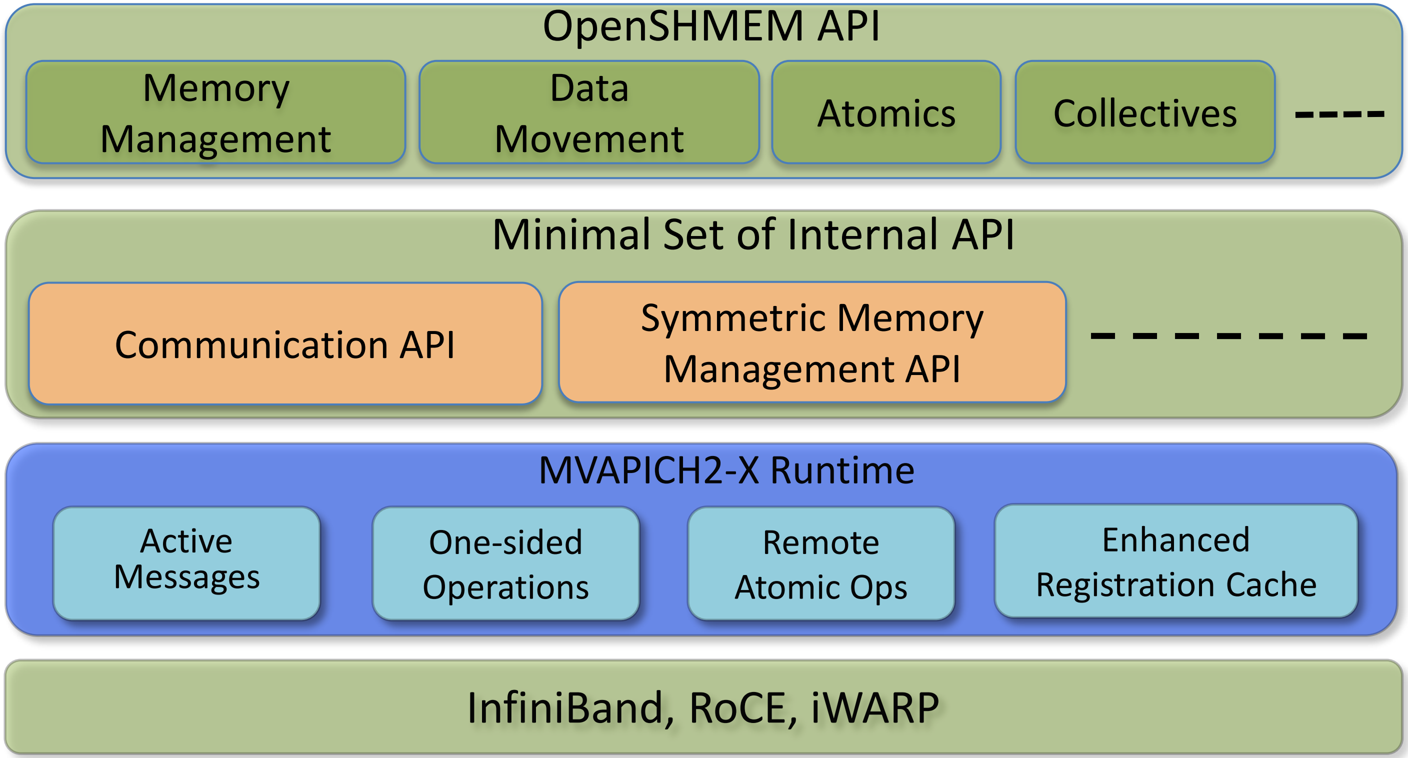


- Unified communication runtime for MPI, UPC, OpenSHMEM available with MVAPICH2-X 1.9 onwards!
  - http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC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

Outline

- OpenSHMEM and UPC for Host
- Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
- OpenSHMEM for MIC
- UPC for MIC

