MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Scalability and Performance of MVAPICH2 on OakForest-PACS

Talk at JCAHPC Booth (SC '17)

by

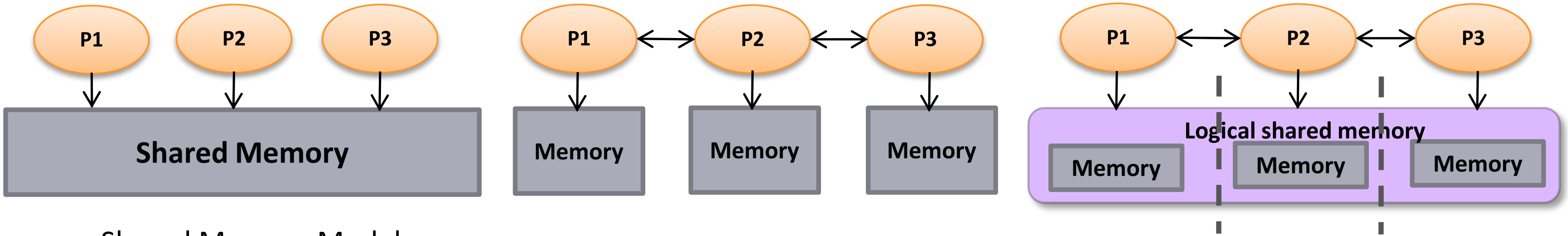
Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

Global Arrays, UPC, Chapel, X10, CAF, …

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

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

Application Kernels/Applications

Middleware

Programming Models

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

Communication Library or Runtime for Programming Models

- Point-to-point Communication
- Collective Communication
- Energy-Awareness
- Synchronization and Locks
- I/O and File Systems
- Fault Tolerance

Networking Technologies (InfiniBand, 40/100GigE, Aries, and OmniPath)

Multi/Many-core Architectures

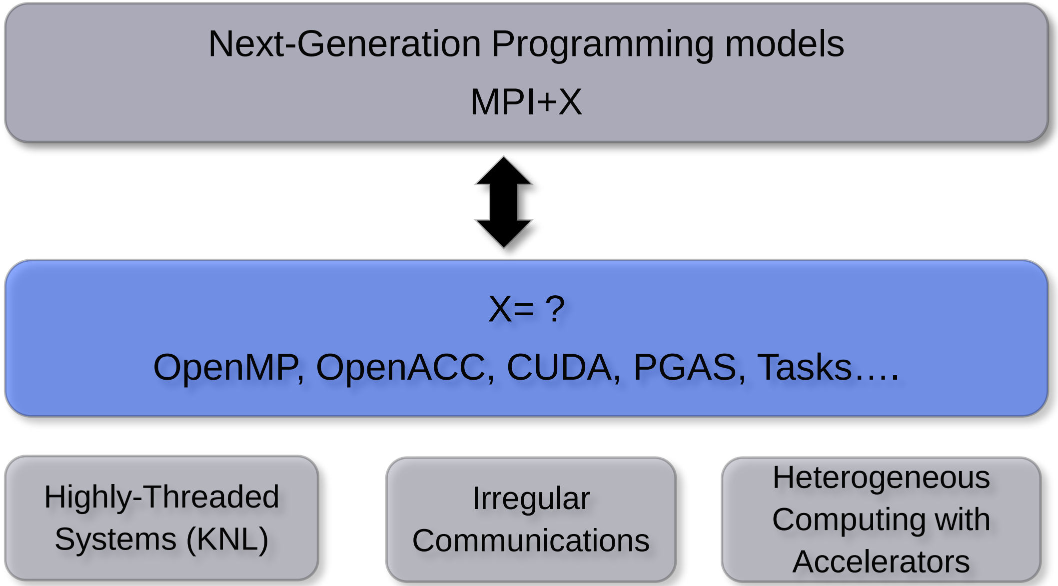
Accelerators (GPU and FPGA)

Co-Design Opportunities and Challenges across Various Layers

- Performance
- Scalability
- Fault-Resilience

Exascale Programming models

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



MPI+X Programming model: Broad Challenges at Exascale

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

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - **Used by more than 2,825 organizations in 85 countries**
  - **More than 432,000 (> 0.4 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (June '17 ranking)
    - **1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China**
    - 15th, 241,108-core (Pleiades) at NASA
    - 20th, 462,462-core (Stampede) at TACC
    - 44th, 74,520-core (Tsubame 2.5) at Tokyo Institute of Technology
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - **http://mvapich.cse.ohio-state.edu**
- Empowering Top500 systems for over a decade
  - System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  - Sunway TaihuLight (1st in Jun'17, 10M cores, 100 PFlops)

MVAPICH2 Software Family

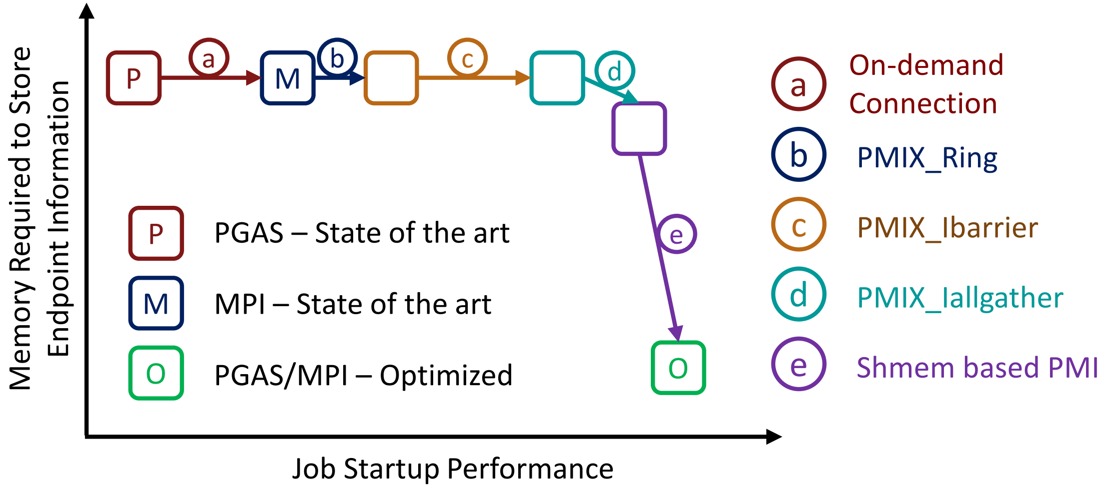
High-Performance Parallel Programming Libraries	
MVAPICH2	Support for InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE
MVAPICH2-X	Advanced MPI features, OSU INAM, PGAS (OpenSHMEM, UPC, UPC++, and CAF), and MPI+PGAS programming models with unified communication runtime
MVAPICH2-GDR	Optimized MPI for clusters with NVIDIA GPUs
MVAPICH2-Virt	High-performance and scalable MPI for hypervisor and container based HPC cloud
MVAPICH2-EA	Energy aware and High-performance MPI
MVAPICH2-MIC	Optimized MPI for clusters with Intel KNC
Microbenchmarks	
OMB	Microbenchmarks suite to evaluate MPI and PGAS (OpenSHMEM, UPC, and UPC++) libraries for CPUs and GPUs
Tools	
OSU INAM	Network monitoring, profiling, and analysis for clusters with MPI and scheduler integration
OEMT	Utility to measure the energy consumption of MPI applications

Outline

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication
  - Scalable Start-up
  - Dynamic and Adaptive Communication Protocols and Tag Matching
  - Optimized Collectives using SHArP and Multi-Leaders
  - Optimized CMA-based Collectives

- Hybrid MPI+PGAS Models for Irregular Applications

Towards High Performance and Scalable 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)

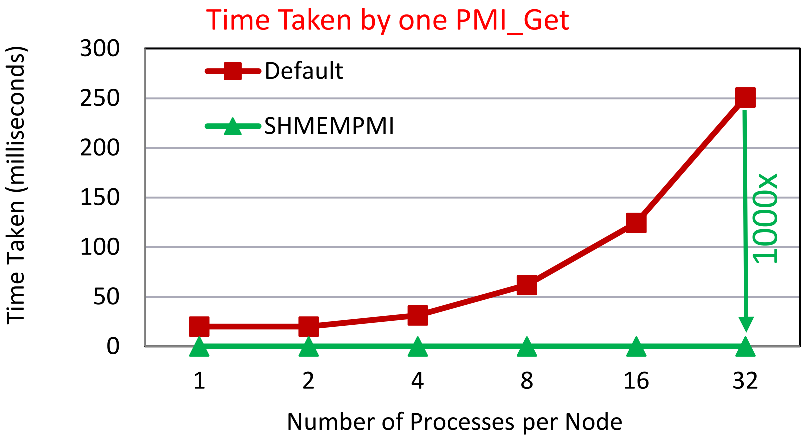
(b) **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia ’14)

