MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Advancing MPI Libraries to the Many-core Era: Designs and Evaluations with MVAPICH2

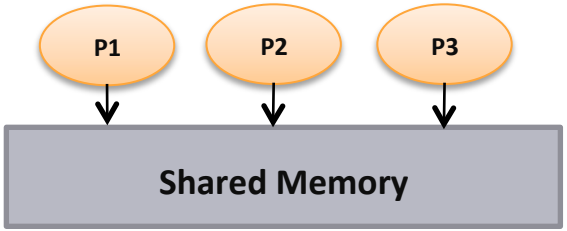
IXPUG '17 Presentation

S. Chakraborty, M. Bayatpour, H. Subramoni and DK Panda

The Ohio State University

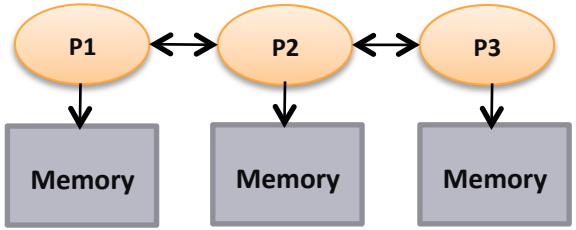
E-mail: {Chakraborty.52,Bayatpour.1,Subramoni.1,panda.2}@osu.edu

Parallel Programming Models Overview



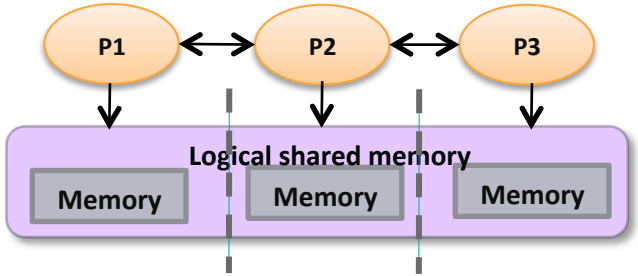
Shared Memory Model

SHMEM, DSM



Distributed Memory Model

MPI (Message Passing Interface)

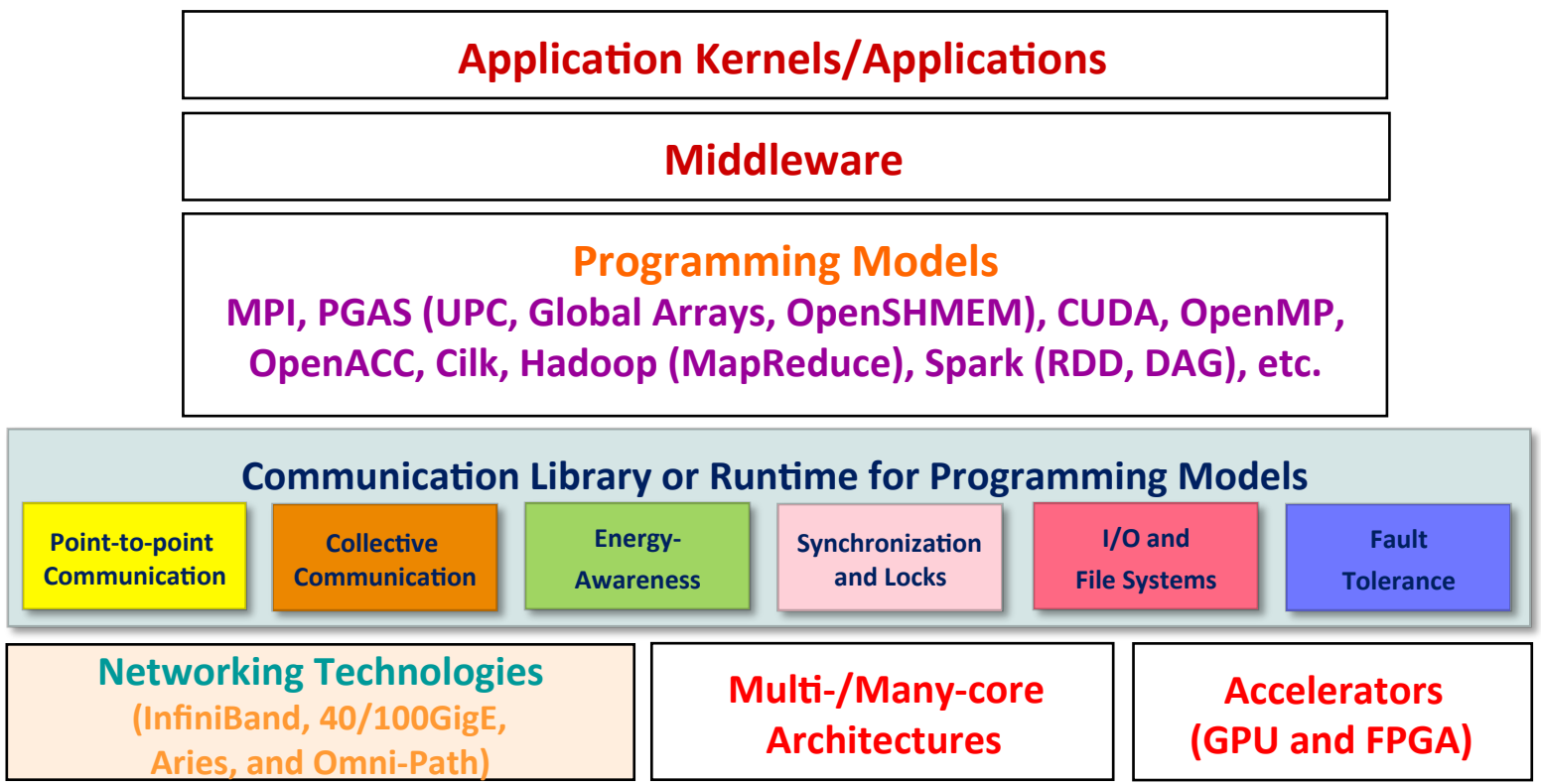


Partitioned Global Address Space (PGAS)

