Support Hybrid MPI+PGAS (UPC/OpenSHMEM/CAF) Programming Models through a Unified Runtime: An MVAPICH2-X Approach

Talk at OSC theater (SC '15)

by

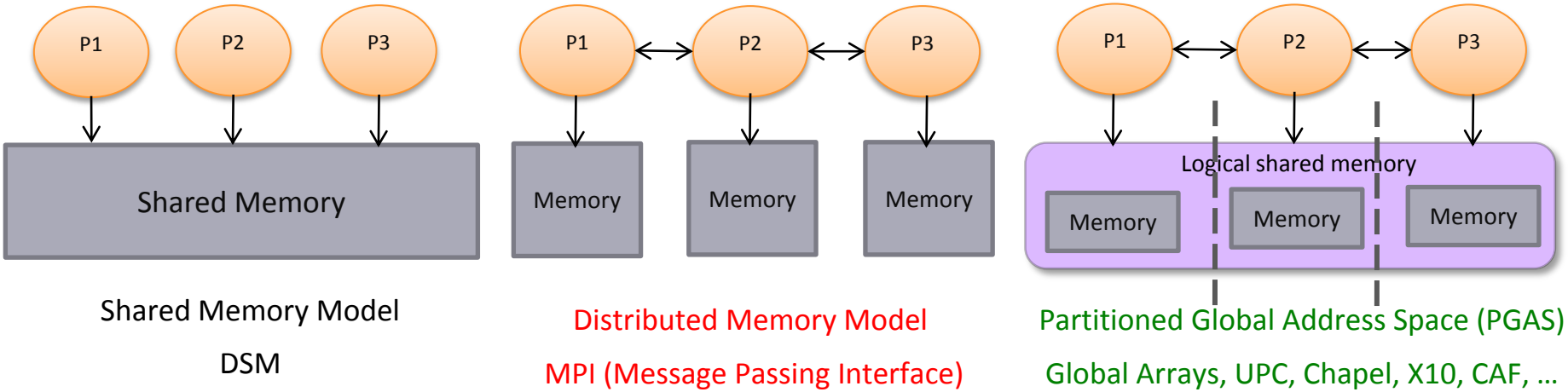
Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

Parallel Programming Models Overview



- 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

OpenSHMEM: PGAS library

- 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

Unified Parallel C: PGAS language

- 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, …

Co-array Fortran (CAF): Language-level PGAS support in Fortran

- An extension to Fortran to support global shared array in parallel Fortran applications
- CAF = CAF compiler + CAF runtime (libcaf)

- Coarray syntax and basic synchronization support in Fortran 2008

E.g.

```
real :: a(n)[*]
x(:)[q] = x(:) + x(:)[p]
name[i] = name
sync all
...
```

- Collective communication and atomic operations in upcoming Fortran 2015

interface	parameters
CO_BROADCAST	A, SOURCE_IMAGE [, STAT, ERRMSG]
CO_MAX	A [, RESULT_IMAGE, STAT, ERRMSG]
CO_MIN	A [, RESULT_IMAGE, STAT, ERRMSG]
CO_SUM	A [, RESULT_IMAGE, STAT, ERRMSG]
CO_REDUCE	A, OPERATOR [, RESULT_IMAGE, STAT, ERRMSG]

```
ATOMIC_ADD
ATOMIC_FETCH_ADD
...
```

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

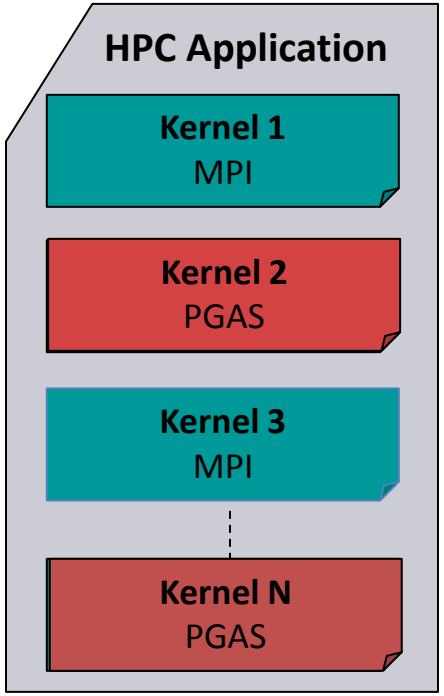
- Re-writing complete applications can be a huge effort
- Port critical kernels to the desired model instead

MVAPICH2 Software

- High Performance open-source MPI Library for InfiniBand, 10-40Gig/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 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
  - **Used by more than 2,475 organizations in 76 countries**
  - **More than 308,000 downloads from the OSU site directly**
  - Empowering many TOP500 clusters (June '15 ranking)
    - 10th ranked 519,640-core cluster (Stampede) at TACC
    - 13th ranked 185,344-core cluster (Pleiades) at NASA
    - 25th 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 (8th in Jun'15, 519,640 cores, 5.168 Plops)

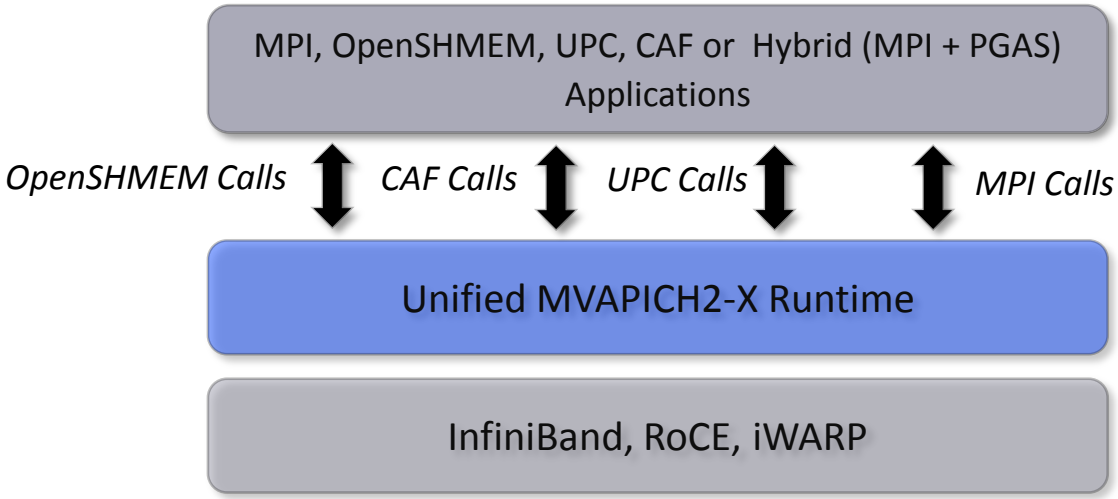
Hybrid (MPI+PGAS) Programming

- 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"