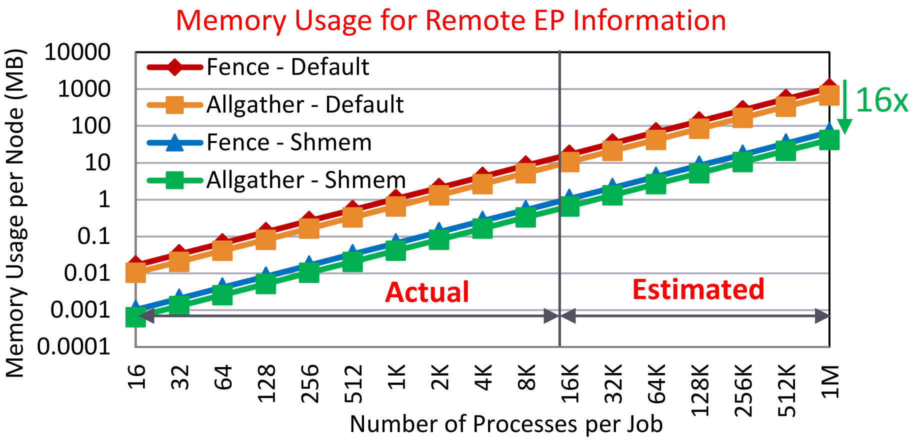
(c) (d) **Non-blocking PMI Extensions for Fast MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’15)

(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)

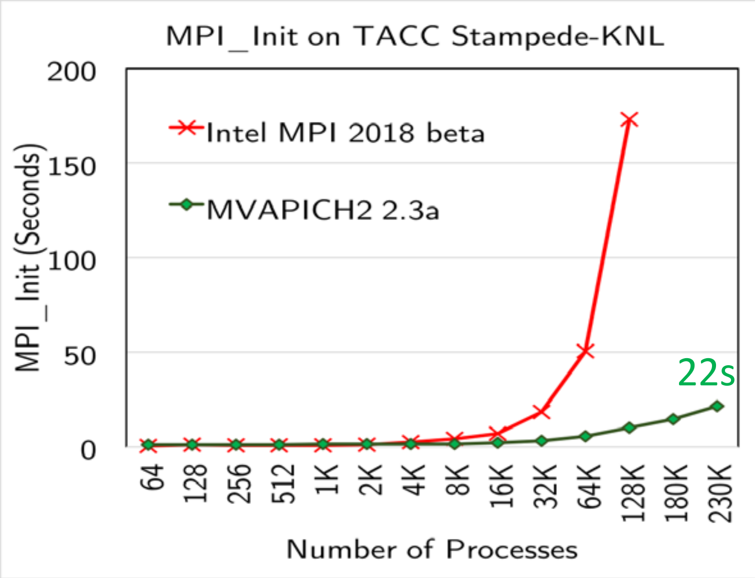
Process Management Interface (PMI) over Shared Memory (SHMEMPMI)

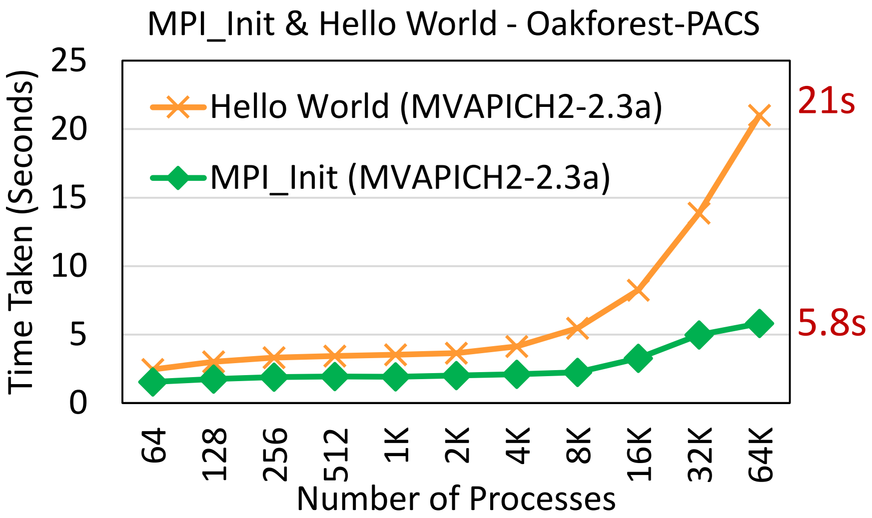
- SHMEMPMI allows MPI processes to directly read remote endpoint (EP) information from the process manager through shared memory segments
- Only a single copy per node - O(processes per node) reduction in memory usage
- Estimated savings of 1GB per node with 1 million processes and 16 processes per node
- Up to 1,000 times faster PMI Gets compared to default design
- **Available for MVAPICH2 2.2rc1 and SLURM-15.08.8**





Startup Performance on KNL + Omni-Path





- MPI_Init takes 22 seconds on 229,376 processes on 3,584 KNL nodes (Stampede2 – Full scale)
- 8.8 times faster than Intel MPI at 128K processes (Courtesy: TACC)
- At 64K processes, MPI_Init and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
- All numbers reported with 64 processes per node

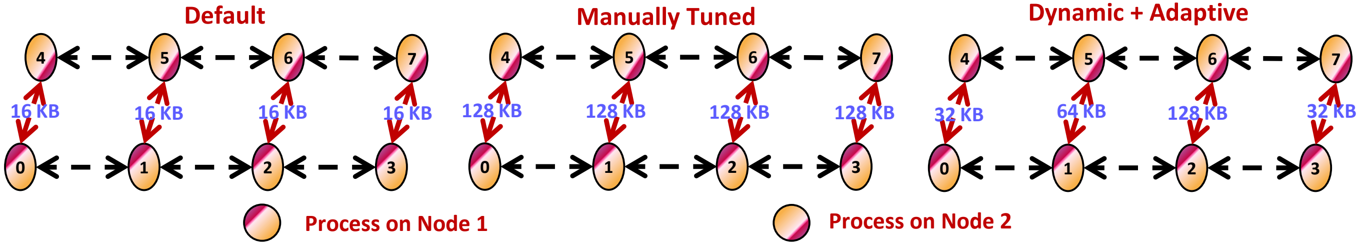
New designs available in latest MVAPICH2 and MVAPICH2-X libraries and as patch for SLURM-15.08.8 and SLURM-16.05.1

Dynamic and Adaptive MPI Point-to-point Communication Protocols

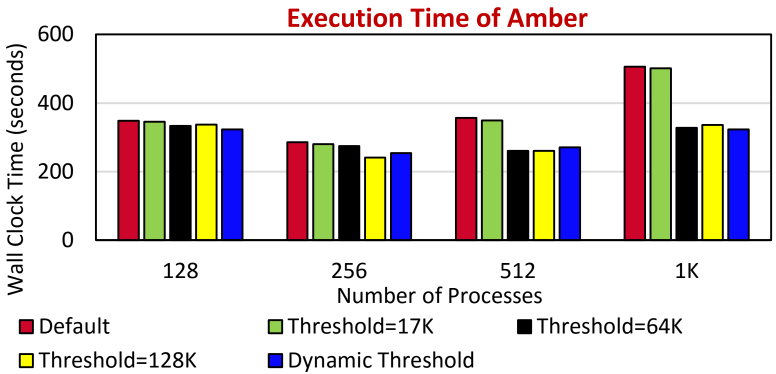
Desired Eager Threshold

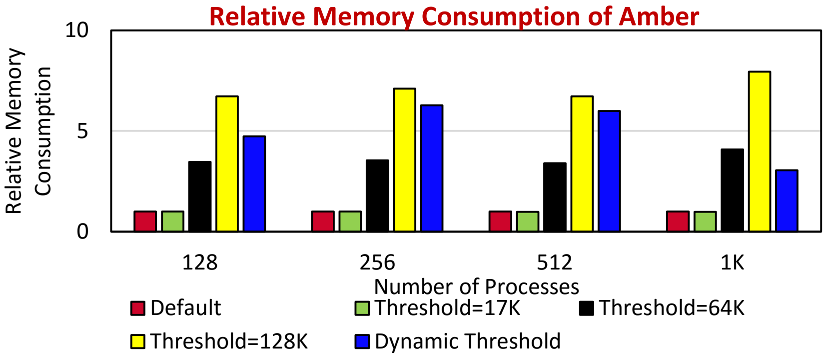
Process Pair	Eager Threshold (KB)
0 – 4	32
1 – 5	64
2 – 6	128
3 – 7	32