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

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



Co-Design Opportunities and Challenges across Various Layers

Performance

Scalability

Resilience

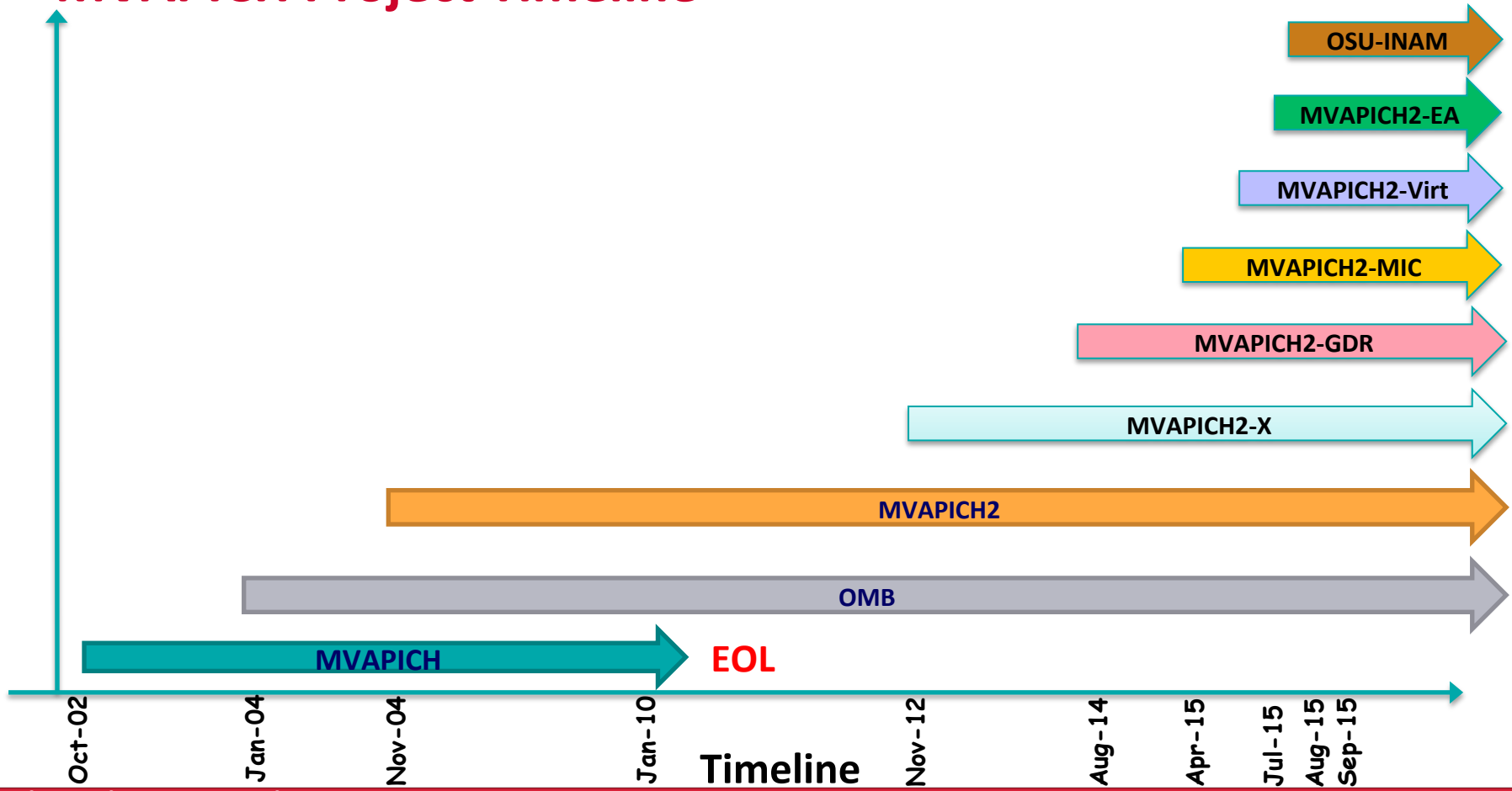
Designing (MPI+X) for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
- Scalable Collective communication
  - Offloaded
  - Non-blocking
  - Topology-aware
- Balancing intra-node and inter-node communication for next generation multi-/many-core (128-1024 cores/node)
  - 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 + OpenSHMEM, CAF, MPI + UPC++…
- Virtualization
- Energy-Awareness