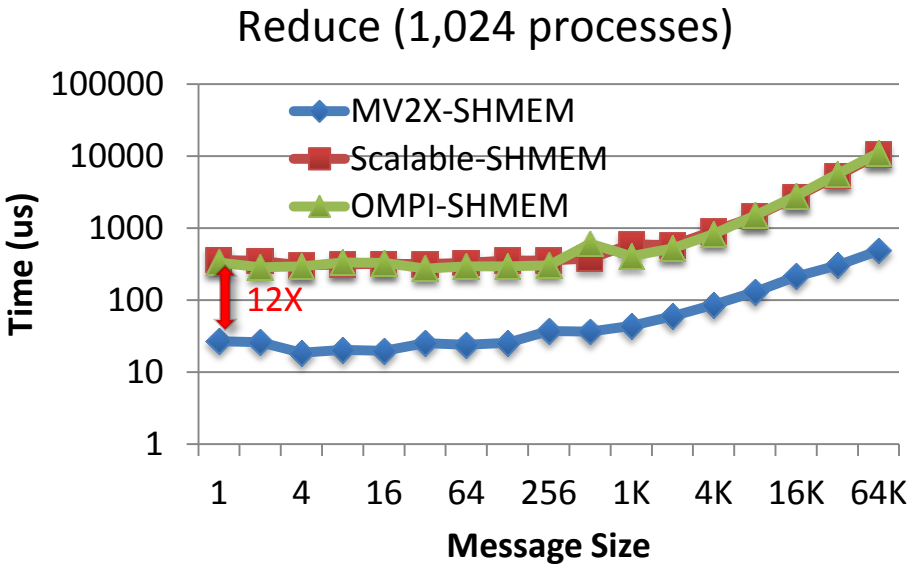
** The International Exascale Software Roadmap, Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420*

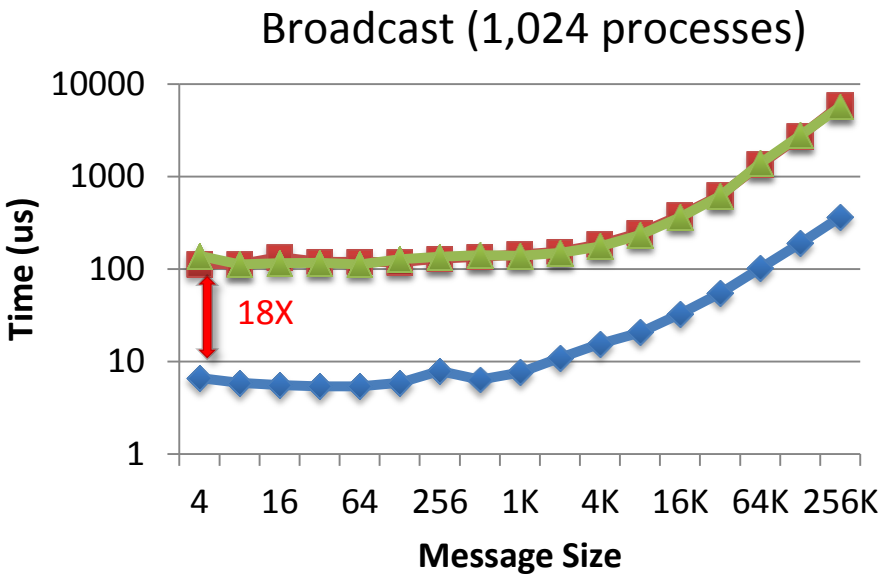
MVAPICH2-X for Hybrid MPI + PGAS Applications