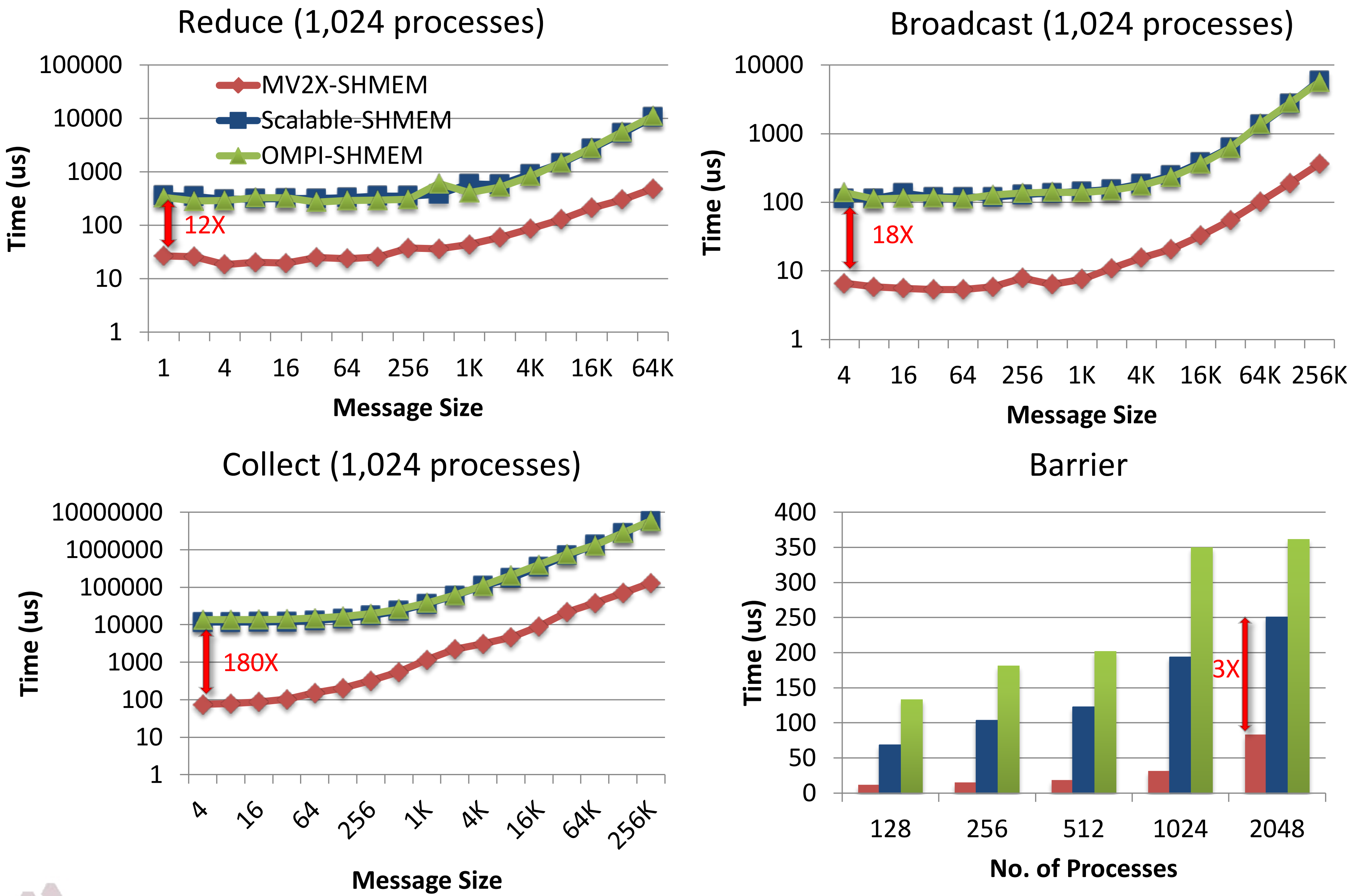
OpenSHMEM Design in MVAPICH2-X



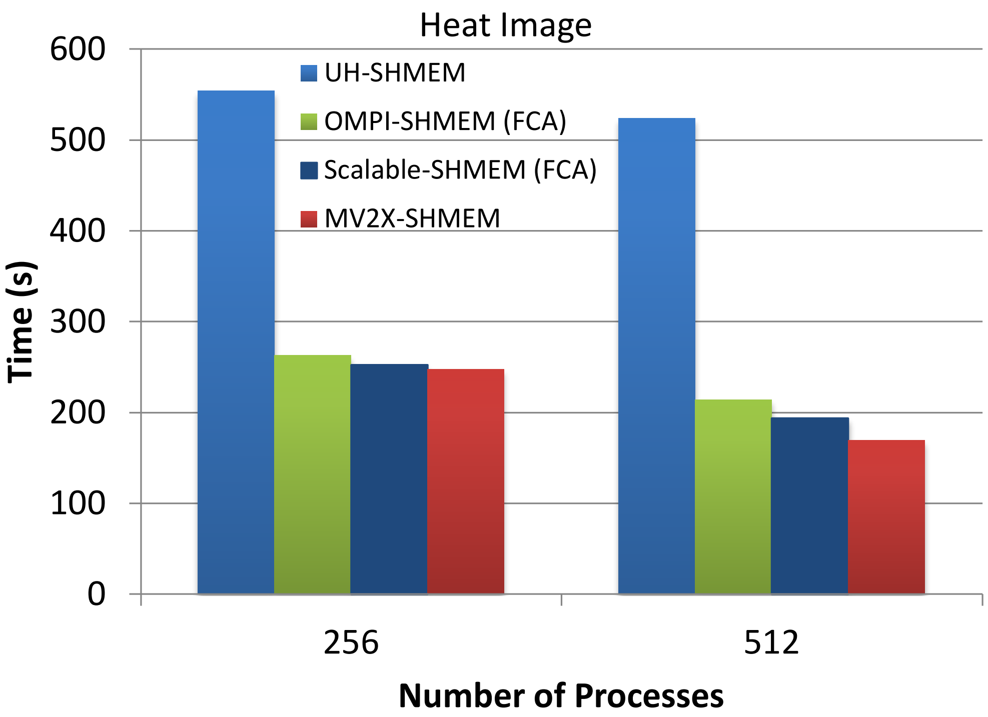
- OpenSHMEM Stack based on OpenSHMEM Reference Implementation
- OpenSHMEM Communication over MVAPICH2-X Runtime
  - Uses active messages, atomic and one-sided operations and remote registration cache

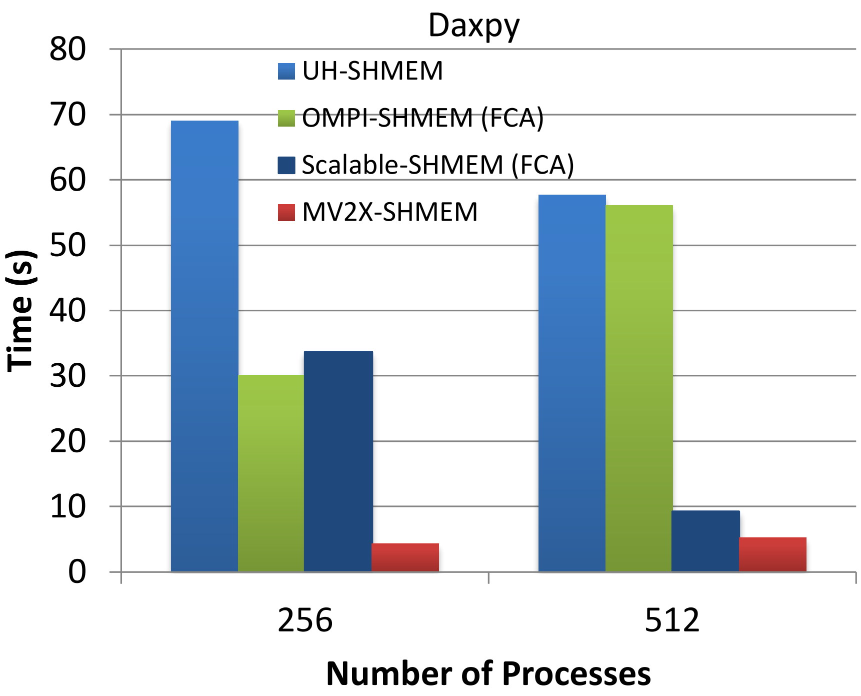
J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012.

OpenSHMEM Collective Communication Performance



OpenSHMEM Application Evaluation

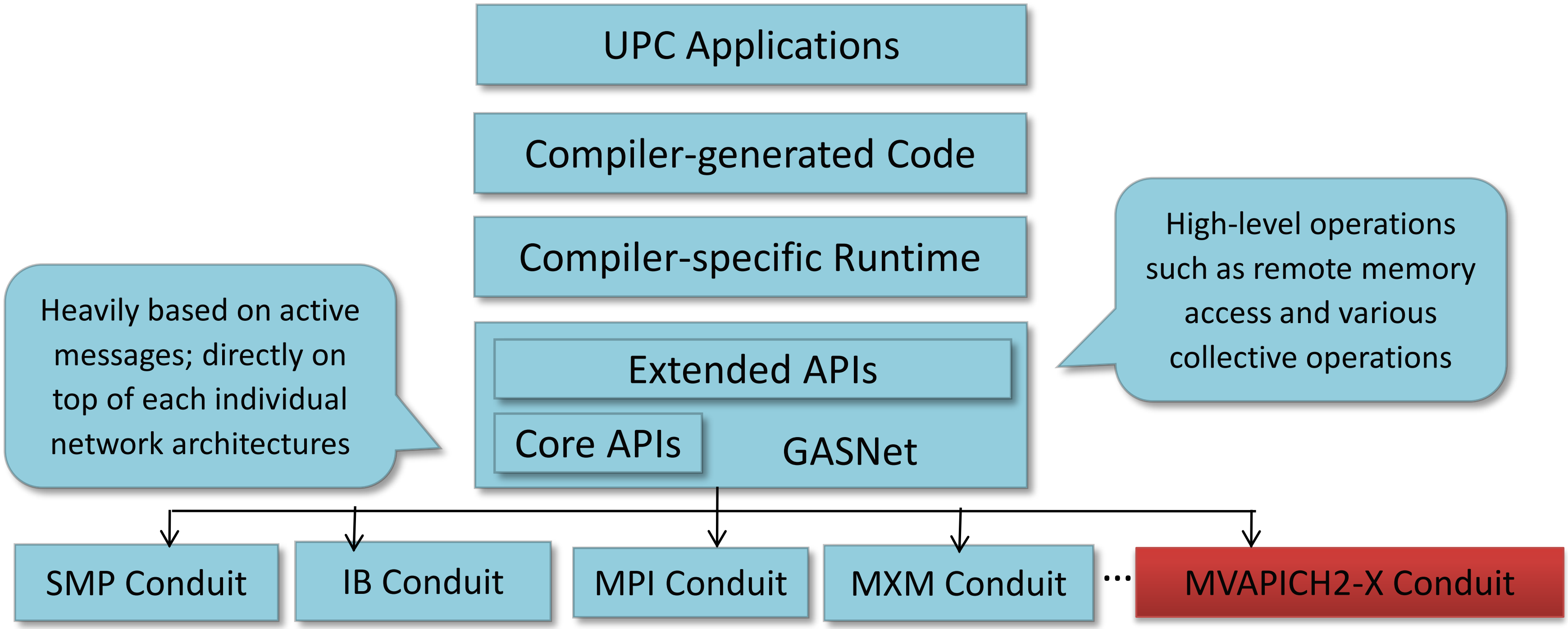




- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- Execution time for 2DHeat Image at 512 processes (sec):
  - UH-SHMEM – 523, OMPI-SHMEM – 214, Scalable-SHMEM – 193, MV2X-SHMEM – 169
- Execution time for DAXPY at 512 processes (sec):
  - UH-SHMEM – 57, OMPI-SHMEM – 56, Scalable-SHMEM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