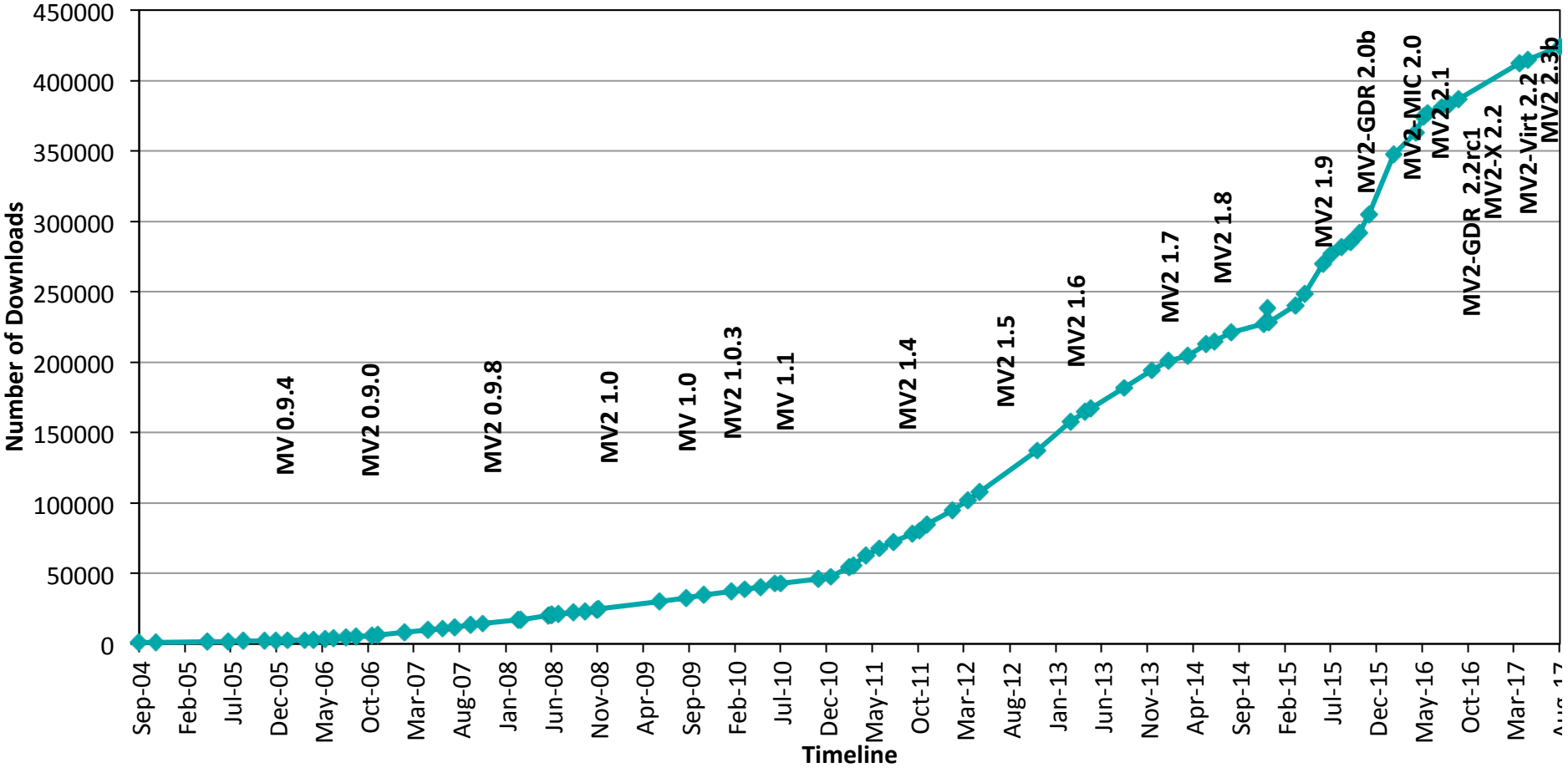
Overview of the MVAPICH2 Project

- 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 427,000 (> 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)