Eager Threshold for Example Communication Pattern with Different Designs



Default	Poor overlap; Low memory requirement	Low Performance; High Productivity
Manually Tuned	Good overlap; High memory requirement	High Performance; Low Productivity
Dynamic + Adaptive	Good overlap; Optimal memory requirement	High Performance; High Productivity





H. Subramoni, S. Chakraborty, D. K. Panda, Designing Dynamic & Adaptive MPI Point-to-Point Communication Protocols for Efficient Overlap of Computation & Communication, ISC'17 - Best Paper

Dynamic and Adaptive Tag Matching

Challenge

Tag matching is a significant overhead for receivers

Existing Solutions are

- Static and do not adapt dynamically to communication pattern
- Do not consider memory overhead

Solution

A new tag matching design

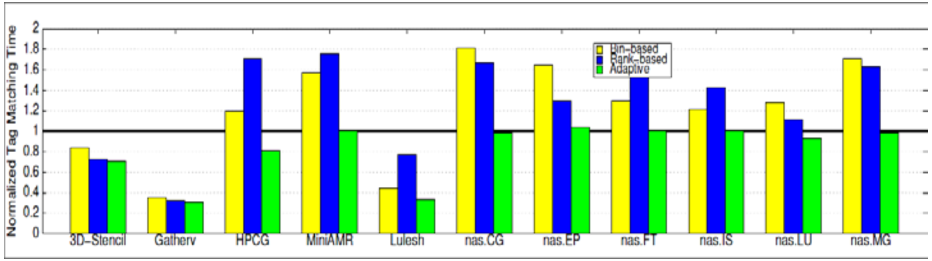
- Dynamically adapt to communication patterns
- Use different strategies for different ranks
- Decisions are based on the number of request object that must be traversed before hitting on the required one

Results

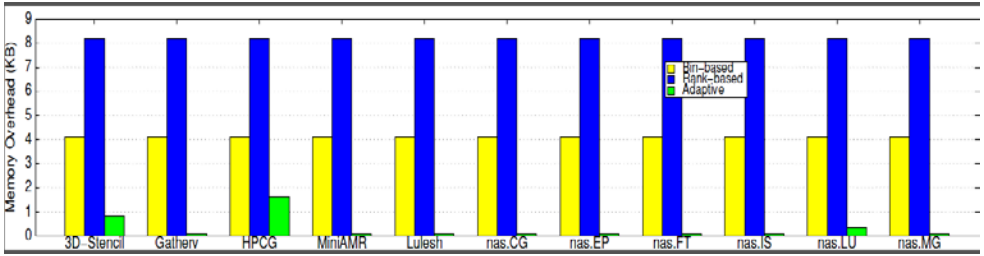
Better performance than other state-of-the art tag-matching schemes

Minimum memory consumption

Will be available in future MVAPICH2 releases



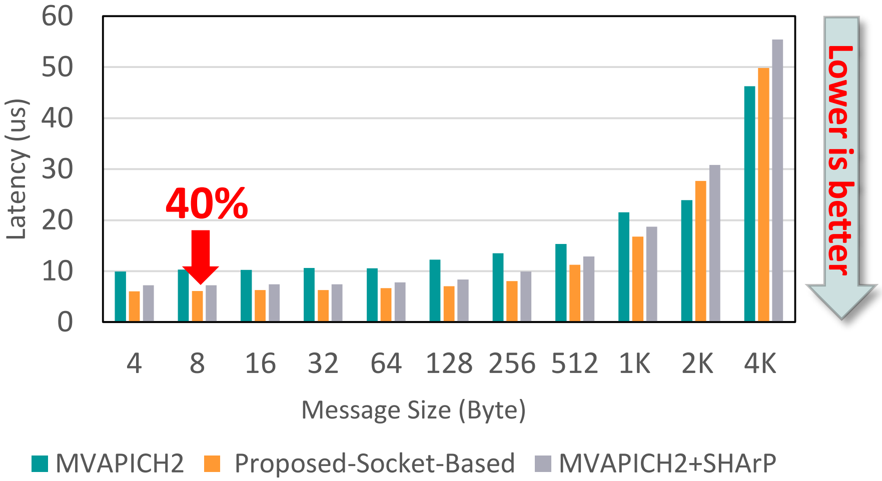
Normalized Total Tag Matching Time at 512 Processes
Normalized to Default (Lower is Better)



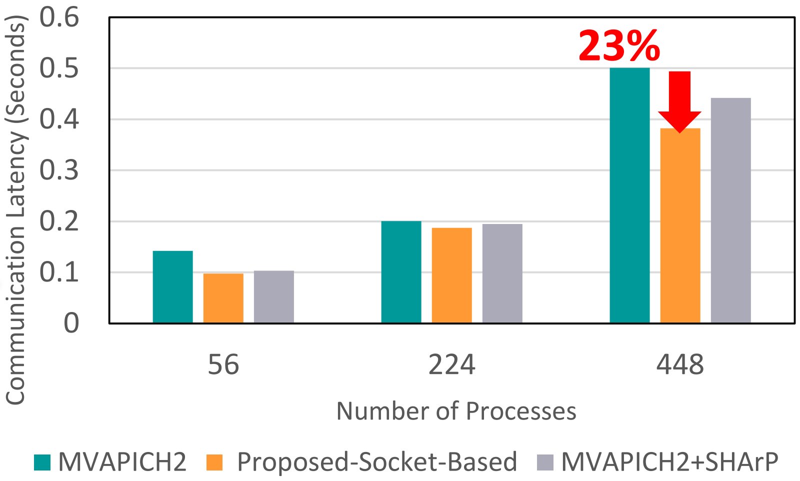
Normalized Memory Overhead per Process at 512 Processes
Compared to Default (Lower is Better)

Adaptive and Dynamic Design for MPI Tag Matching; M. Bayatpour, H. Subramoni, S. Chakraborty, and D. K. Panda; IEEE Cluster 2016. [Best Paper Nominee]

Advanced Allreduce Collective Designs Using SHArP and Multi-Leaders



OSU Micro Benchmark (16 Nodes, 28 PPN)

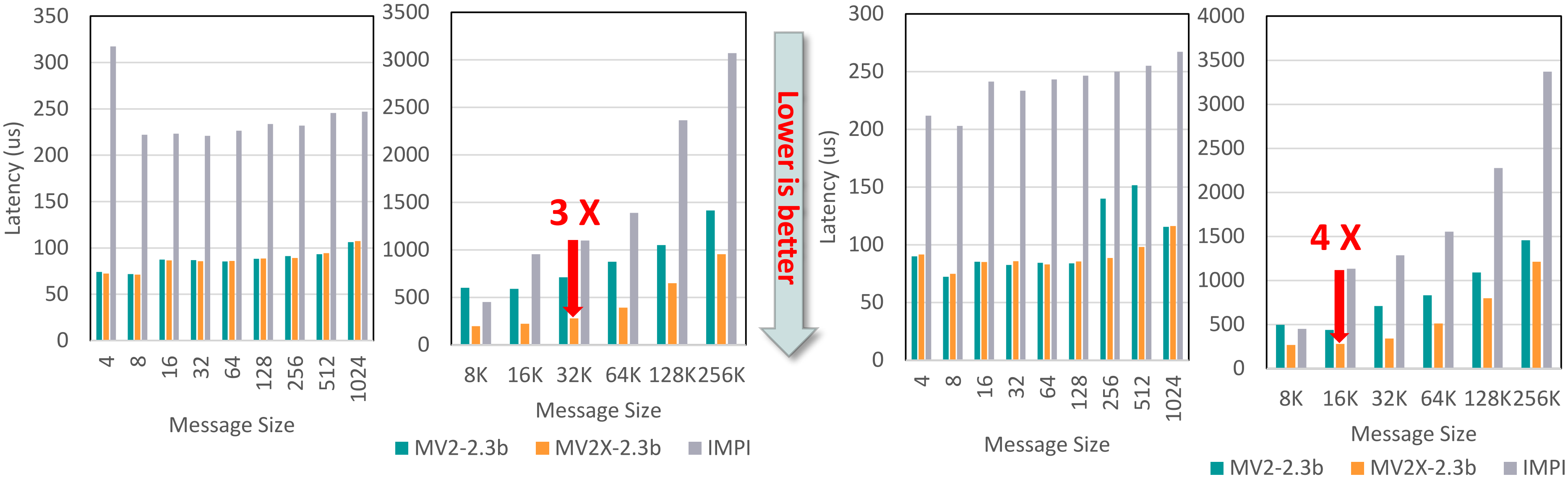


HPCG (28 PPN)

- Socket-based design can reduce the communication latency by 23% and 40% on Xeon + IB nodes
- **Support is available in MVAPICH2 2.3a and MVAPICH2-X 2.3b**

M. Bayatpour, S. Chakraborty, H. Subramoni, X. Lu, and D. K. Panda, Scalable Reduction Collectives with Data Partitioning-based Multi-Leader Design, Supercomputing '17.