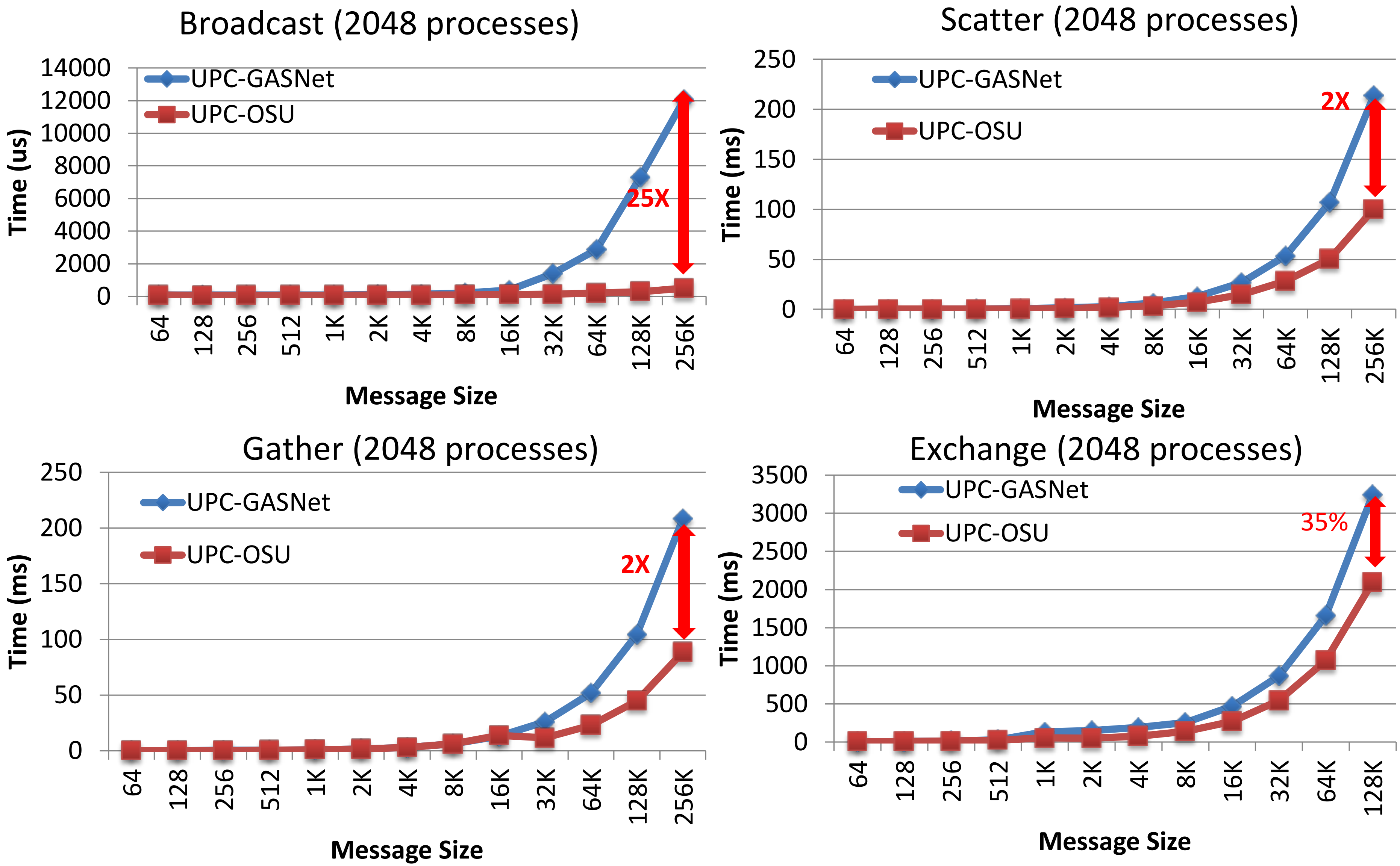
J. Jose, J. Zhang, A. Venkatesh, S. Potluri, and D. K. Panda, A Comprehensive Performance Evaluation of OpenSHMEM Libraries on InfiniBand Clusters, OpenSHMEM Workshop (OpenSHMEM'14), March 2014

MVAPICH2-X Support for Berkeley UPC Runtime



- GASNet (Global-Address Space Networking) is a language-independent, low-level networking layer that provides support for PGAS language
- Support multiple networks through different conduit: MVAPICH2-X Conduit is available in MVAPICH2-X release, which support UPC/OpenMP/MPI on InfiniBand

UPC Collectives Performance

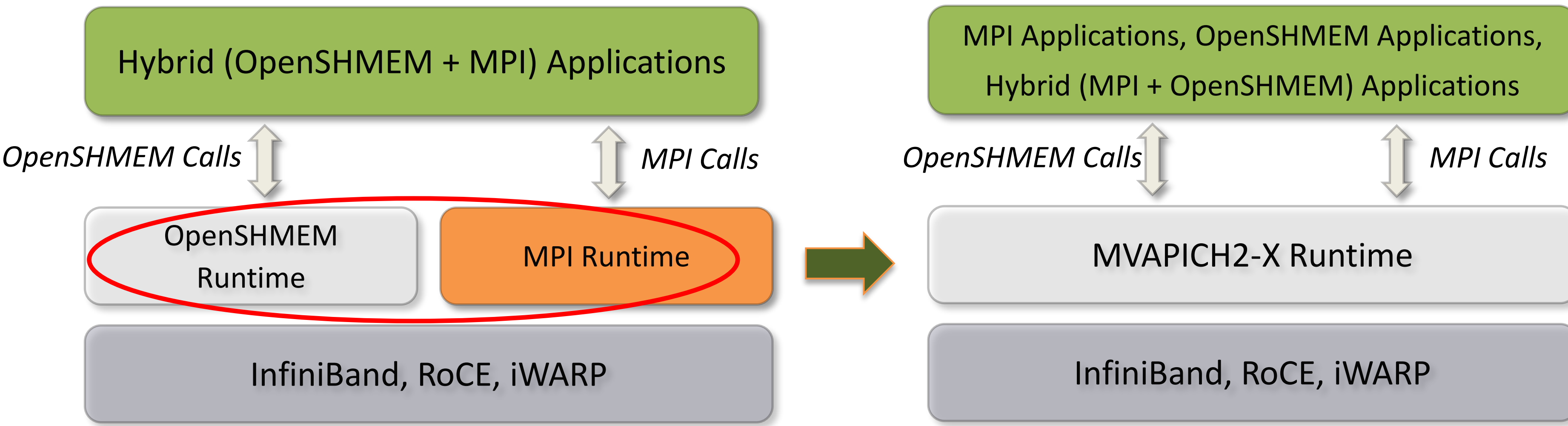


J. Jose, K. Hamidouche, J. Zhang, A. Venkatesh, and D. K. Panda, Optimizing Collective Communication in UPC (HiPS'14, in association with IPDPS'14)

Outline

- OpenSHMEM and UPC for Host
- Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
- OpenSHMEM for MIC
- UPC for MIC

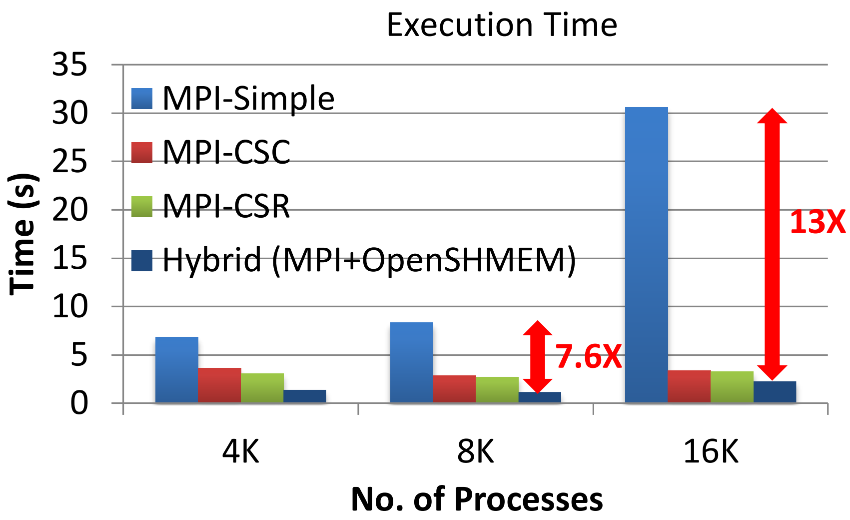
Unified Runtime for Hybrid MPI + OpenSHMEM Applications

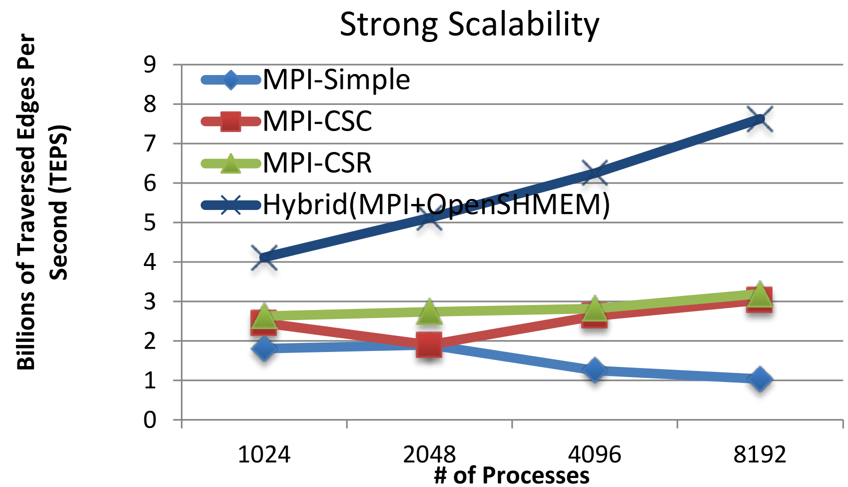


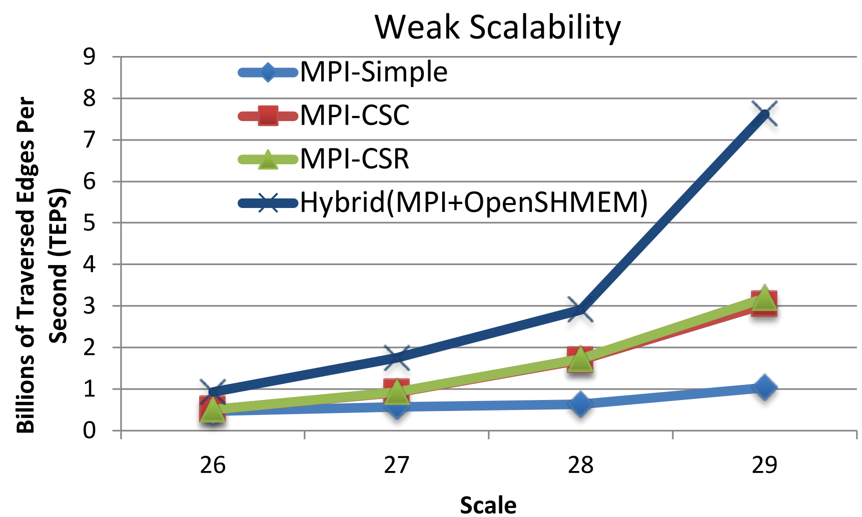
- Optimal network resource usage
- No deadlock because of single runtime
- Better performance