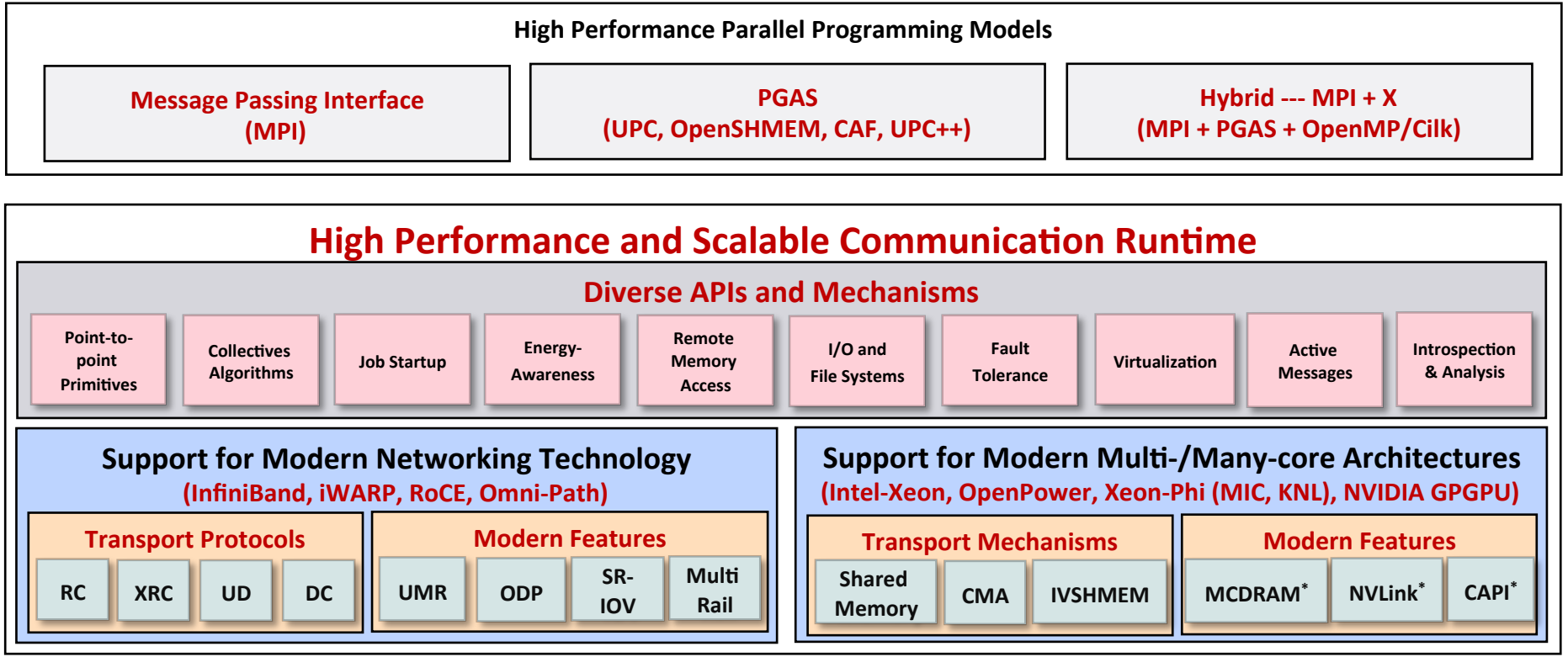
MVAPICH Project Timeline



MVAPICH2 Release Timeline and Downloads



Architecture of MVAPICH2 Software Family



* Upcoming

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

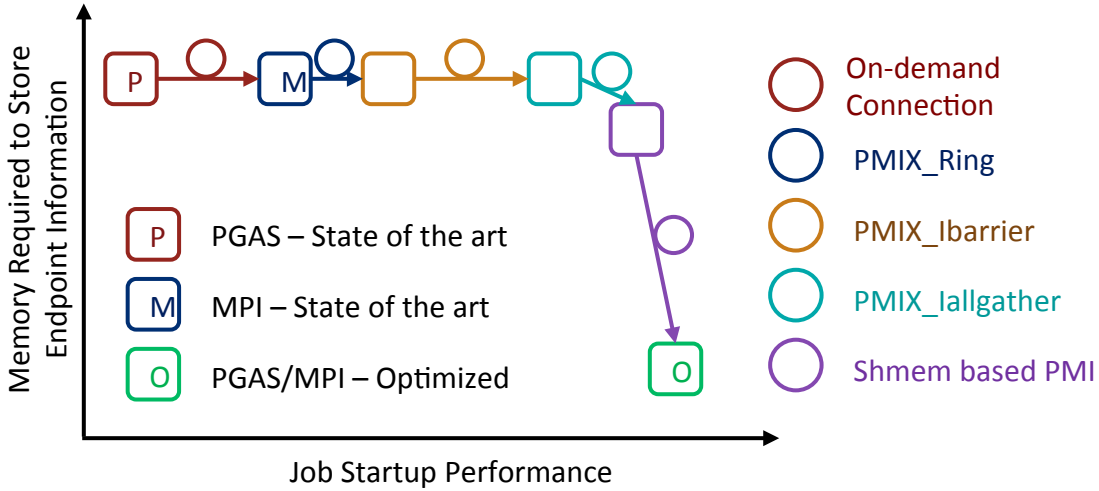
MVAPICH2 2.3b

- Released on 08/10/2017
- Major Features and Enhancements
  - Based on MPICH-3.2
  - Enhance performance of point-to-point operations for CH3-Gen2 (InfiniBand), CH3-PSM, and CH3-PSM2 (Omni-Path) channels
  - Improve performance for MPI-3 RMA operations
  - Introduce support for Cavium ARM (ThunderX) systems
  - Improve support for process to core mapping on many-core systems
    - New environment variable MV2_THREADS_BINDING_POLICY for multi-threaded MPI and MPI+OpenMP applications
    - Support `linear' and `compact' placement of threads
    - Warn user if over-subscription of core is detected
  - Improve launch time for large-scale jobs with mpirun_rsh
  - Add support for non-blocking Allreduce using Mellanox SHARP
  - Efficient support for different Intel Knight's Landing (KNL) models
  - Improve performance for Intra- and Inter-node communication for OpenPOWER architecture
  - Improve support for large processes per node and huge pages on SMP systems
  - Enhance collective tuning for many architectures/systems
  - Enhance support for MPI_T PVARs and CVARs

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Many-core Era

- Fast and Scalable Job Start-up
- Dynamic and Adaptive Communication Protocols and Tag Matching
- Contention-aware Designs for Intra-node Collectives
- Scalable Multi-leader Designs for Collectives
- Kernel-Assisted Communication Designs for KNL
- Efficient RMA-based Designs for Graph500 on KNL

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

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)

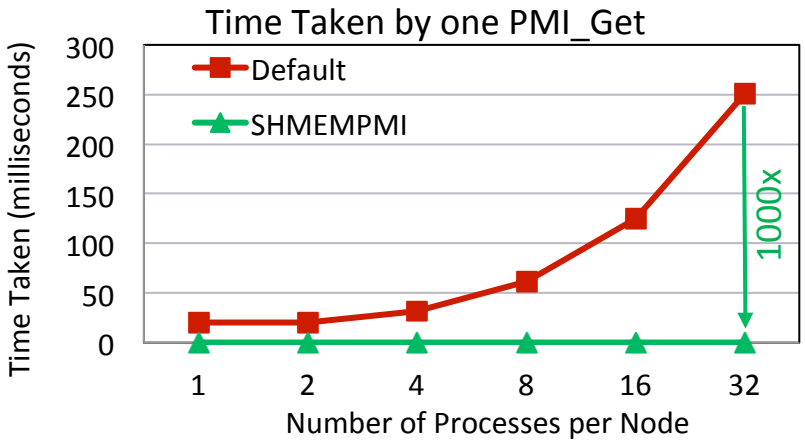
PMI Extensions for Scalable MPI Startup. S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia '14)

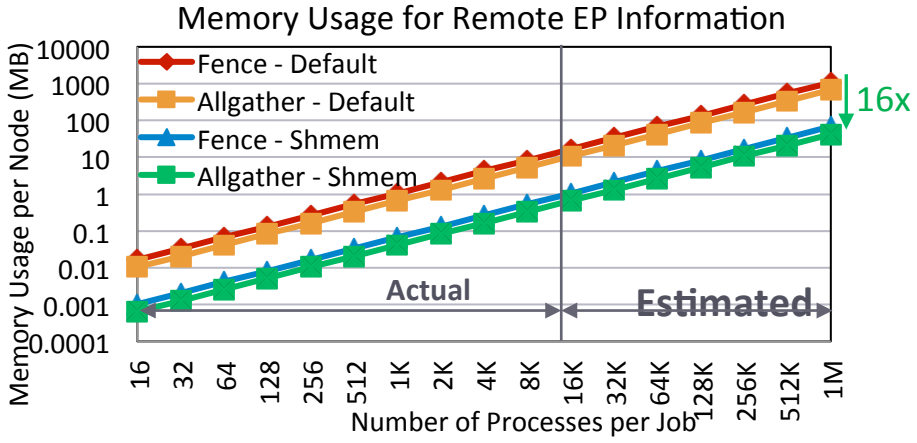
Non-blocking PMI Extensions for Fast MPI Startup. S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '15)