Performance of MPI_Allreduce On Oakforest



(64 Nodes, 32 PPN**)

(64 Nodes, 64 PPN)

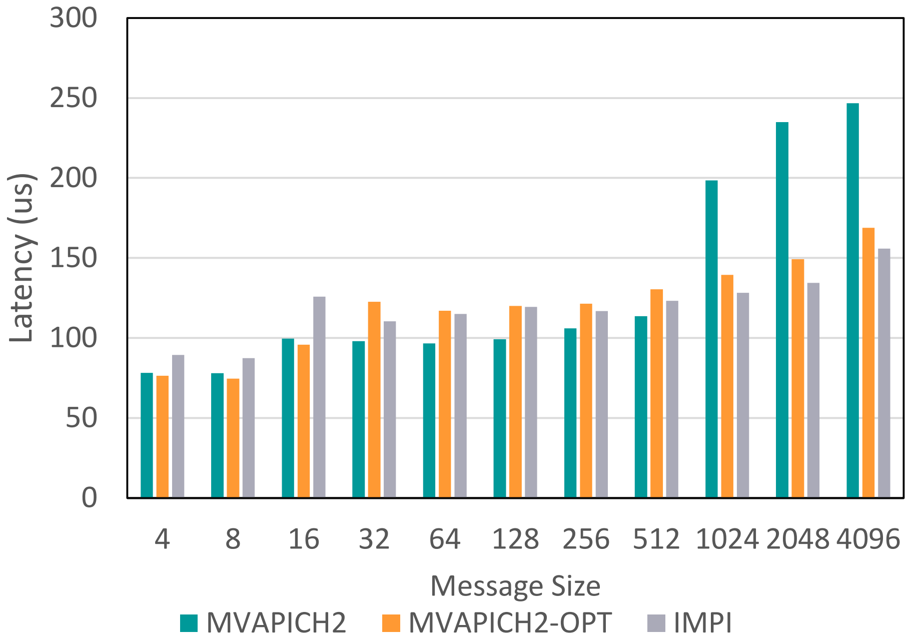
OSU MPI_Allreduce Micro Benchmark

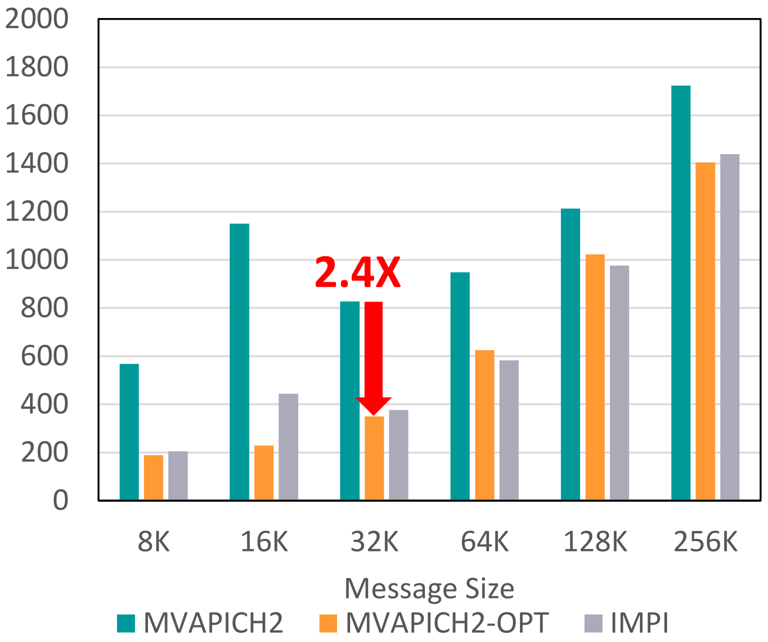
At 2K and 4K processes, MV2X outperforms IntelMPI with 3X and 4X less latency, respectively

**Processes Per Node

* Intel MPI Version 2017.3.196 is used

Performance of MPI_Allreduce On Stampede2 (10,240 Processes)