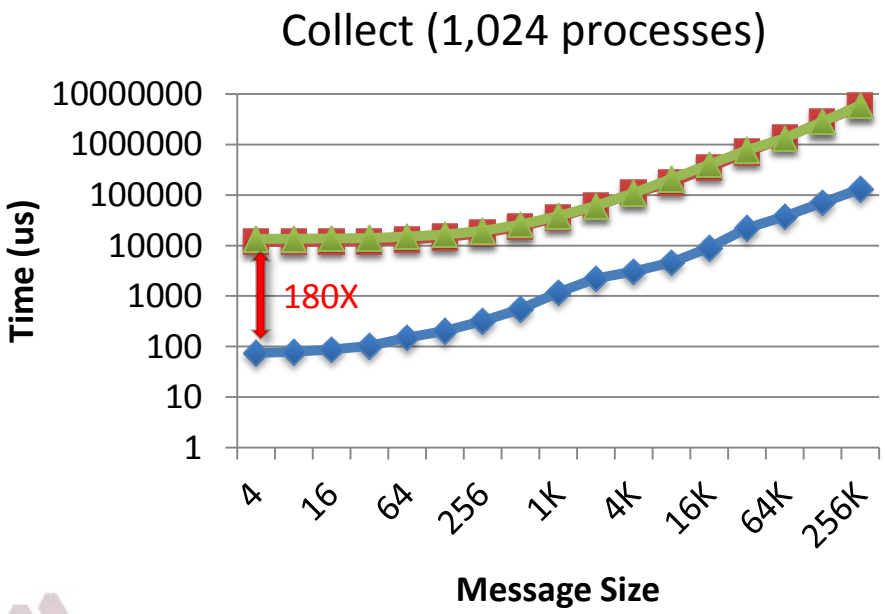


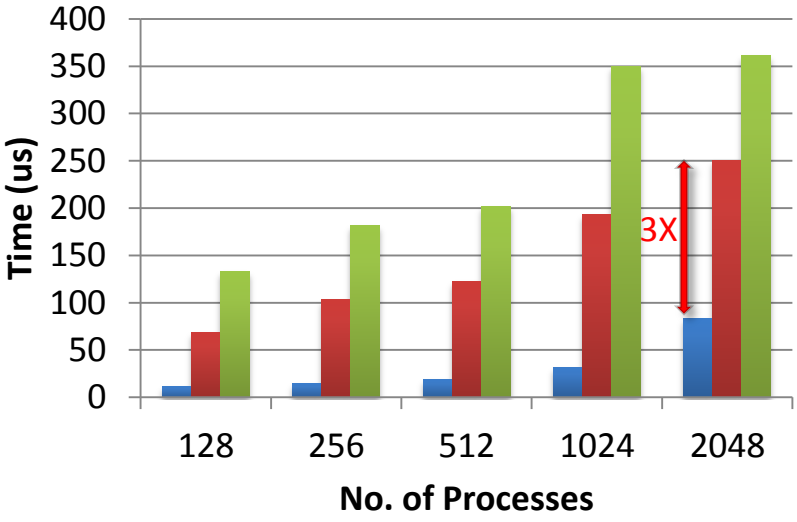
- Unified communication runtime for MPI, UPC, OpenSHMEM, CAF available with MVAPICH2-X 1.9 (2012) onwards!
  - http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard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