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)

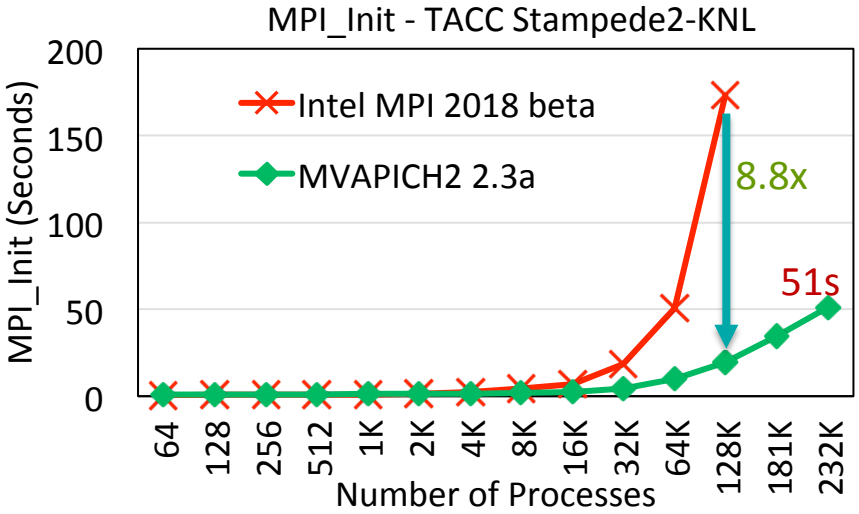
Process Management Interface (PMI) over Shared Memory (SHMEMPMI)

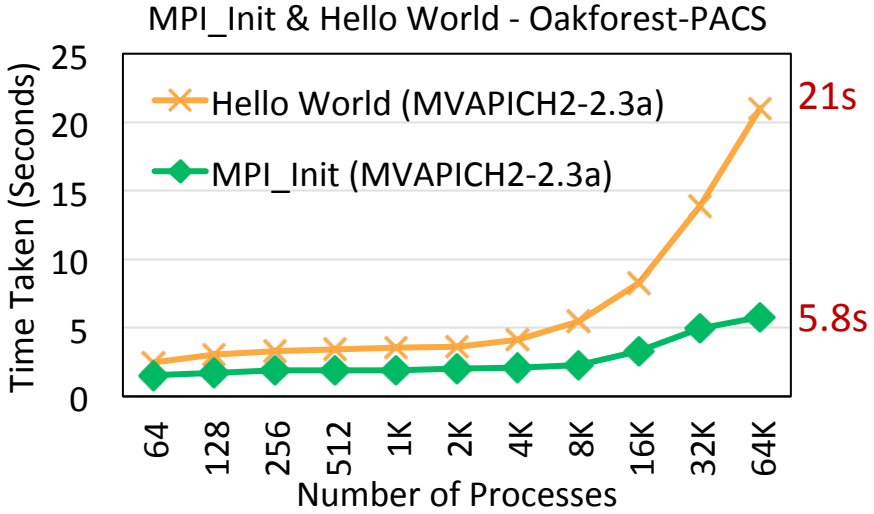
- SHMEMPMI allows MPI processes to directly read remote endpoint (EP) information from the process manager through shared memory segments
- Only a single copy per node - O(processes per node) reduction in memory usage
- Estimated savings of 1GB per node with 1 million processes and 16 processes per node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





Startup Performance on KNL + Omni-Path





- MPI_Init takes 51 seconds on 231,956 processes on 3,624 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 MVAPICH2-2.3a and as patch for SLURM-15.08.8 and SLURM-16.05.1

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Many-core Era

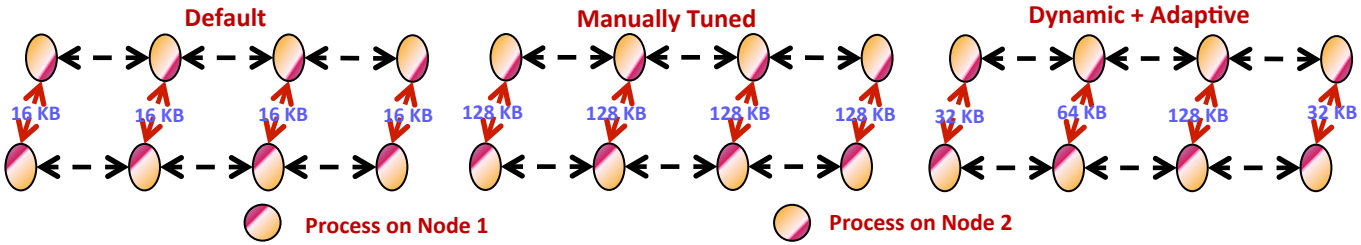
- Fast and Scalable Job Start-up
- Dynamic and Adaptive Communication Protocols and Tag Matching
- Contention-aware Designs for Intra-node Collectives
- Scalable Multi-leader Designs for Collectives
- Kernel-Assisted Communication Designs for KNL
- Efficient RMA-based Designs for Graph500 on KNL