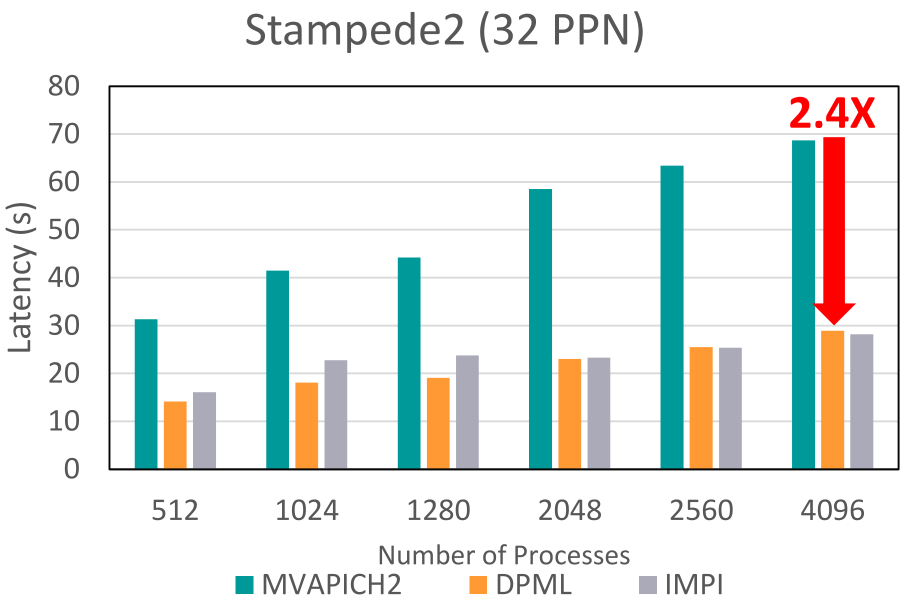


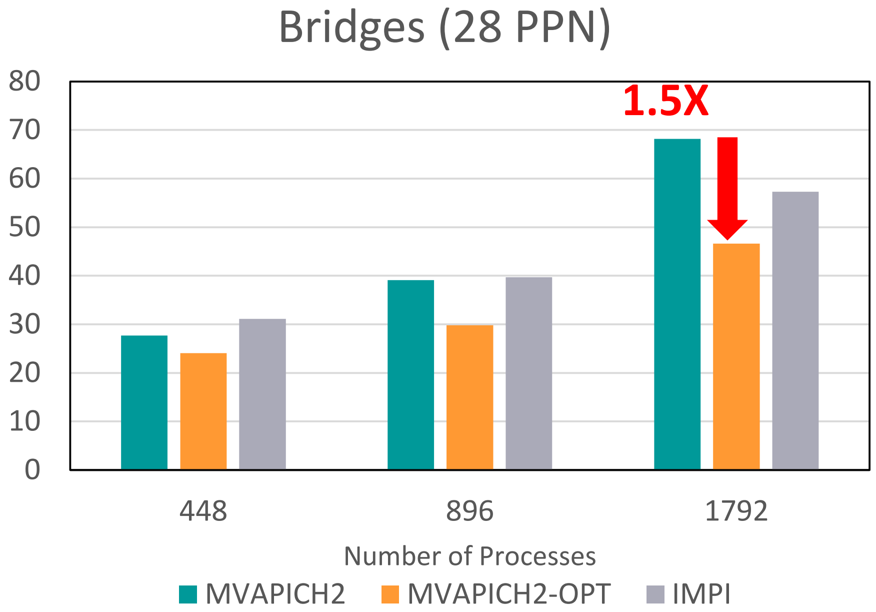


OSU Micro Benchmark 64 PPN

- MPI_Allreduce latency with 32K bytes reduced by 2.4X

Performance of MiniAMR Application On Stampede2 and Bridges



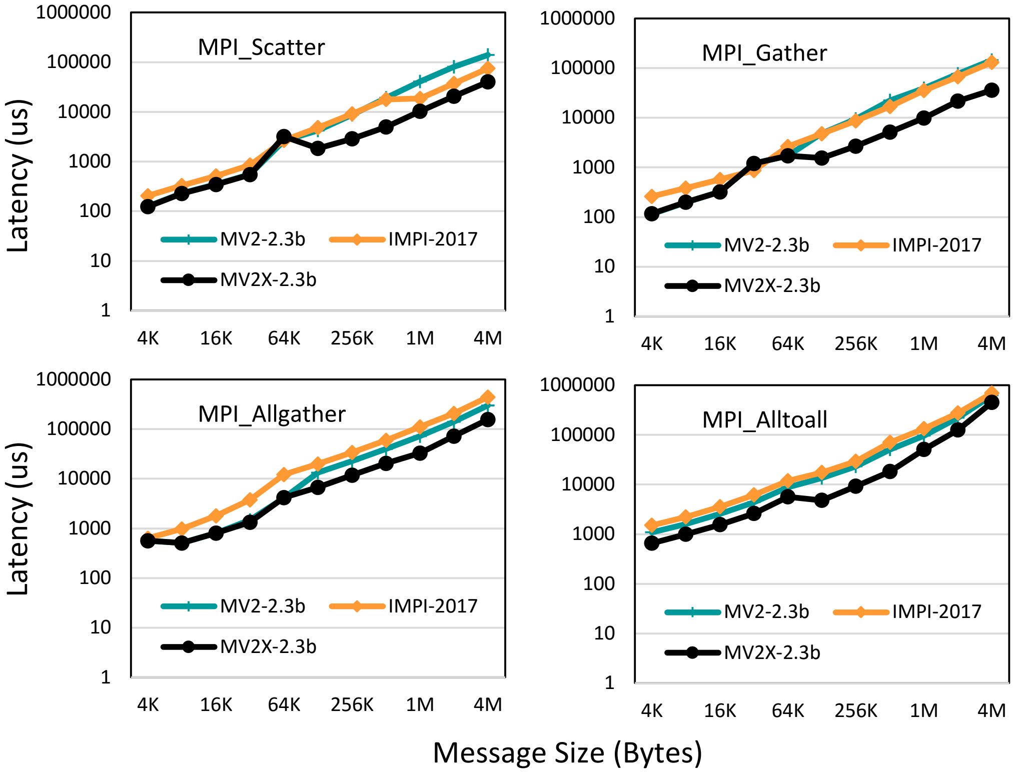


- For MiniAMR Application latency with 2,048 processes, MVAPICH2-OPT can reduce the latency by 2.6X on Stampede2
- On Bridges, with 1,792 processes, MVAPICH2-OPT can reduce the latency by 1.5X

Contention-Aware Kernel-Assisted Collectives

- Kernel-Assisted transfers (CMA, LiMIC, KNEM) offers single-copy transfers for large messages
  - Significant contention with many concurrent reads/writes
  - Contention-aware designs can improve the performance
- Up to 5x improvement for rooted collectives
- Up to 50% improvement for non-rooted collectives

Intra-Node Performance (64 Processes)

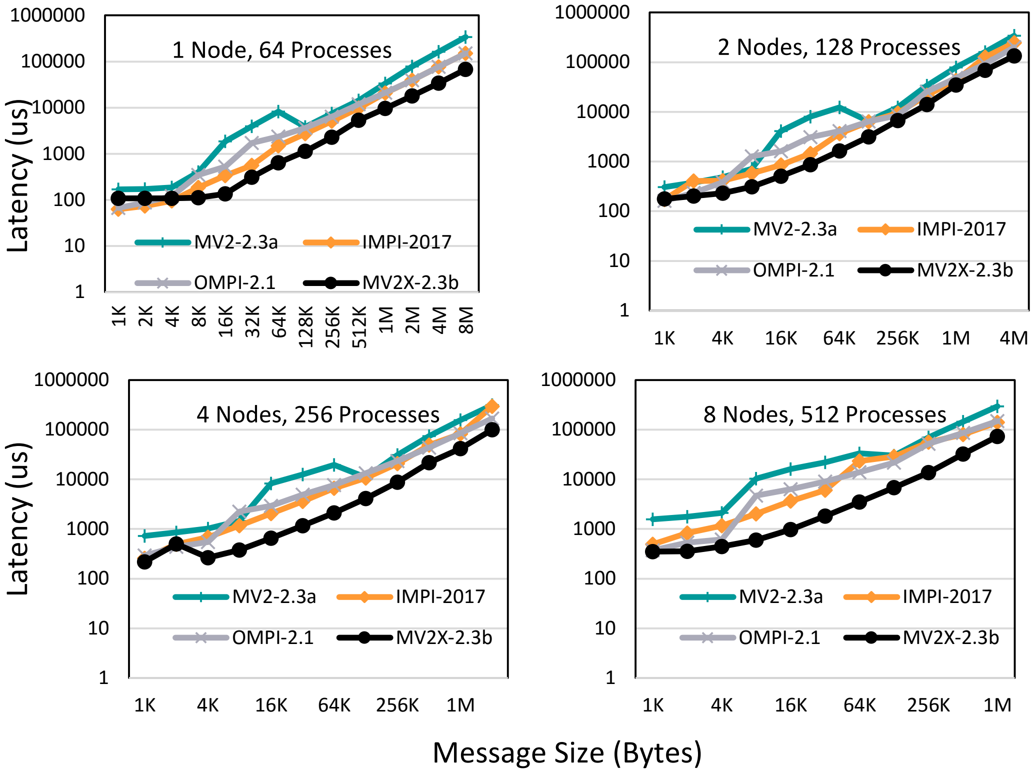


Oakforest-PACS: 68 core Intel Knights Landing (KNL) 7250 @ 1.4 GHz, Intel Omni-Path HCA (100GBps), 16GB MCDRAM (Cache Mode)

Contention-Aware Two-level Collectives

- Fast intra-node algorithms can be used to design improved hieararchical collectives
- Up to 17x improvement for MPI_Gather with 8 nodes, 512 processes
- Similar improvements observed for MPI_Scatter

Performance of MPI_Gather



TACC Stampede2: 68 core Intel Knights Landing (KNL) 7250 @ 1.4 GHz, Intel Omni-Path HCA (100GBps), 16GB MCDRAM (Cache Mode)

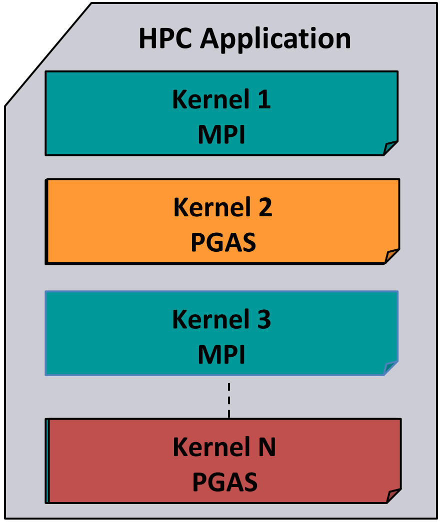
S. Chakraborty, H. Subramoni, and D. K. Panda,* Contention Aware Kernel-Assisted MPI Collectives for Multi/Many-core Systems, *IEEE Cluster '17, BEST Paper Finalist

Outline

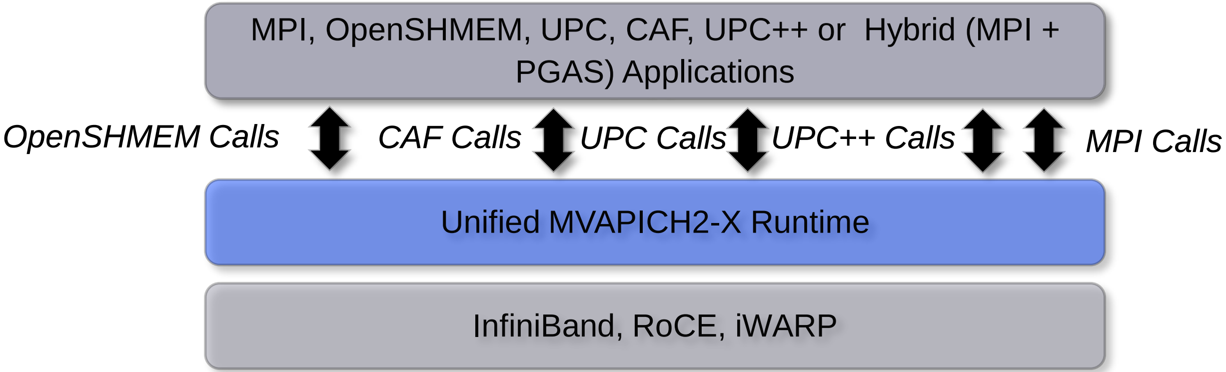
- Scalability for million to billion processors
- Hybrid MPI+PGAS Models for Irregular Applications