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 Graph500 Design





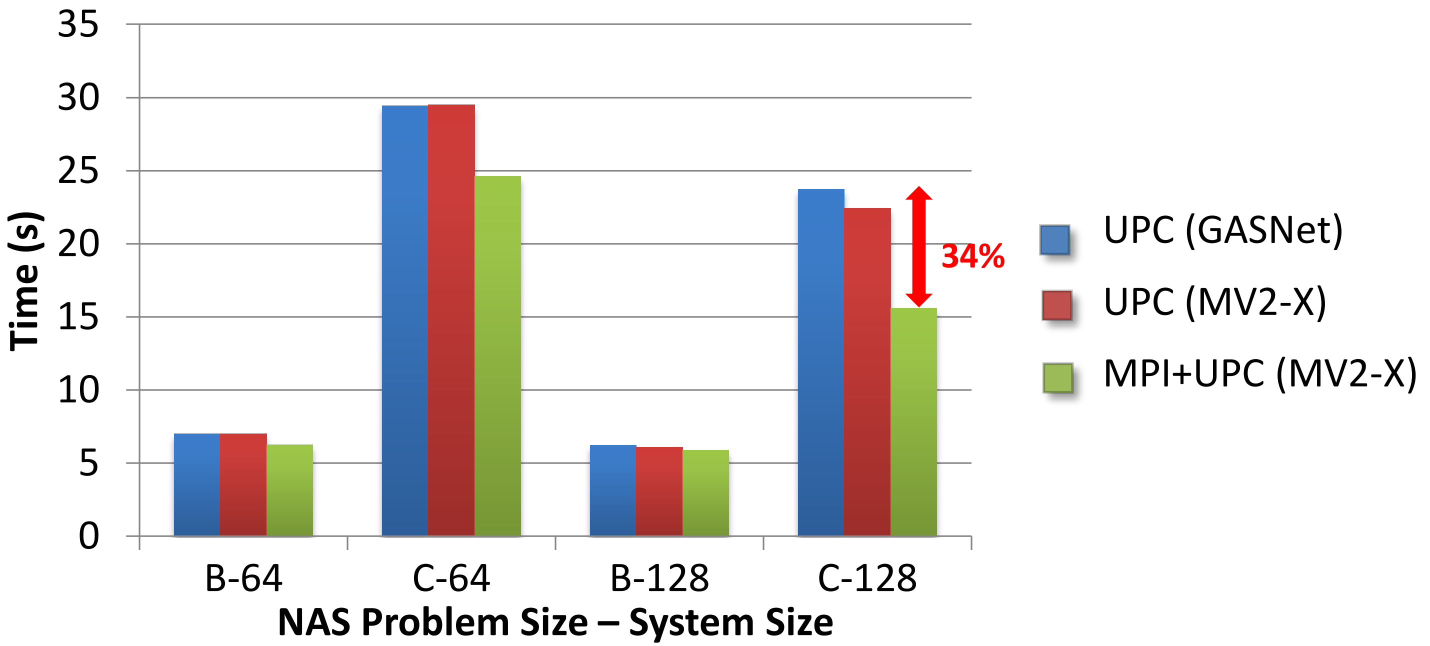


- Performance of Hybrid (MPI+OpenSHMEM) Graph500 Design
  - 8,192 processes
    - 2.4X improvement over MPI-CSR
    - 7.6X improvement over MPI-Simple
  - 16,384 processes
    - 1.5X improvement over MPI-CSR
    - 13X improvement over MPI-Simple

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+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 (MV2-X)

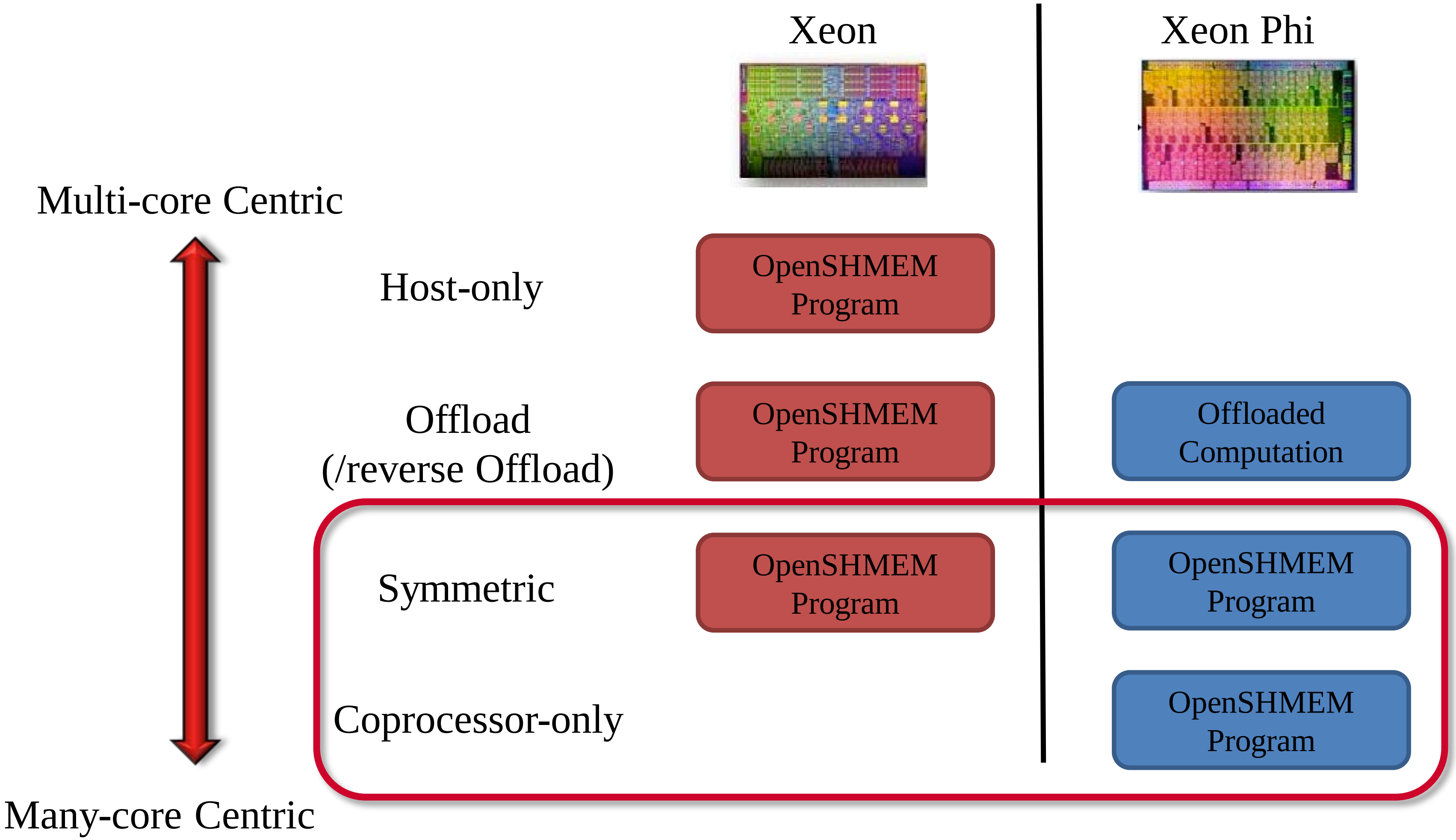
J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, Int'l Conference on Partitioned Global Address Space Programming Models (PGAS) 2010