OpenSHMEM Collective Communication: Performance

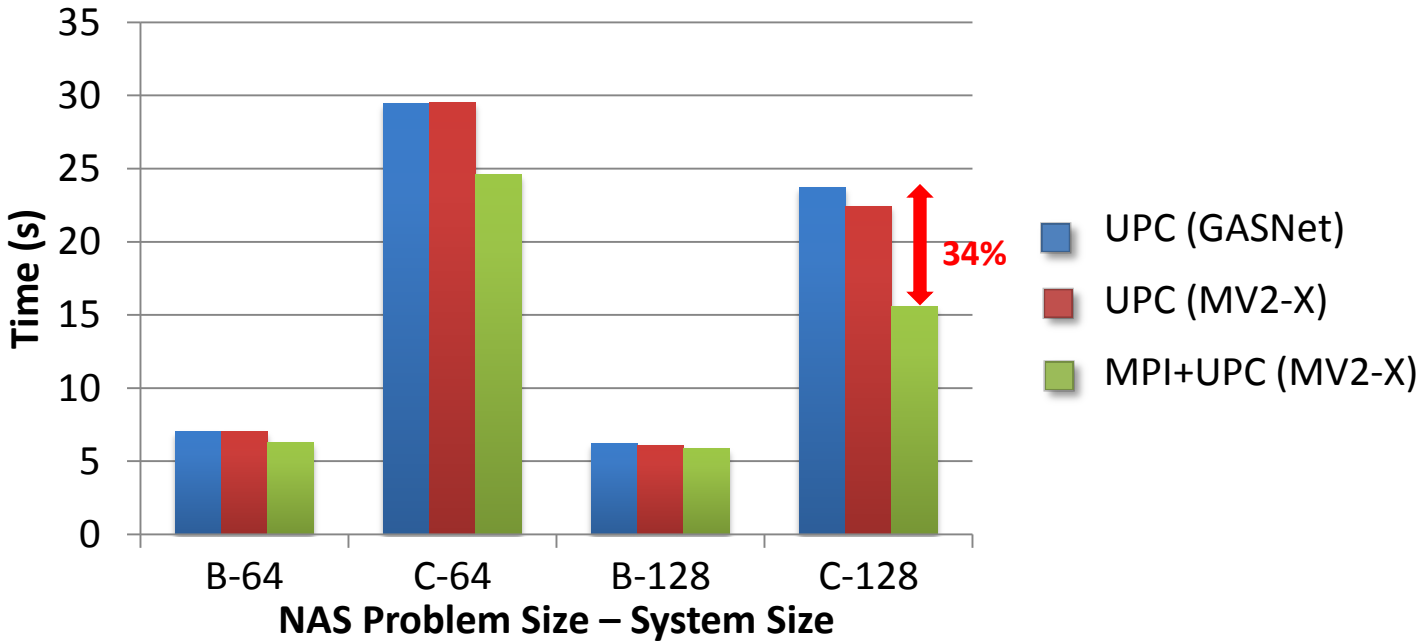








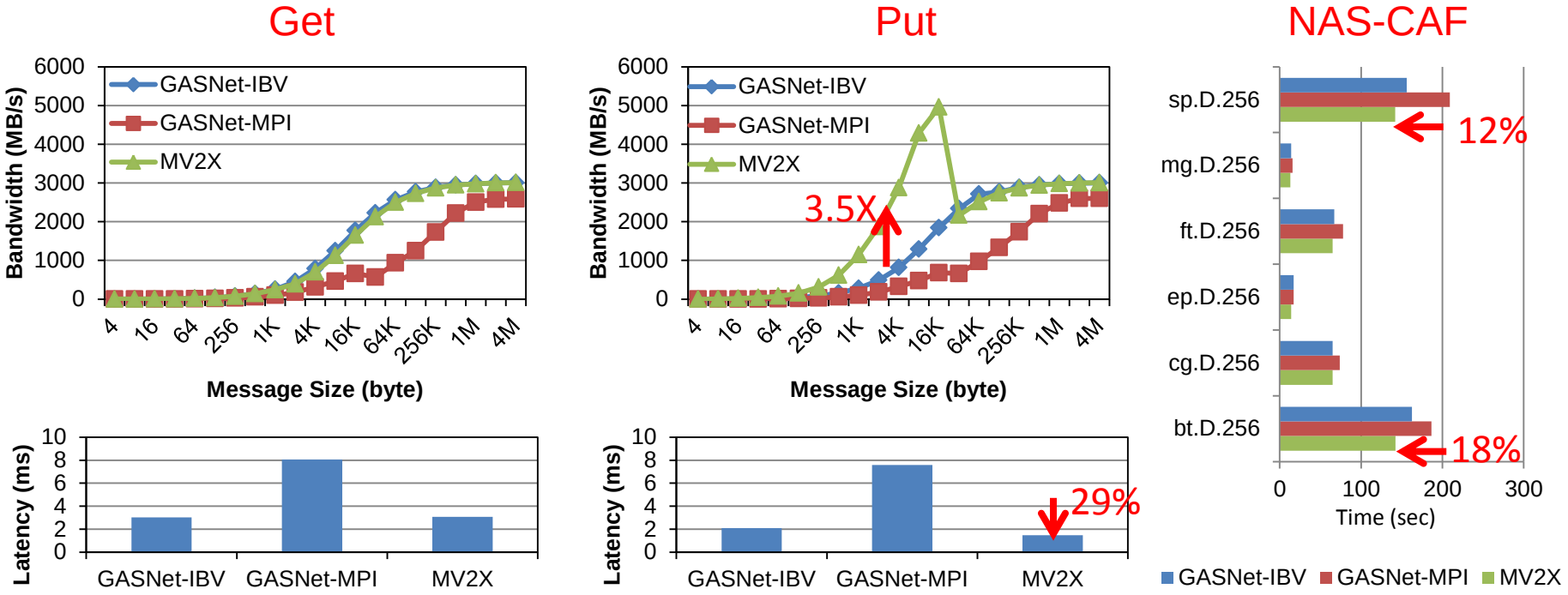
Hybrid MPI+UPC NAS-FT



- 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, PGAS 2010

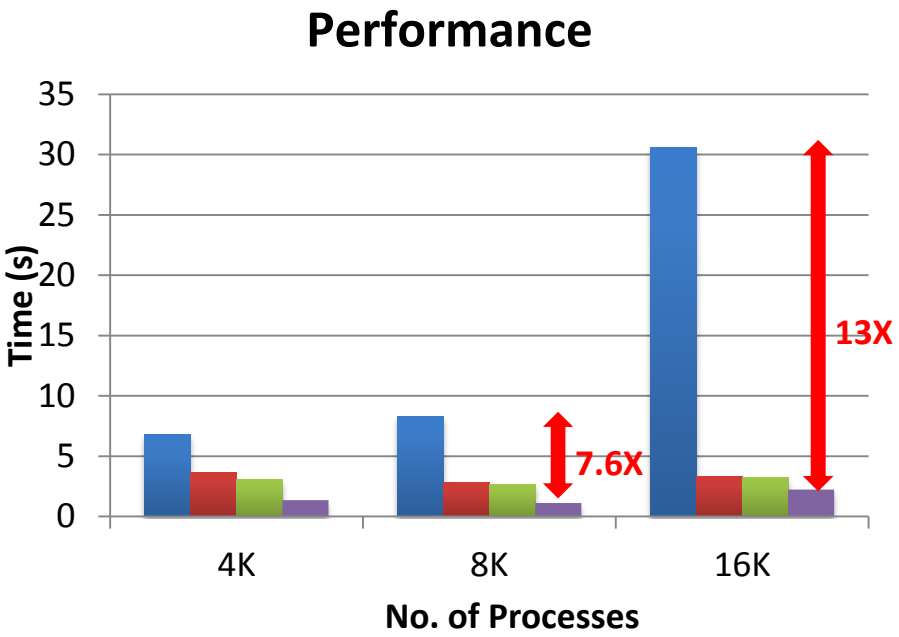
CAF One-sided Communication: Performance

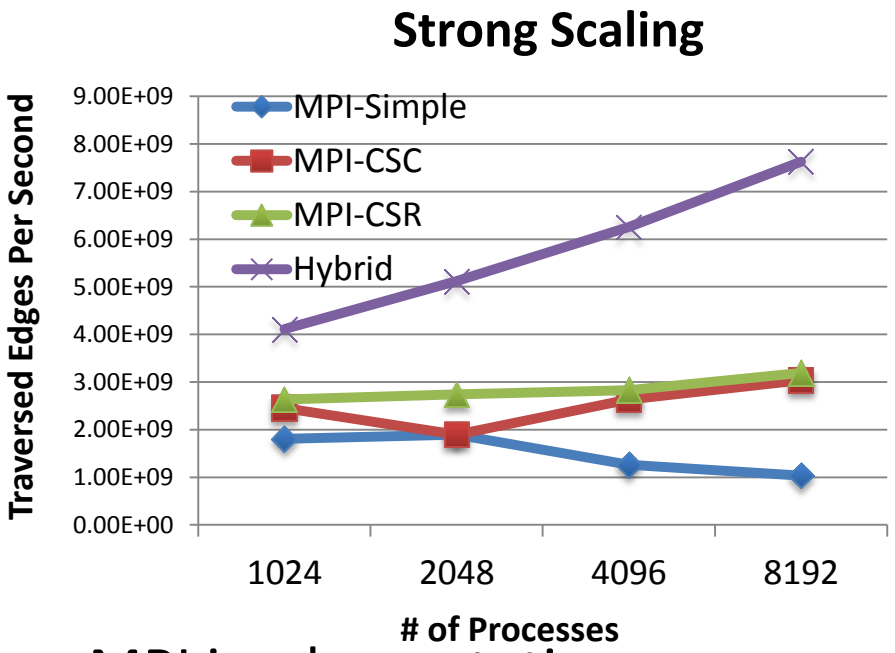


- Micro-benchmark improvement (MV2X vs. GASNet-IBV, UH CAF test-suite)
  - Put bandwidth: 3.5X improvement on 4KB; Put latency: reduce 29% on 4B
- Application performance improvement (NAS-CAF one-sided implementation)
  - Reduce the execution time by 12% (SP.D.256), 18% (BT.D.256)