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
- Cons
  - Two different runtimes
  - Need great care while programming
  - Prone to deadlock if not careful

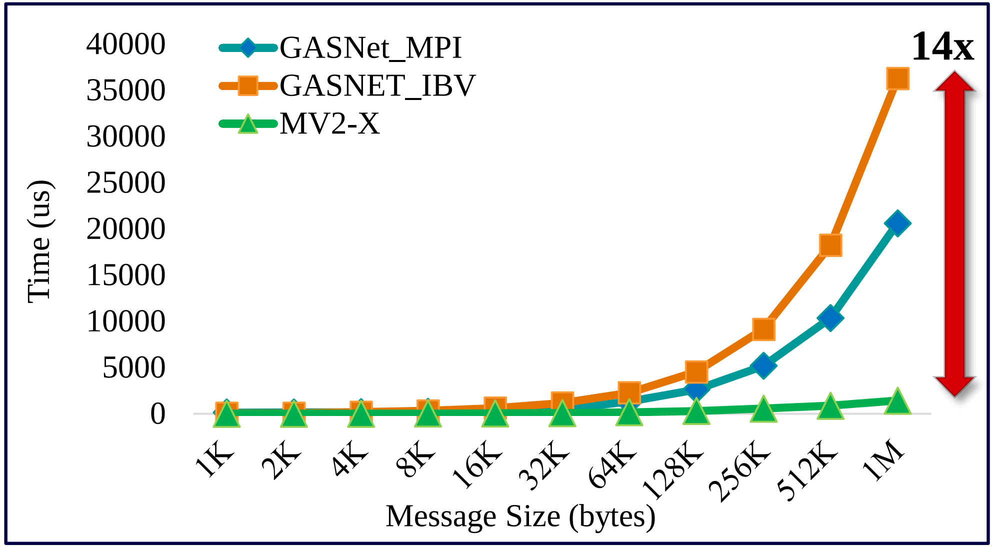


MVAPICH2-X for Hybrid MPI + PGAS Applications

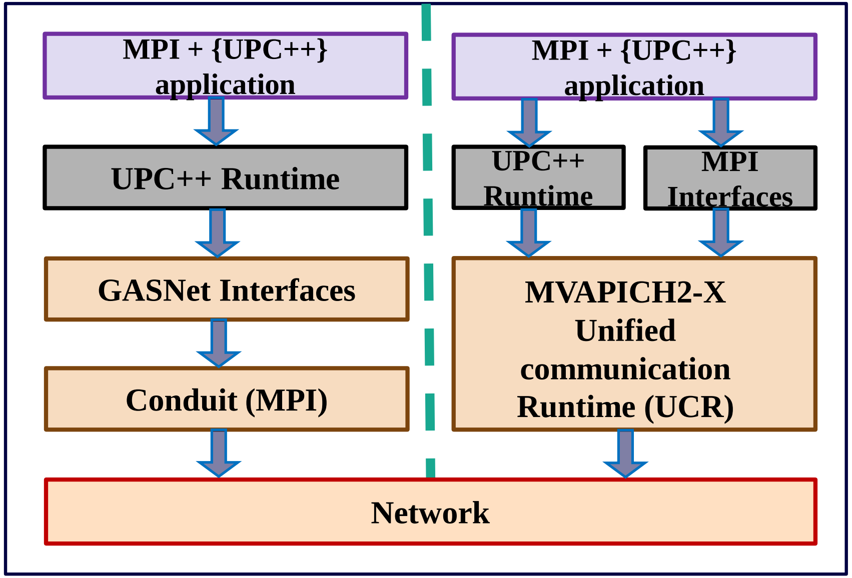


- Unified communication runtime for MPI, UPC, OpenSHMEM, CAF, UPC++ available with MVAPICH2-X 1.9 onwards! (since 2012)
  - http://mvapich.cse.ohio-state.edu
- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, UPC++, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM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

UPC++ Collectives Performance



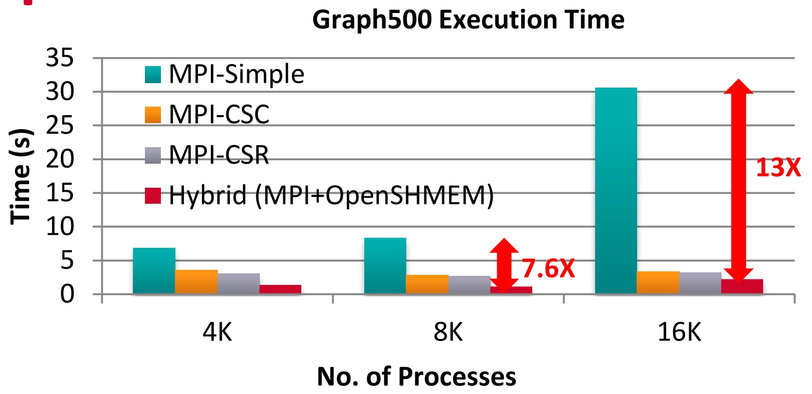
Inter-node Broadcast (64 nodes 1:ppn)

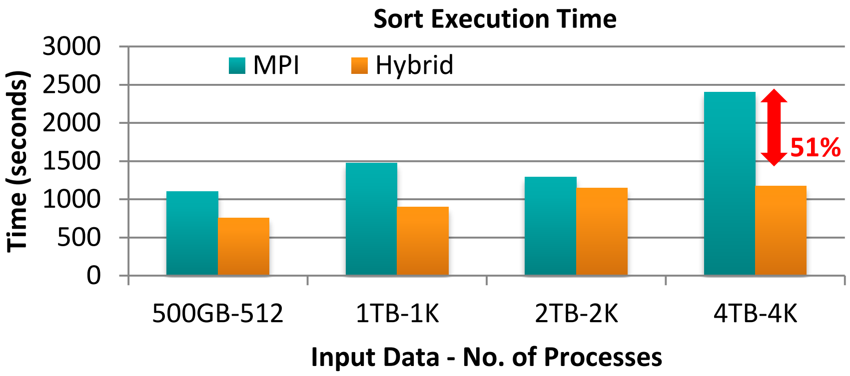


- Full and native support for hybrid MPI + UPC++ applications
- Better performance compared to IBV and MPI conduits
- OSU Micro-benchmarks (OMB) support for UPC++
- Available since MVAPICH2-X 2.2RC1

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

Application Level Performance with Graph500 and Sort





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

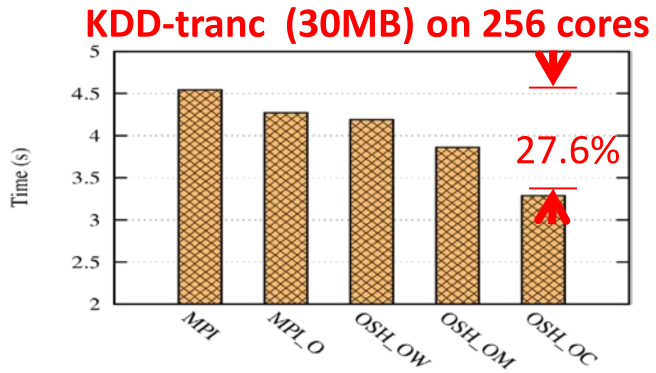
- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - 4,096 processes, 4 TB Input Size
    - MPI – 2408 sec; 0.16 TB/min
    - Hybrid – 1172 sec; 0.36 TB/min
    - **51%** improvement over MPI-design