Outline

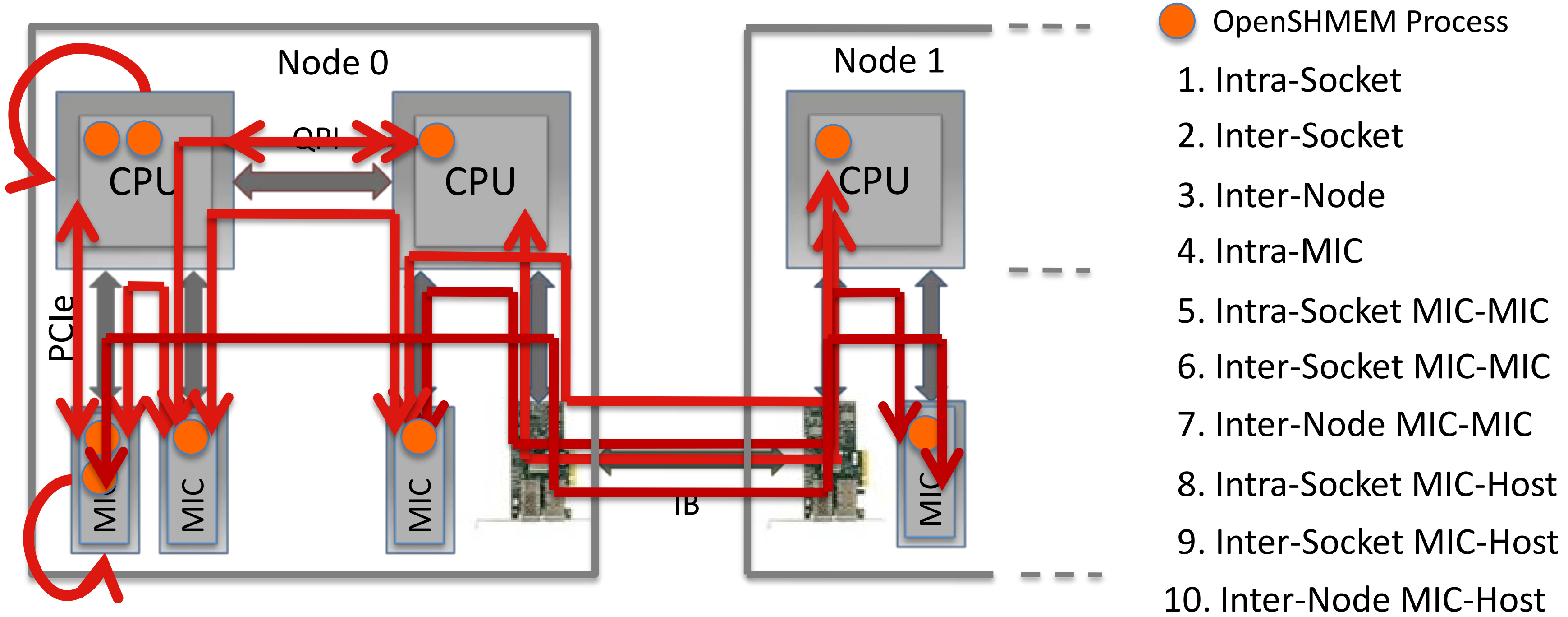
- OpenSHMEM and UPC for Host
- Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
- OpenSHMEM for MIC
- UPC for MIC

HPC Applications on MIC Clusters

- Flexibility in launching HPC jobs on Xeon Phi Clusters



Data Movement on Intel Xeon Phi Clusters



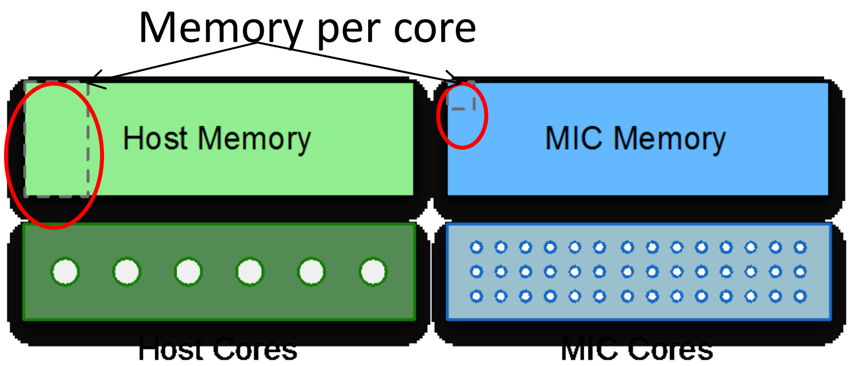
11. Inter-Node MIC-MIC with IB on remote socket

and more . . .

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

Need for Non-Uniform Memory Allocation in OpenSHMEM

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



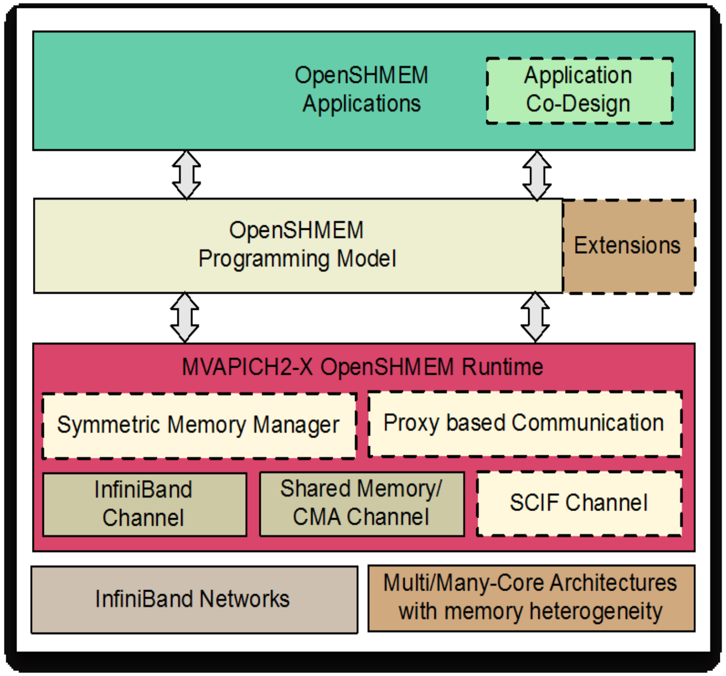
- shmalloc() allocates same amount of memory on all PEs
- For applications running in symmetric mode, this limits the total heap size
- Similar issues for applications (even host-only) with memory load imbalance (Graph500, Out-of-Core Sort, etc.)
- How to allocate different amounts of memory on host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

OpenSHMEM Design for MIC Clusters

- Non-Uniform Memory Allocation:
  - Team-based Memory Allocation

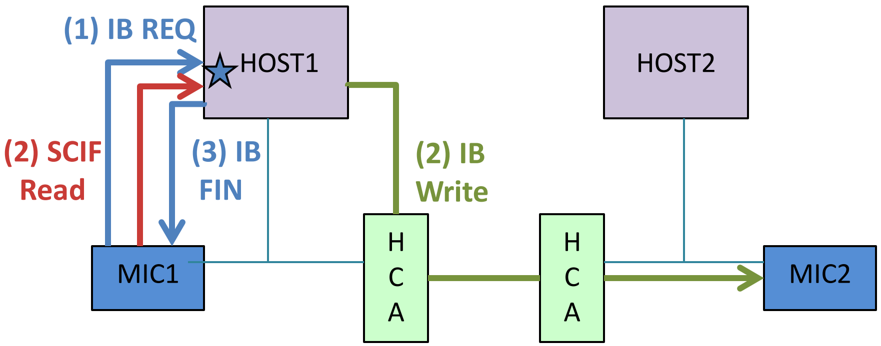
    (Proposed Extensions)

```
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);
```