Dynamic and Adaptive MPI Point-to-point Communication Protocols

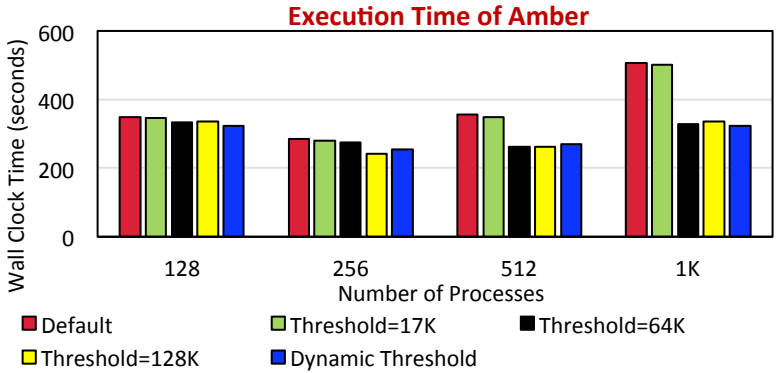
Desired Eager Threshold

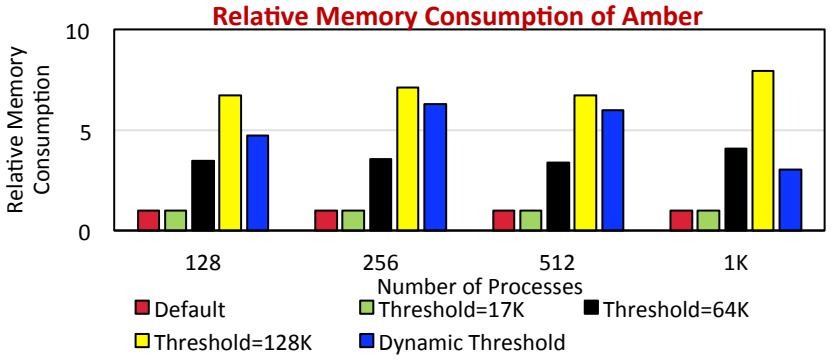
Process Pair	Eager Threshold (KB)
0 – 4	32
1 – 5	64
2 – 6	128
3 – 7	32

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

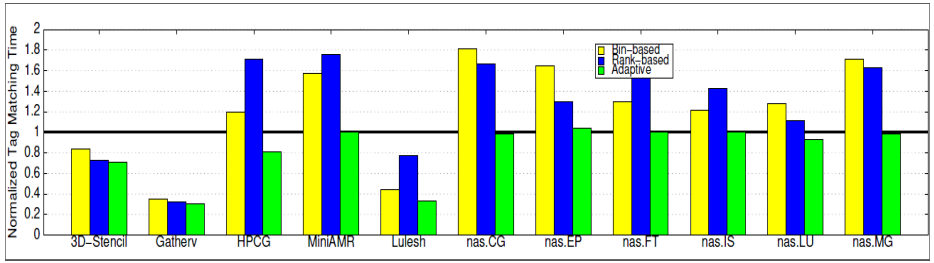
- Dynamically adapt to communication patterns
- Use different strategies for different ranks
- Decisions are based on the number of request object that must be traversed before hitting on the required one

Results

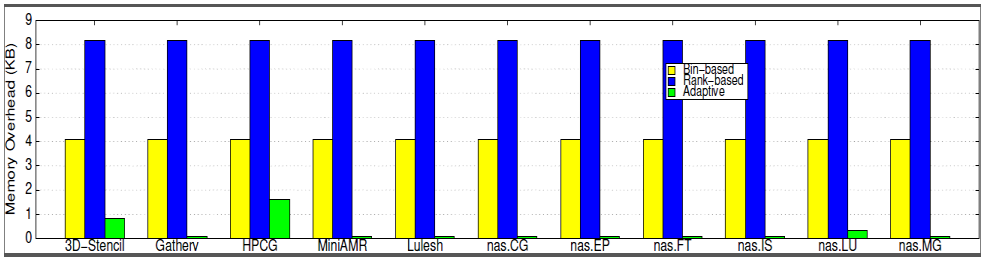
Better performance than other state-of-the art tag-matching schemes

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]

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Many-core Era

- Fast and Scalable Job Start-up
- Dynamic and Adaptive Communication Protocols and Tag Matching
- Contention-aware Designs for Intra-node Collectives
- Scalable Multi-leader Designs for Collectives
- Kernel-Assisted Communication Designs for KNL
- Efficient RMA-based Designs for Graph500 on KNL

Different Kernel-Assisted Single Copy Mechanisms

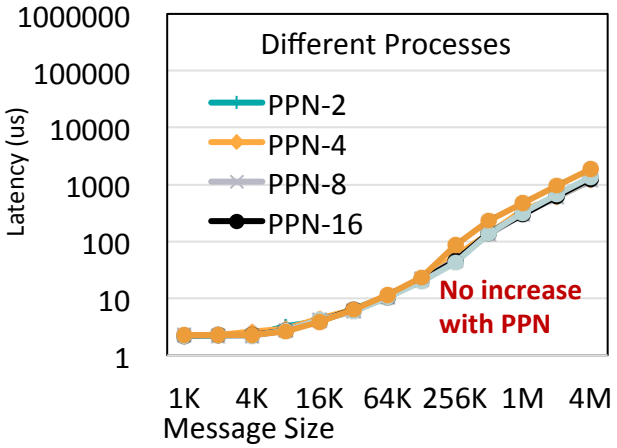
	CMA	KNEM	LiMiC
Cookie/Region Creation	Not Required	Required	Required
Permission Check	Supported	Supported	Not Supported
Availability	Included in Linux 3.2+	Kernel Module	Kernel Module