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

Graph500 - BFS Traversal Time

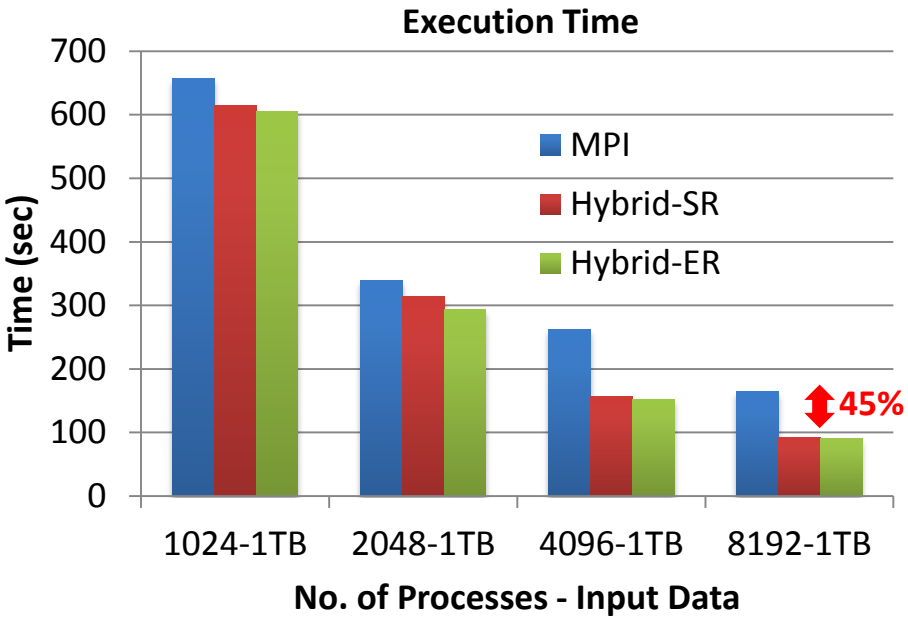


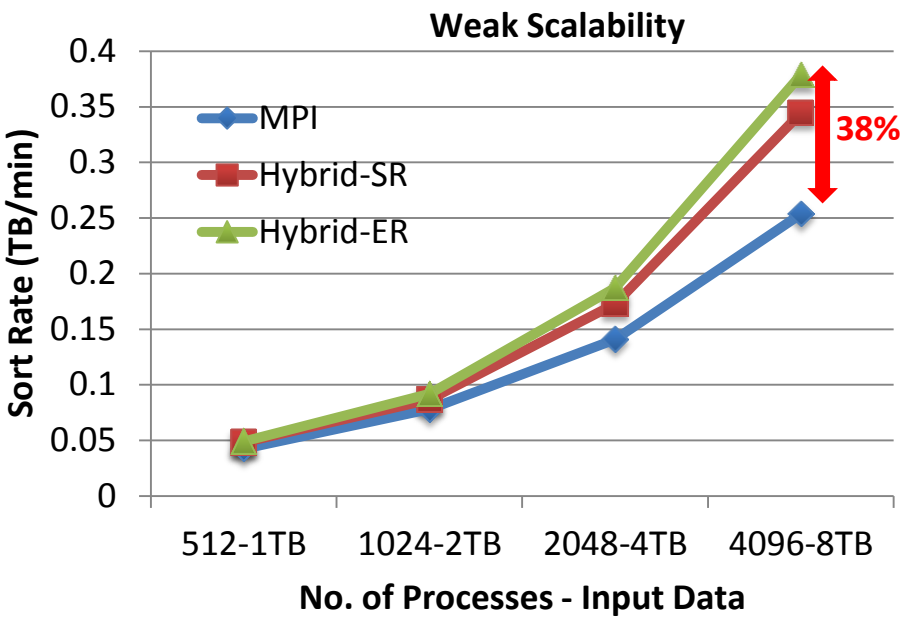


- Hybrid design performs better than MPI implementations
- 16,384 processes
  - 1.5X improvement over MPI-CSR
  - 13X improvement over MPI-Simple (Same communication characteristics)
- Strong Scaling

  Graph500 Problem Scale = 29

Hybrid MPI+OpenSHMEM Sort Application



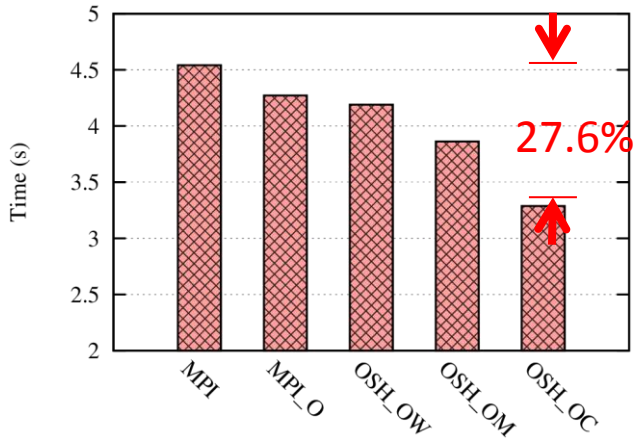


- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - Execution Time (seconds)
    - 1TB Input size at 8,192 cores: MPI – 164, Hybrid-SR (Simple Read) – 92.5, Hybrid-ER (Eager Read) - 90.36
    - 45% improvement over MPI-based design
  - Weak Scalability (configuration: input size of 1TB per 512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

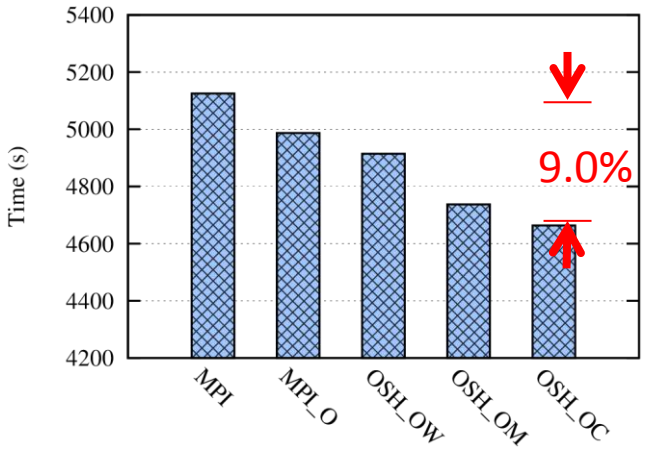
Accelerating MaTEx k-NN with Hybrid MPI and OpenSHMEM