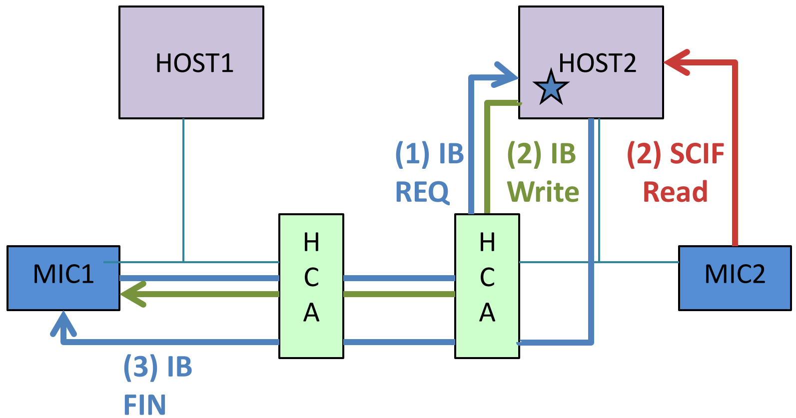


  - Address Structure for non-uniform memory allocations

Proxy-based designs for OpenSHMEM



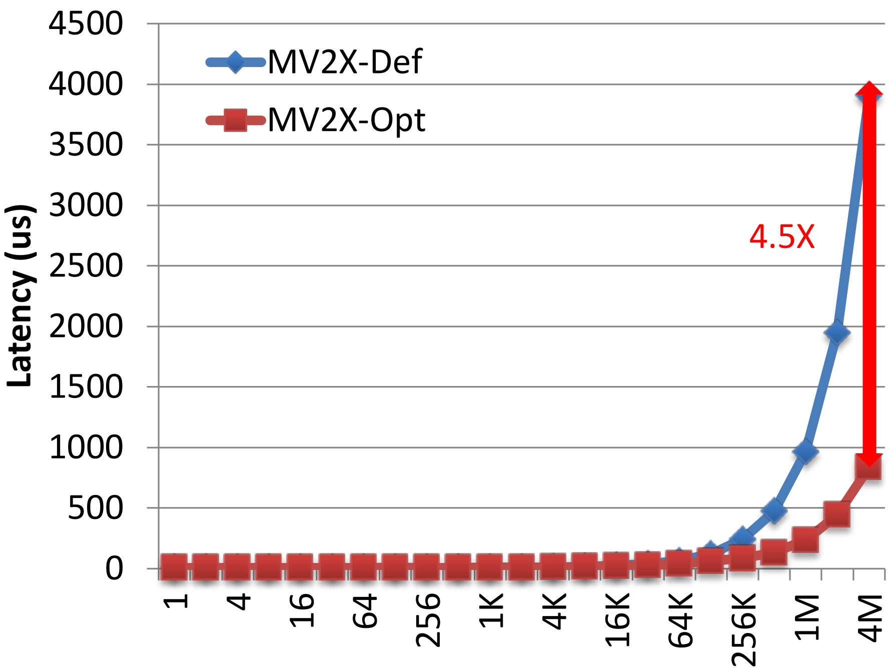
OpenSHMEM Put using Active Proxy



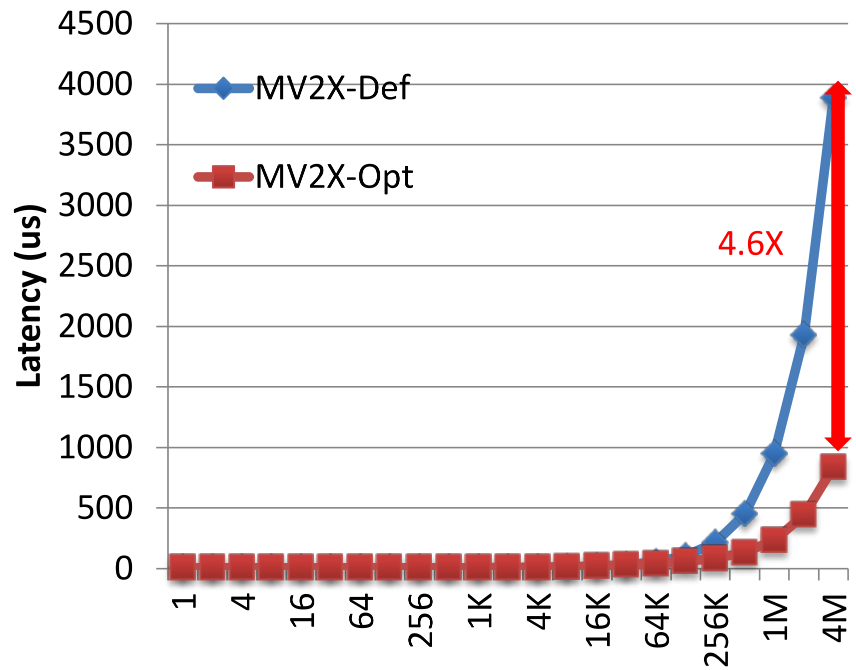
OpenSHMEM Get using Active Proxy

- 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



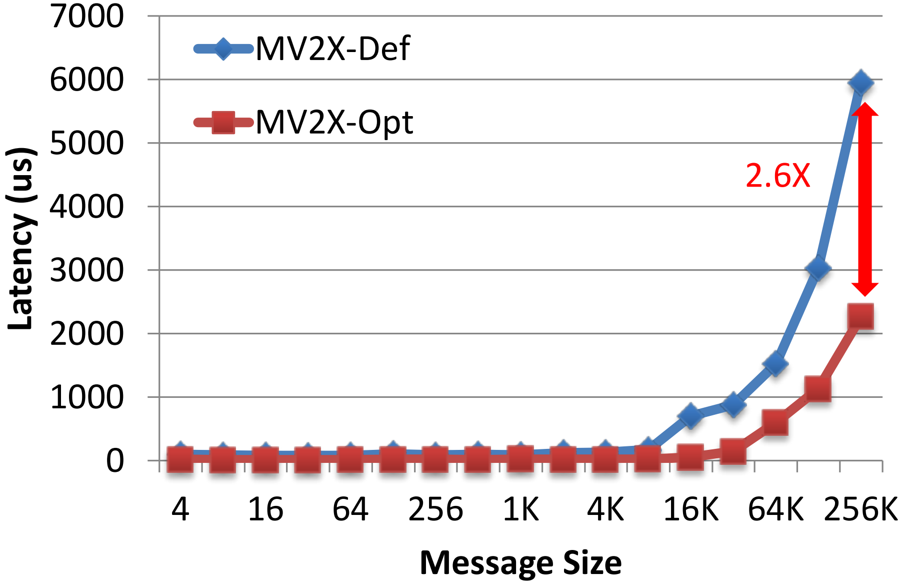
Message Size
OpenSHMEM Put Latency



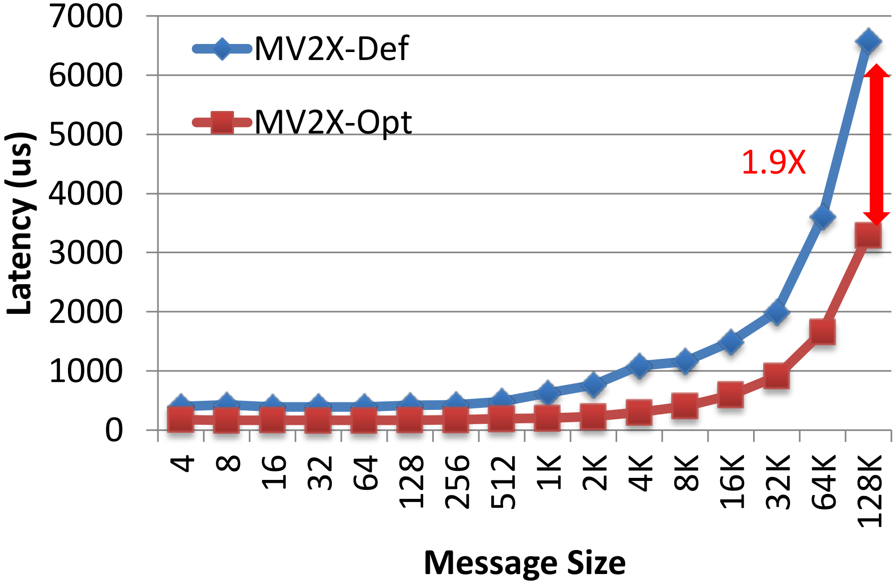
Message Size
OpenSHMEM Get Latency

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

OpenSHMEM Collectives Performance



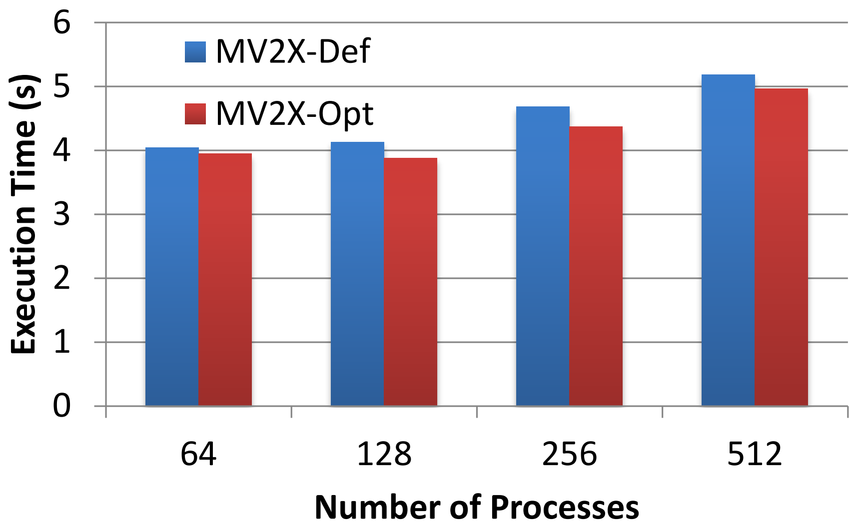
Broadcast Latency (1,024 processes)