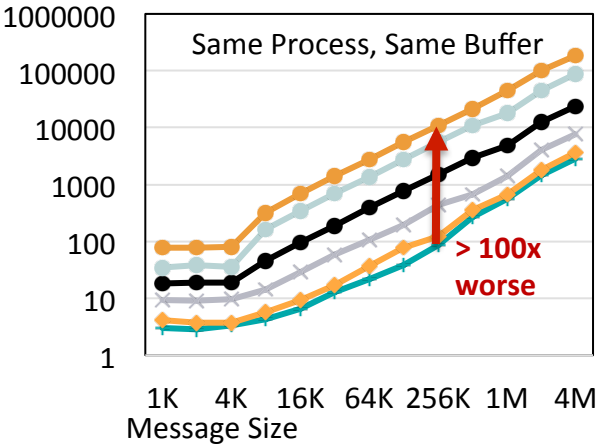
MPI Library Support

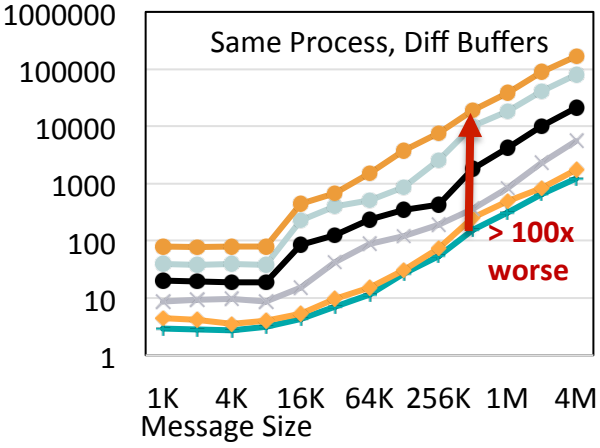
	CMA	KNEM	LiMiC
MVAPICH2-2.3a	√	x	√
OpenMPI 2.1.0	√	√	x
Intel MPI 2017	√	x	x

CMA (Cross Memory Attach) is the most widely supported kernel-assisted transfer mechanism

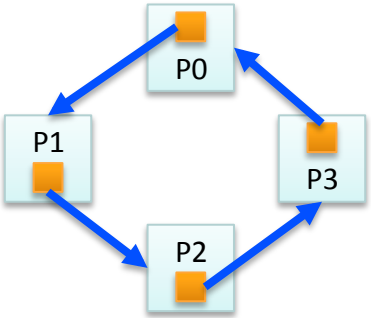
Impact of Communication Pattern on CMA Performance