- 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

KDD-tranc (30MB) on 256 cores



KDD (2.5GB) on 512 cores

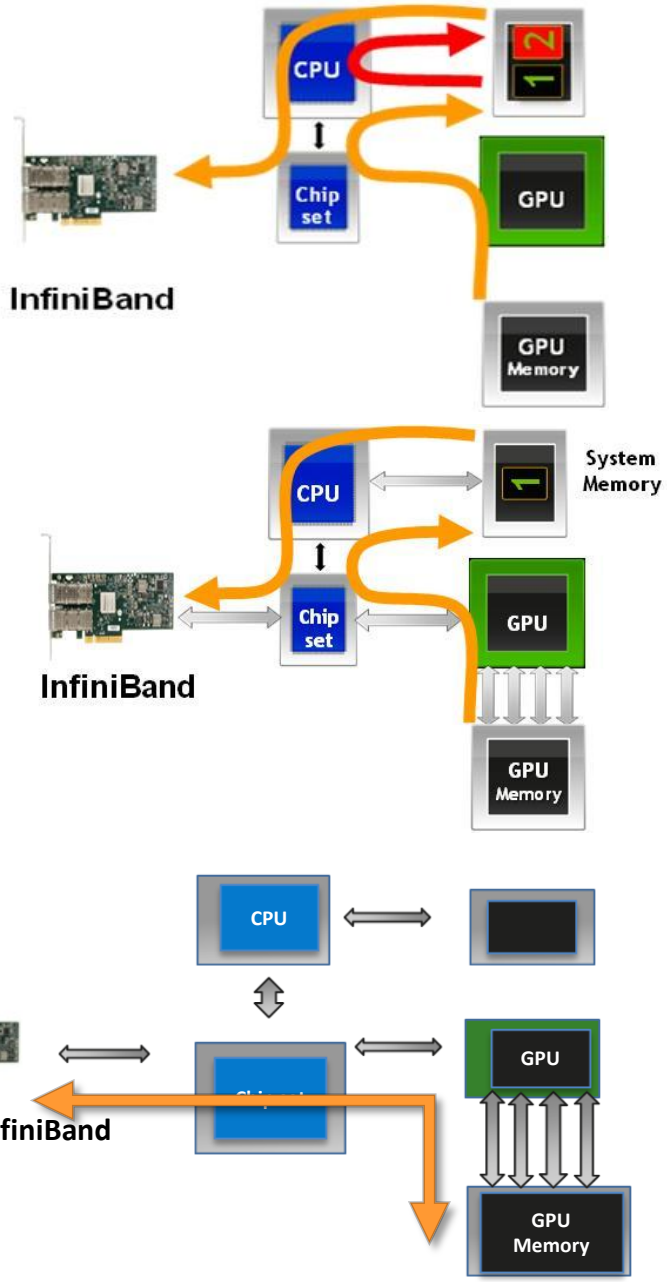


- Benchmark: KDD Cup 2010 (8,407,752 records, 2 classes, k=5)
- For truncated KDD workload on 256 cores, reduce 27.6% execution time
- For full KDD workload on 512 cores, reduce 9.0% execution time

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

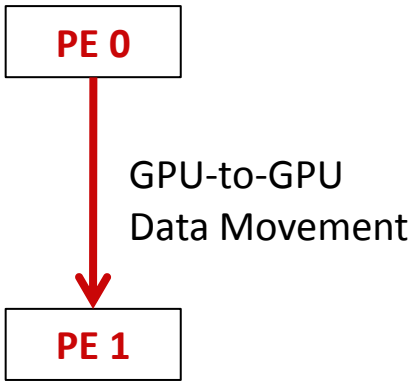
CUDA-Aware Concept

- Before CUDA 4: Additional copies
  - Low performance and low productivity
- After CUDA 4: Host-based pipeline
  - Unified Virtual Address
  - Pipeline CUDA copies with IB transfers
  - High performance and high productivity
- After CUDA 5.5: GPUDirect-RDMA support
  - GPU to GPU direct transfer
  - Bypass the host memory
  - Hybrid design to avoid PCI bottlenecks



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 = shmalloc (…)

cudaMemcpy (host_buf, dev_buf, . . .)

shmem_putmem (host_buf, host_buf, size, pe)

shmem_barrier (…)

PE 1

host_buf = shmalloc (…)