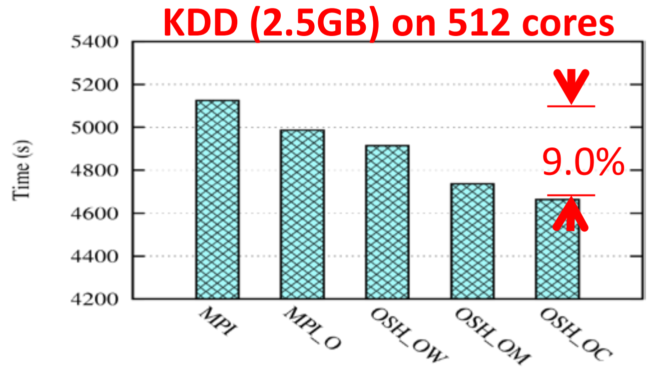
J. Jose, S. Potluri, H. Subramoni, X. Lu, K. Hamidouche, K. Schulz, H. Sundar and D. Panda Designing Scalable Out-of-core Sorting with Hybrid MPI+PGAS Programming Models, PGAS'14

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

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

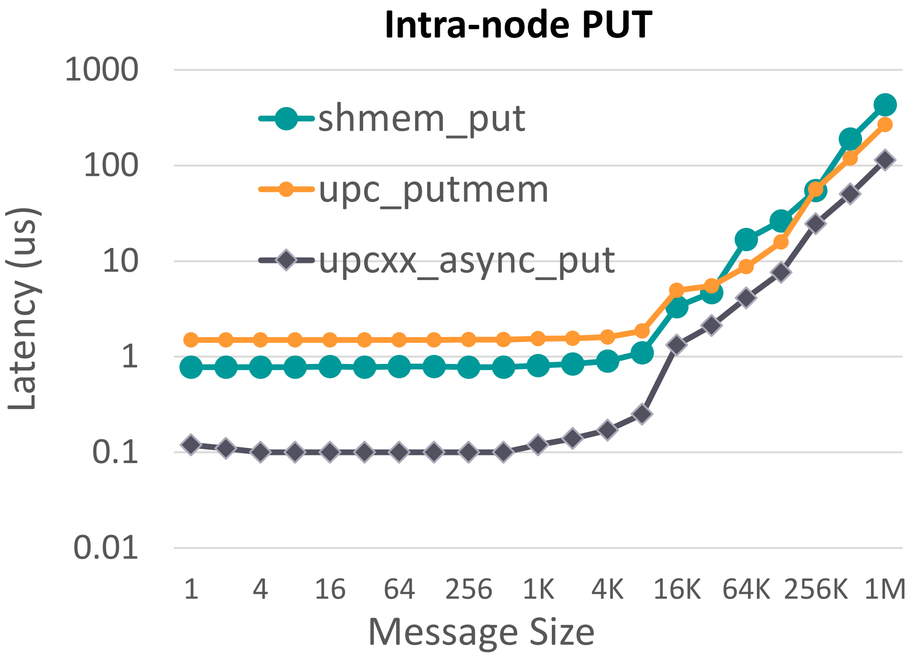


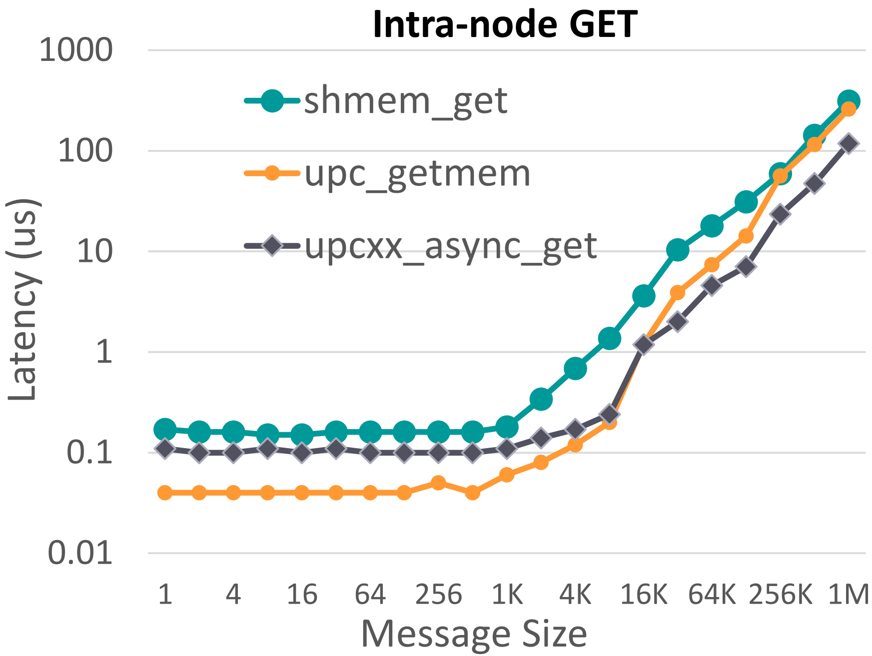


- Benchmark: KDD Cup 2010 (8,407,752 records, 2 classes, k=5)
- For truncated KDD workload on 256 cores, reduce 27.6% execution time
- For full KDD workload on 512 cores, reduce 9.0% execution time

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

Performance of PGAS Models on KNL using MVAPICH2-X

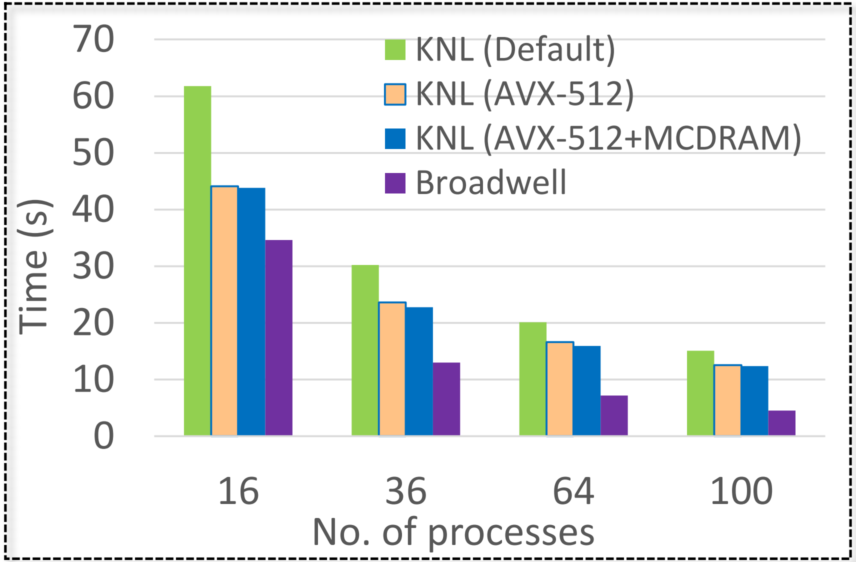




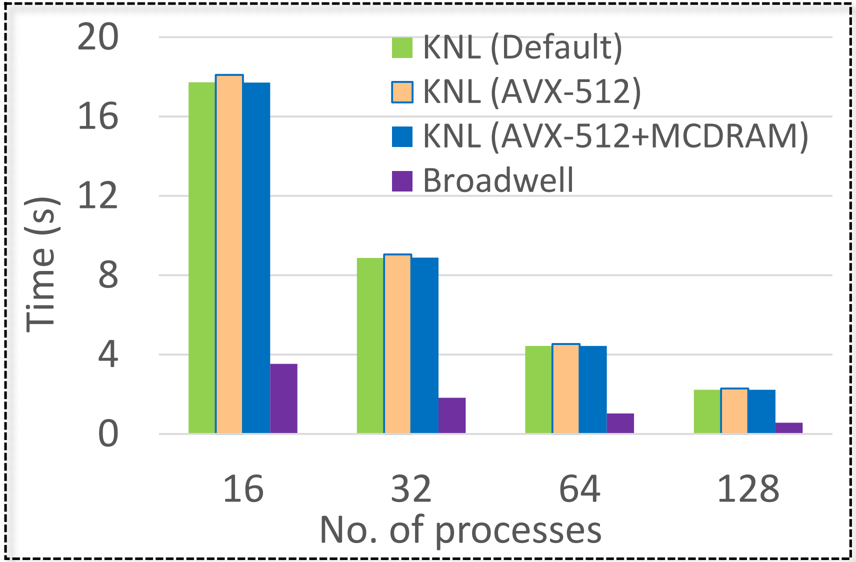
- Intra-node performance of one-sided Put/Get operations of PGAS libraries/languages using MVAPICH2-X communication conduit
- Near-native communication performance is observed on KNL