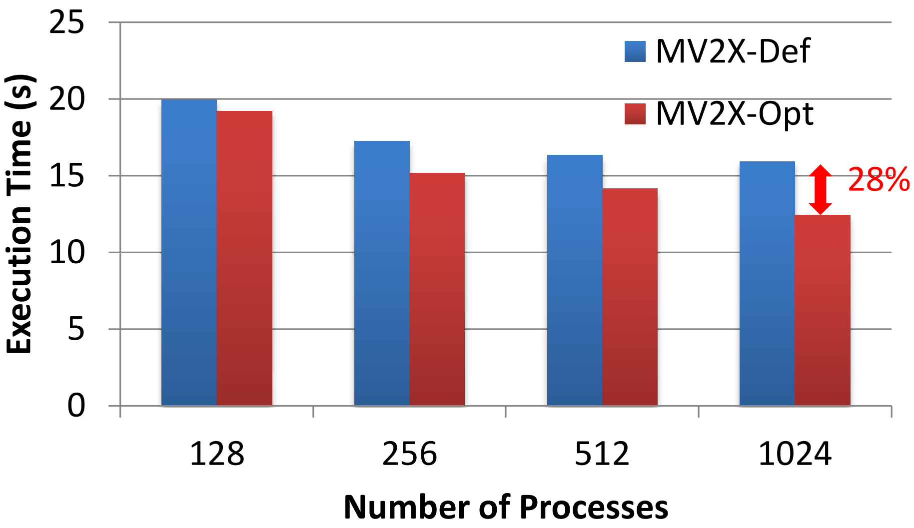
Reduce Latency (1,024 processes)

- Optimized designs for OpenSHMEM collective operations
- Broadcast Latency of 256KB message at 1,024 processes:
  - Default: 5955us, Optimized: 2268us
- Reduce Latency of 256KB message at 1,024 processes:
  - Default: 6581us, Optimized: 3294us

Performance Evaluations using Graph500



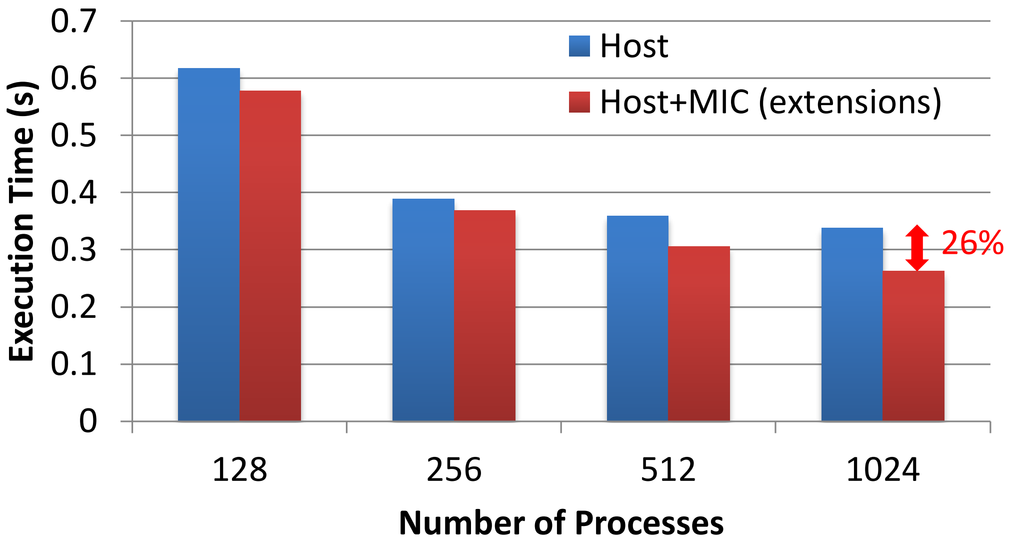
Native Mode (8 procs/MIC)



Symmetric Mode (16 Host+16MIC)

- Graph500 Execution Time (Native Mode):
  - At 512 processes , Default: 5.17s, Optimized: 4.96s
  - Performance Improvement from MIC-aware collectives design
- Graph500 Execution Time (Symmetric Mode):
  - 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:
  - 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

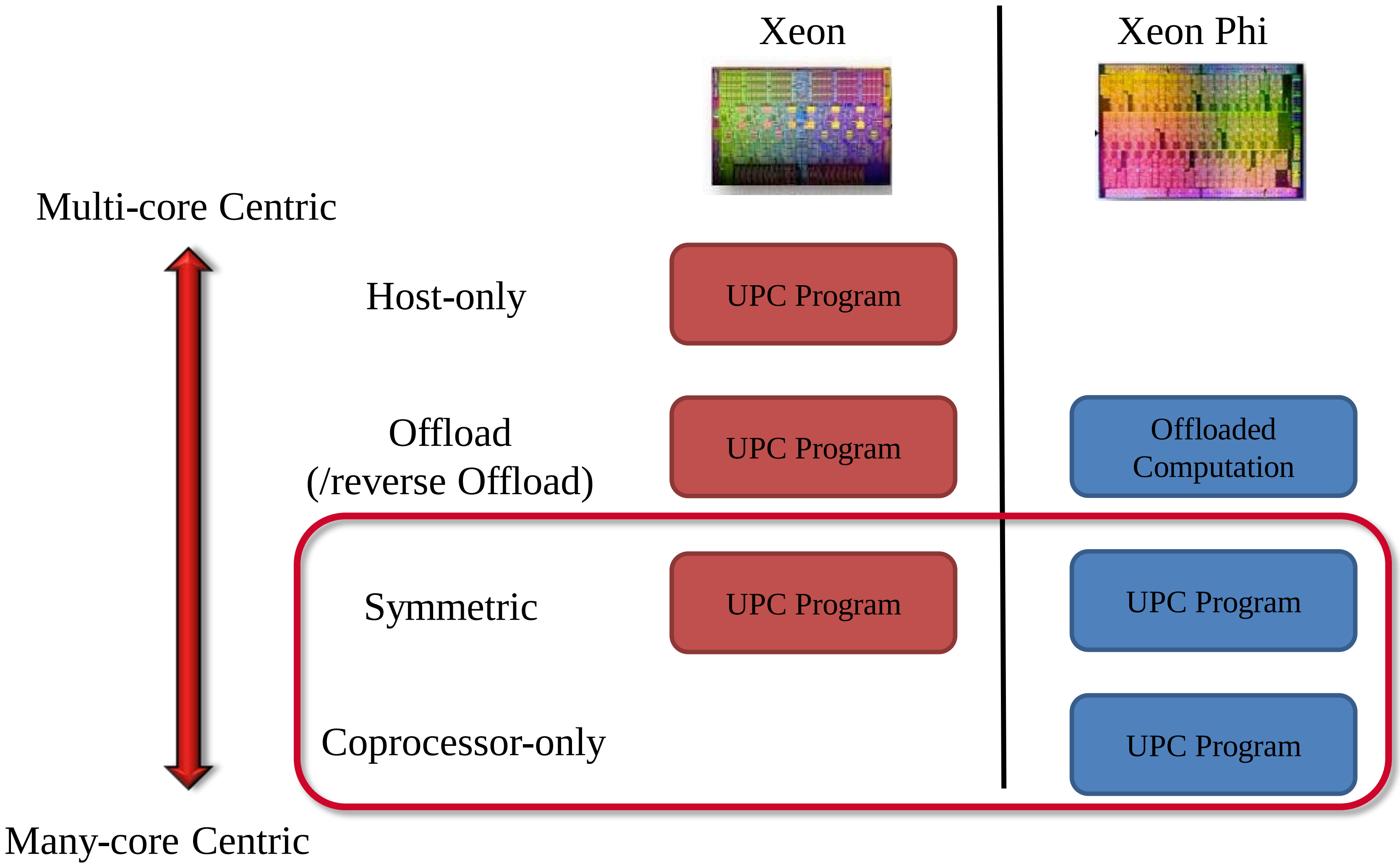
J. Jose, K. Hamidouche, X. Lu, S. Potluri, J. Zhang, K. Tomko and D. K. Panda, High Performance OpenSHMEM for Intel MIC Clusters: Extensions, Runtime Designs and Application Co-Design, IEEE International Conference on Cluster Computing (CLUSTER '14) (Best Paper Nominee)

Outline

- OpenSHMEM and UPC for Host
- Hybrid MPI + PGAS (OpenSHMEM and UPC) for Host
- OpenSHMEM for MIC
- UPC for MIC