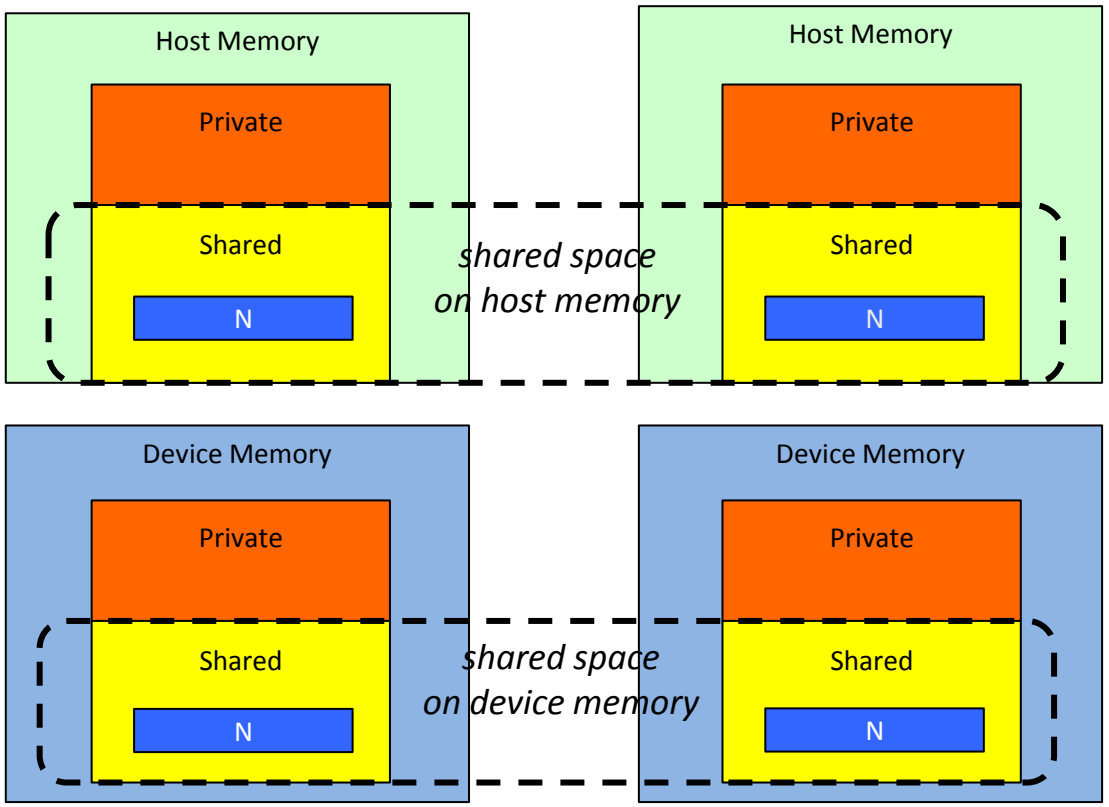
shmem_barrier (…)

cudaMemcpy (host_buf, dev_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);

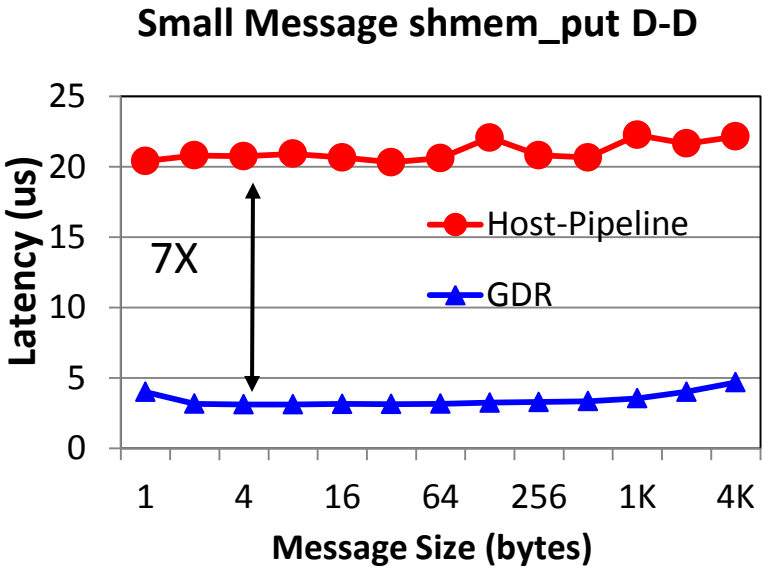
S. Potluri, D. Bureddy, H. Wang, H. Subramoni and D. K. Panda, Extending OpenSHMEM for GPU Computing, IPDPS'13

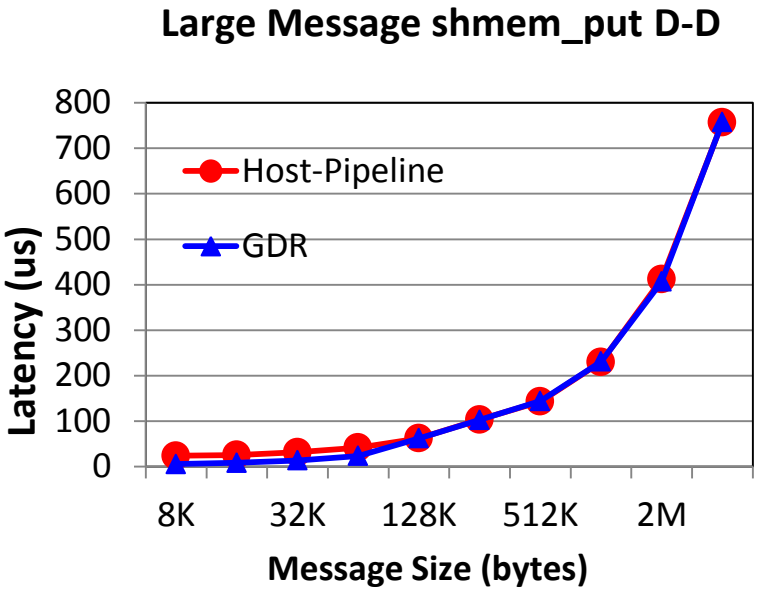
CUDA-aware OpenSHMEM and UPC runtimes

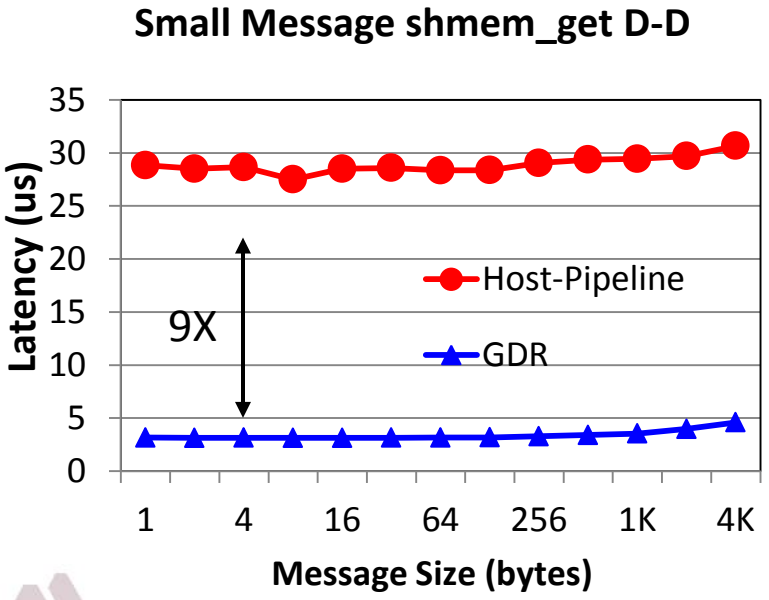
- After device memory becomes part of the global shared space:
  - Accessible through standard UPC/OpenSHMEM communication APIs
  - Data movement transparently handled by the runtime
  - Preserves one-sided semantics at the application level
- Efficient designs to handle communication
  - Inter-node 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