NAS Parallel Benchmark Evaluation

NAS-BT (PDE solver), CLASS=B

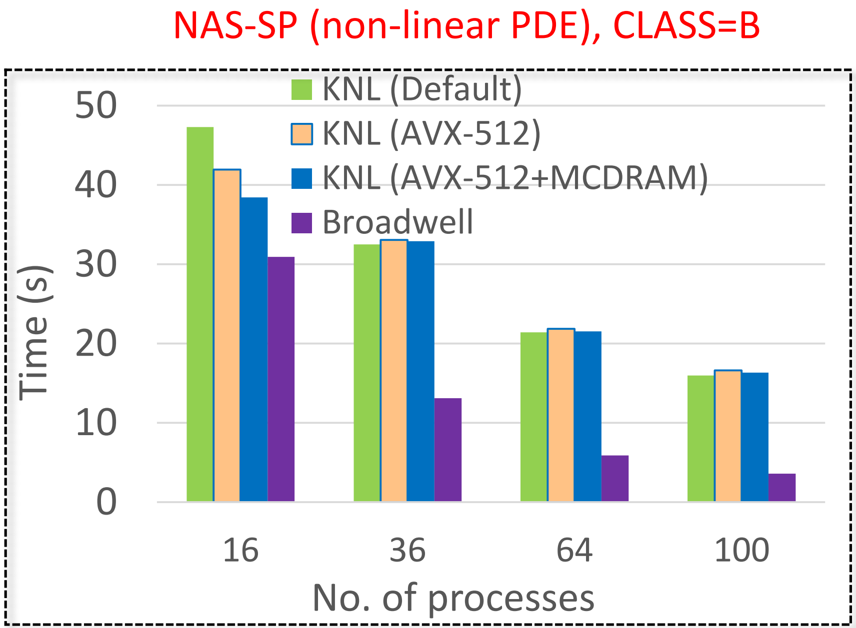


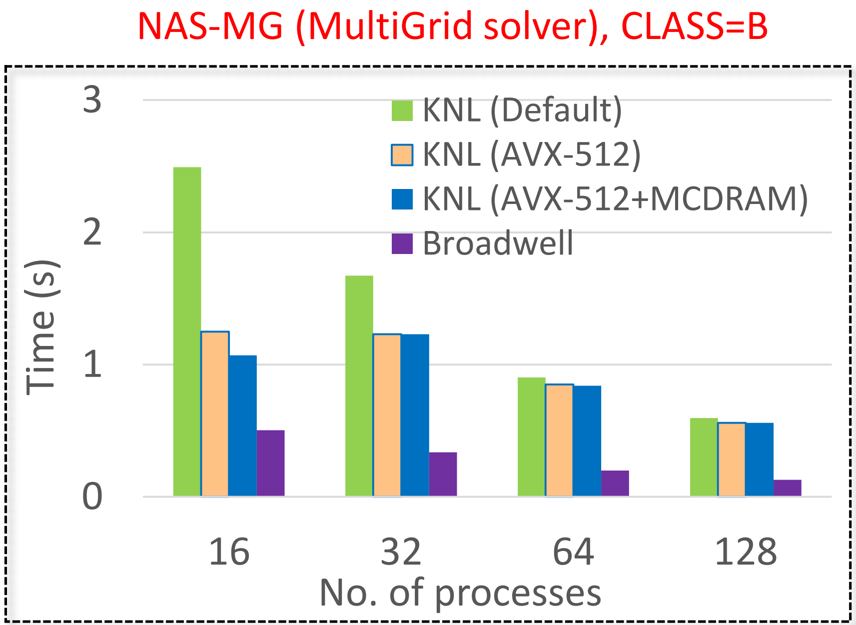
NAS-EP (RNG), CLASS=B



- AVX-512 vectorized execution of BT kernel on KNL showed 30% improvement over default execution while EP kernel didn't show any improvement
- Broadwell showed 20% improvement over optimized KNL on BT and 4X improvement over all KNL executions on EP kernel (random number generation)

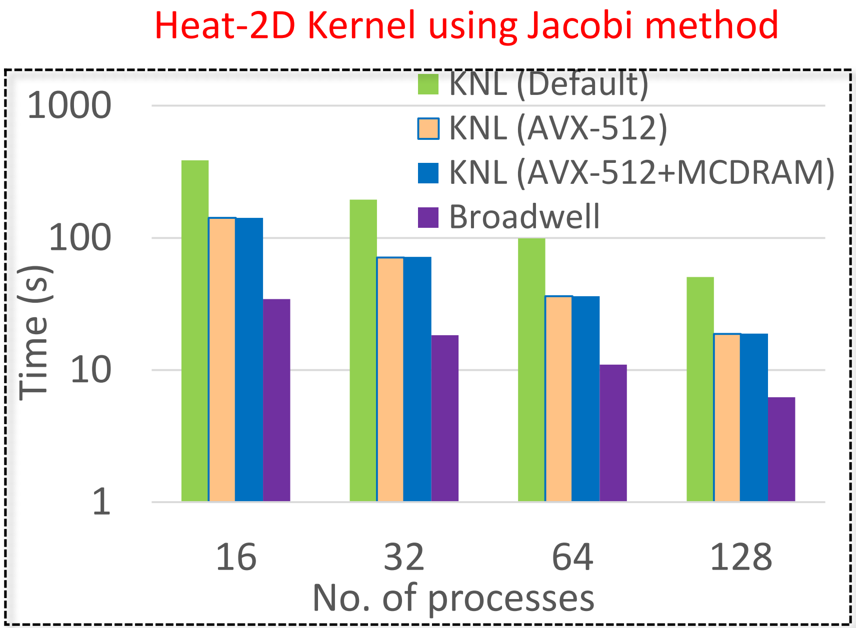
NAS Parallel Benchmark Evaluation (Cont'd)

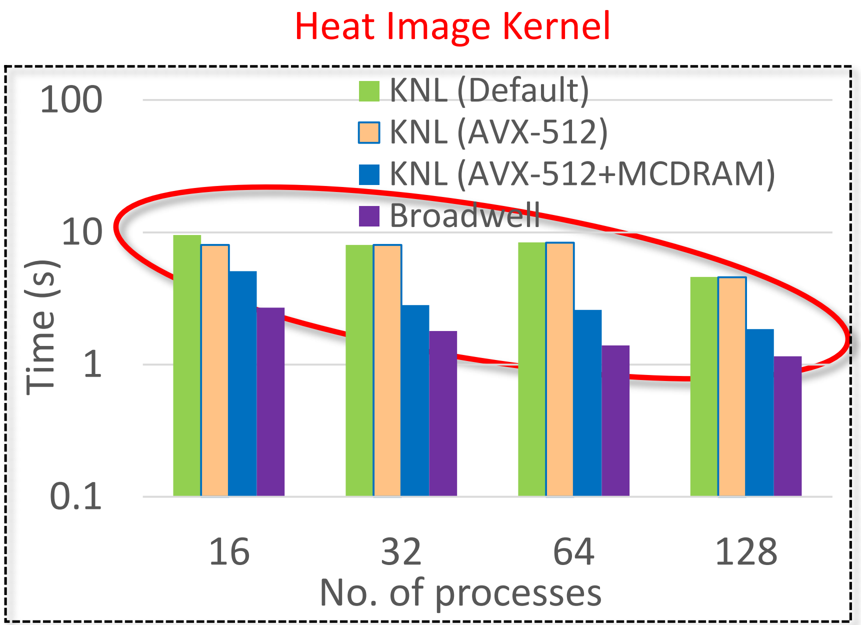




- Similar performance trends are observed on BT and MG kernels as well
- On SP kernel, MCDRAM based execution showed up to 20% improvement over default at 16 processes

Optimized OpenSHMEM with AVX and MCDRAM: Application Kernels Evaluation





- On heat diffusion based kernels AVX-512 vectorization showed better performance
- MCDRAM showed significant benefits on Heat-Image kernel for all process counts. Combined with AVX-512 vectorization, it showed up to 4X improved performance

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

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

Current Research Scientists

- X. Lu
- H. Subramoni

Current Post-doc

- A. Ruhela

Current Research Specialist

- J. Smith
- M. Arnold

Past Students

- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Research Scientist

- K. Hamidouche
- S. Sur

Past Programmers

- D. Bureddy
- J. Perkins

Past Post-Docs

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

Thank You!

panda@cse.ohio-state.edu

Network-Based Computing Laboratory

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

The High-Performance MPI/PGAS Project

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

The High-Performance Deep Learning Project

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