HPC Applications on MIC Clusters

- Flexibility in launching HPC jobs on Xeon Phi Clusters



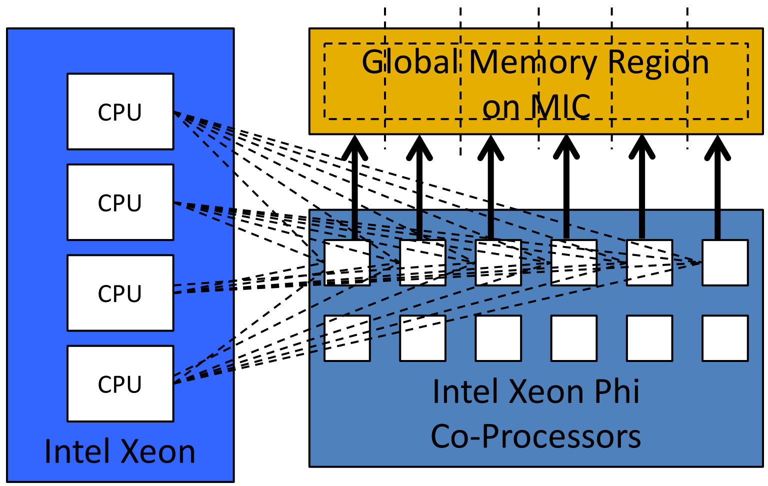
UPC on MIC clusters

- Process Model
  - Communication via shared memory – single copy/double copy
  - One connection per process
  - Drawbacks: Communication Overheads, Network Connections
- Thread Model
  - Direct access to shared array: same memory space
  - No extra copy
  - No shared memory mapping entries
  - Drawback: sharing of network connections
  - Our work on multi-endpoint addresses this issue

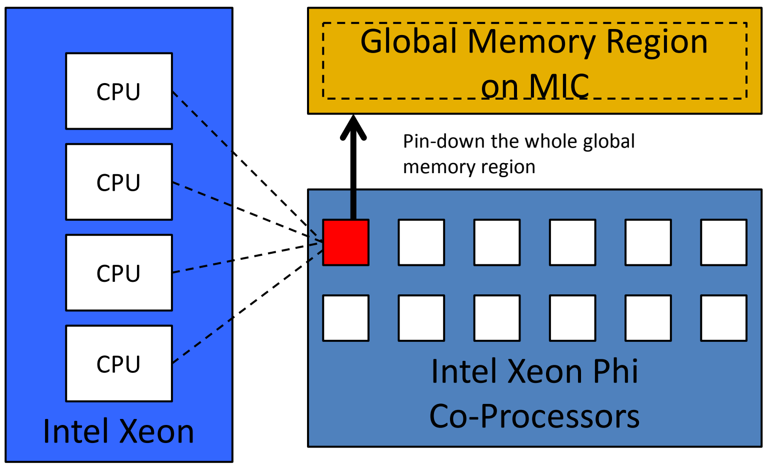
M. Luo, J. Jose, S. Sur and D. K. Panda, Multi-threaded UPC Runtime with Network Endpoints: Design Alternatives and Evaluation on Multi-core Architectures, Int'l Conference on High Performance Computing (HiPC '11), Dec. 2011

UPC Runtime on MIC:
Remote Memory Access between Host and MIC

Original all-to-all connection mode

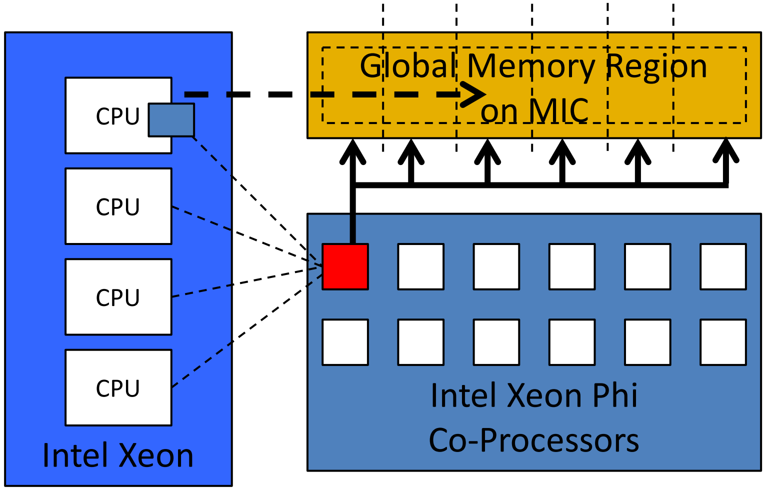


Leader-to-all connection mode



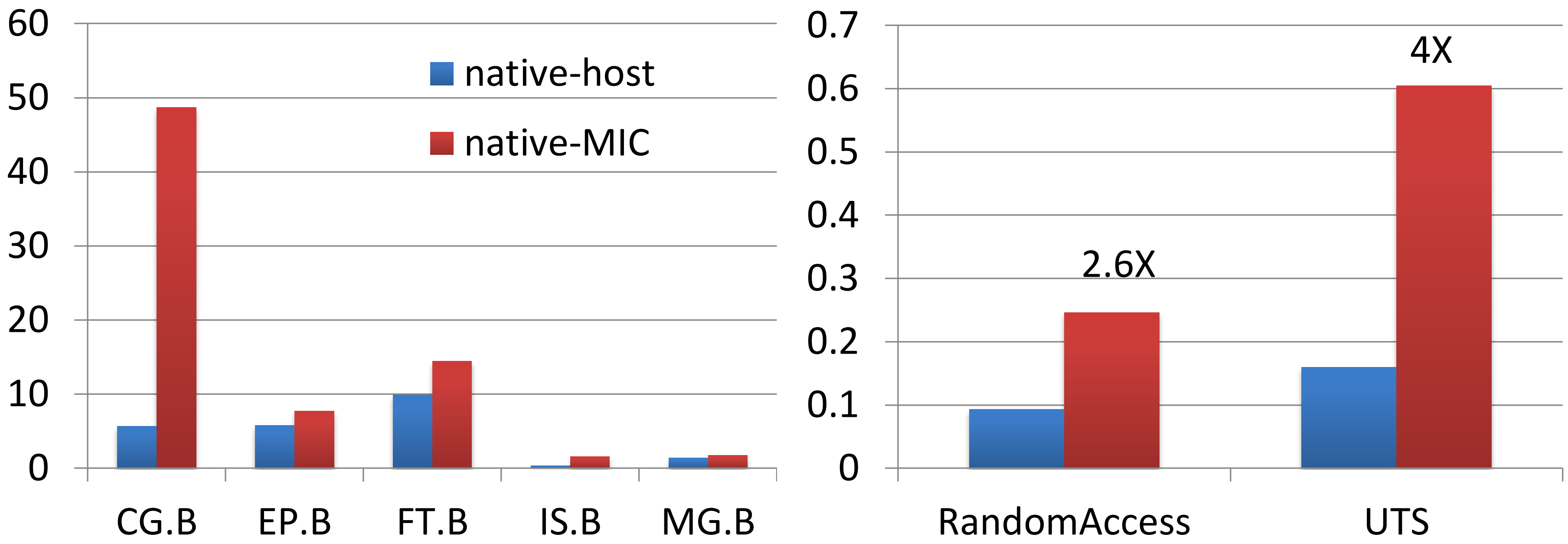
- Original connection mode: Each UPC thread pins down the global memory region that has affinity with this UPC thread; Establish all-to-all connections between every pair of UPC threads between MIC and host
- Leader-to-all connection mode: Only one UPC thread pinned down the whole global memory space on MIC; all UPC threads on the host will get access with the leader UPC thread
- Connections: $N_{MIC} * N_{HOST}$ -> $N_{MIC} + N_{HOST}$

UPC Runtime on MIC: Remote Memory Access between Host and MIC



- Process-based Runtime: The whole global memory region need to be mapped as shared memory region, such as PSHM
- Thread-based Runtime: As the threads share the same memory space, the leader can access other global memory region on MIC without mapping to shared memory

Application Benchmark Evaluation for Native Mode



- Host and MIC are occupied with 16 and 60 UPC threads, respectively
- MIC performs 80%, 67%, and 54% as 16-CPU host for MG, EP, and FT, respectively
- For communication-intensive benchmarks CG and IS, MIC spends 7x and 3x execution times
- Default applications without modification – no representation of optimal performance

M. Luo, M. Li, A. Venkatesh, X. Lu and D. K. Panda, UPC on MIC: Early Experiences with Native and Symmetric Modes, Int'l Conference on Partitioned Global Address Space Programming Models (PGAS '13), October 2013.

Concluding Remarks

- HPC Systems are evolving to support growing needs of exascale applications
- Hybrid programming (MPI + PGAS) is a practical way to program exascale systems
- Presented designs to demonstrate performance potential of PGAS and hybrid MPI+PGAS models
- MIC support for OpenSHMEM and UPC will be available in future MVAPICH2-X releases

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.)
- 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 Associates

- H. Subramoni
- X. Lu

Current Post-Docs

- K. Hamidouche
- J. Lin

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.)
- S. Potluri (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 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

Web Pointers

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

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

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

MVAPICH Web Page

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

MVAPICH

panda@cse.ohio-state.edu

hamidouc@cse.ohio-state.edu