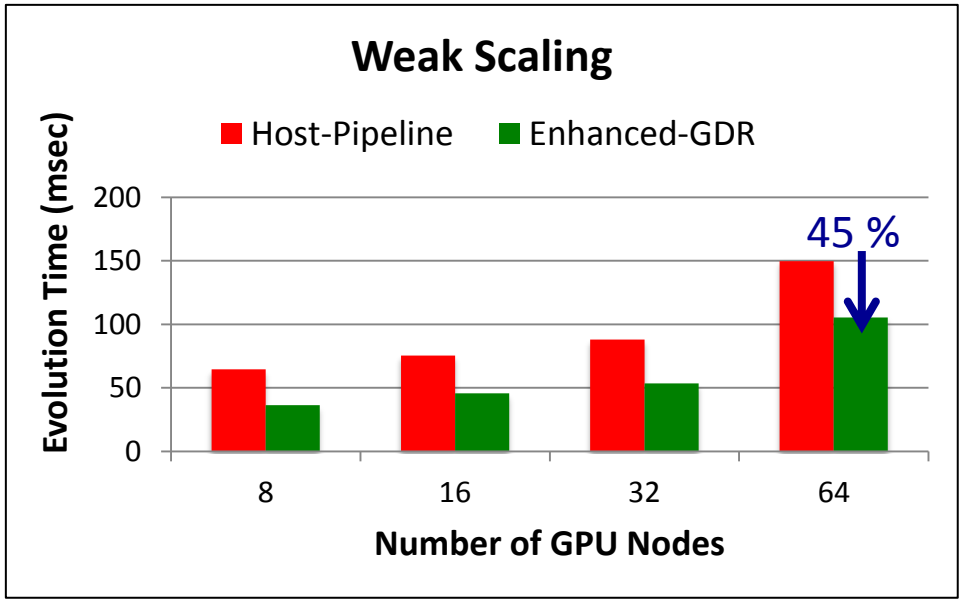




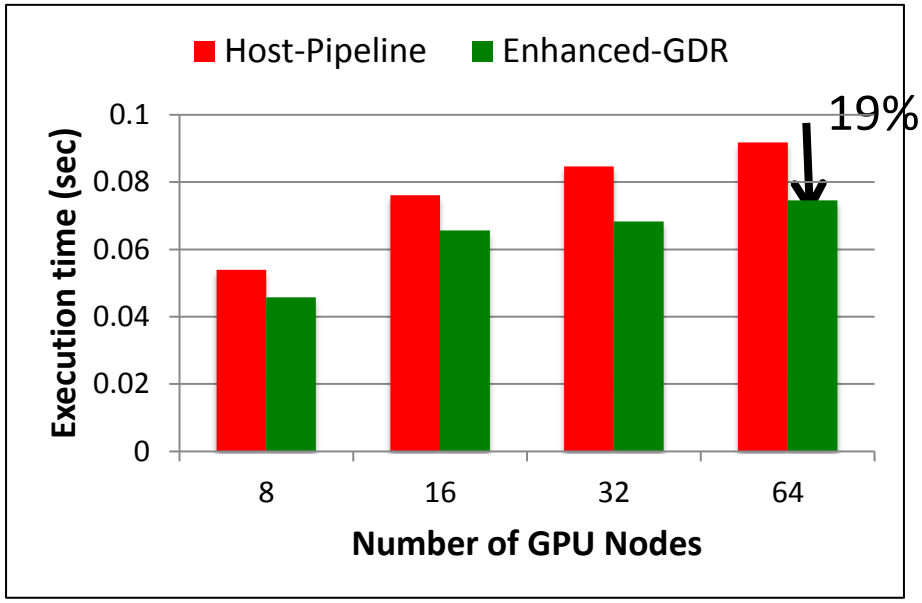


- GDR for small/medium message sizes
- Host-staging for large message to avoid PCIe bottlenecks
- Hybrid design brings best of both
- 3.13 us Put latency for 4B (7X improvement) and 4.7 us latency for 4KB
- 9X improvement for Get latency of 4B

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

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

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

Concluding Remarks

- Presented an overview of a Hybrid MPI+PGAS models
- Outlined research challenges in designing efficient runtimes for these models on clusters with InfiniBand
- Demonstrated the benefits of Hybrid MPI+PGAS models for a set of applications on Host based systems
- Presented the model extension and runtime support for Accelerator-Aware hybrid MPI+PGAS model
- Hybrid MPI+PGAS model is an emerging paradigm which can lead to high-performance and scalable implementation of applications on exascale computing systems using accelerators

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

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

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.)
- 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.)
- 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.)

Past Post-Docs

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

Current Research Scientists

- H. Subramoni
- X. Lu

Current Post-Docs

- J. Lin
- D. Shankar

Past Research Scientist

- S. Sur

Current Senior Research Associate

- K. Hamidouche

Current Programmer

- J. Perkins

Current Research Specialist

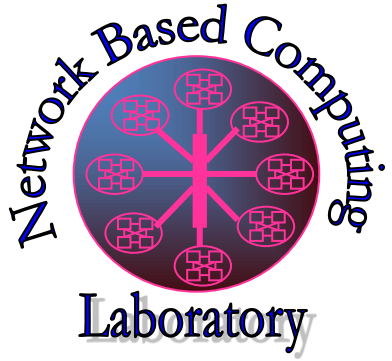
- M. Arnold
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Programmers

- D. Bureddy

Thank You!

panda@cse.ohio-state.edu



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

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/