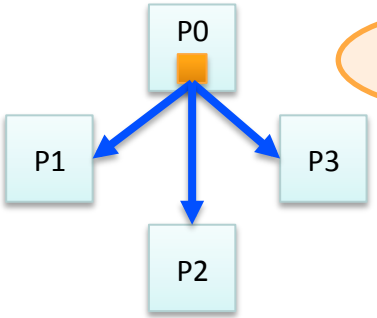




All-to-All – Good Scalability

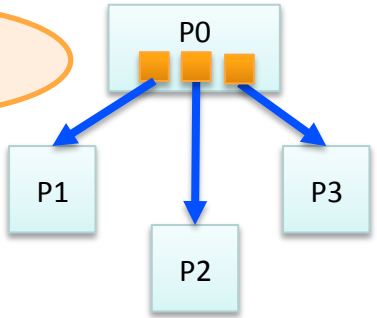


One-to-All - Poor Scalability

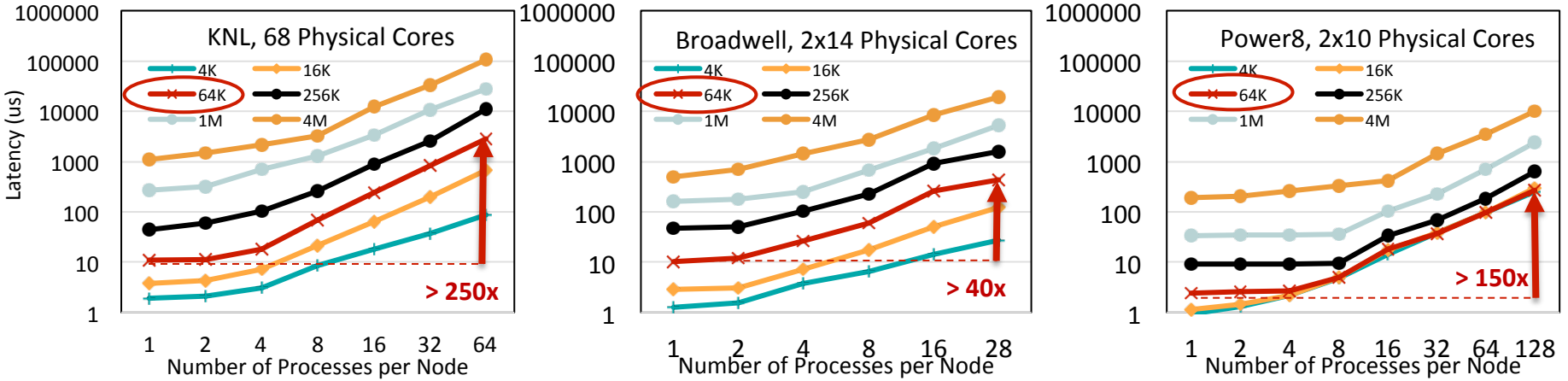


Contention is at Process level

One-to-All – Poor Scalability

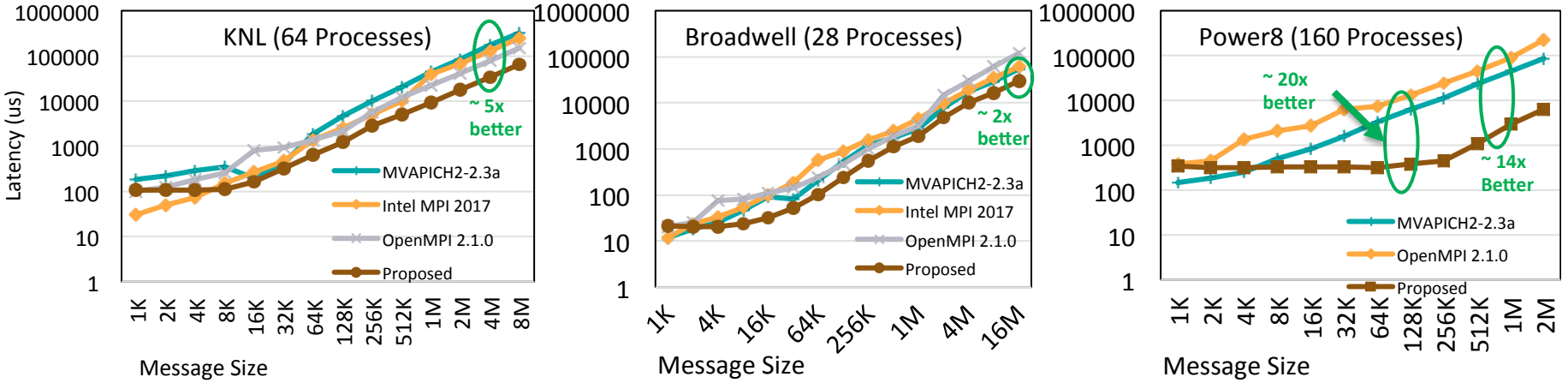


One-to-all Communication with CMA on Different Architectures



- Super-linear degradation in all three architectures
- More Cores => More Contention => More Degradation
- Contention aware design: limit number of concurrent reads/writes
  - Hit the "sweet spot" between contention and concurrency

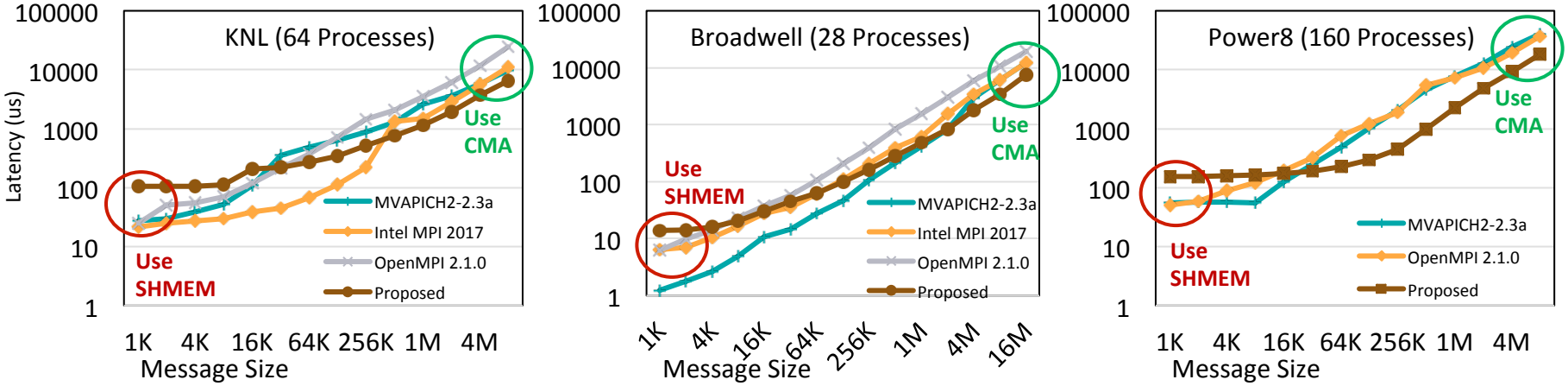
Performance Comparison of MPI_Scatter



- Up to 2x-5x improvement on KNL and Broadwell for medium to large messages
- Up to 20x improvement on Power due to large process count
- Significantly faster than Intel MPI and Open MPI for messages > 4KB
- Similar improvements observed for MPI_Gather

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***

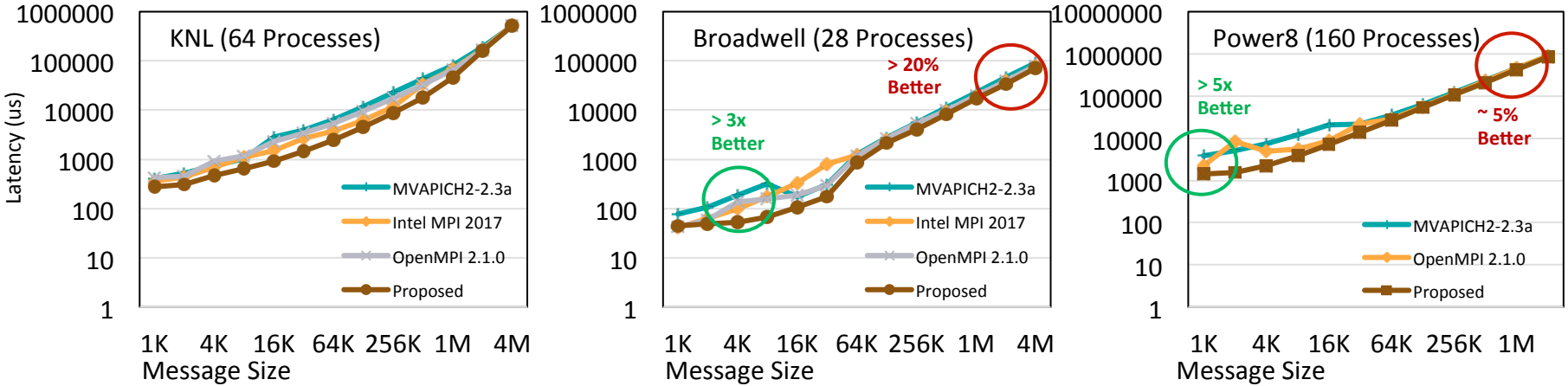
Performance Comparison of MPI_Bcast



- Up to 2x - 4x improvement over existing implementation for 1MB messages
- Up to 1.5x – 2x faster than Intel MPI and Open MPI for 1MB messages

- Improvements obtained for large messages only
  - p-1 copies with CMA, p copies with Shared memory
  - Fallback to SHMEM for small messages

Performance Comparison of MPI_Alltoall



- Improvement from avoiding exchange of control messages
- Improvement observed even for 1KB messages

- Up to 3x-5x improvement for small and medium messages (compared to default)
- Large message performance bound by system bandwidth (5-20% improvement)
- Similar improvements for MPI_Allgather

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Many-core Era

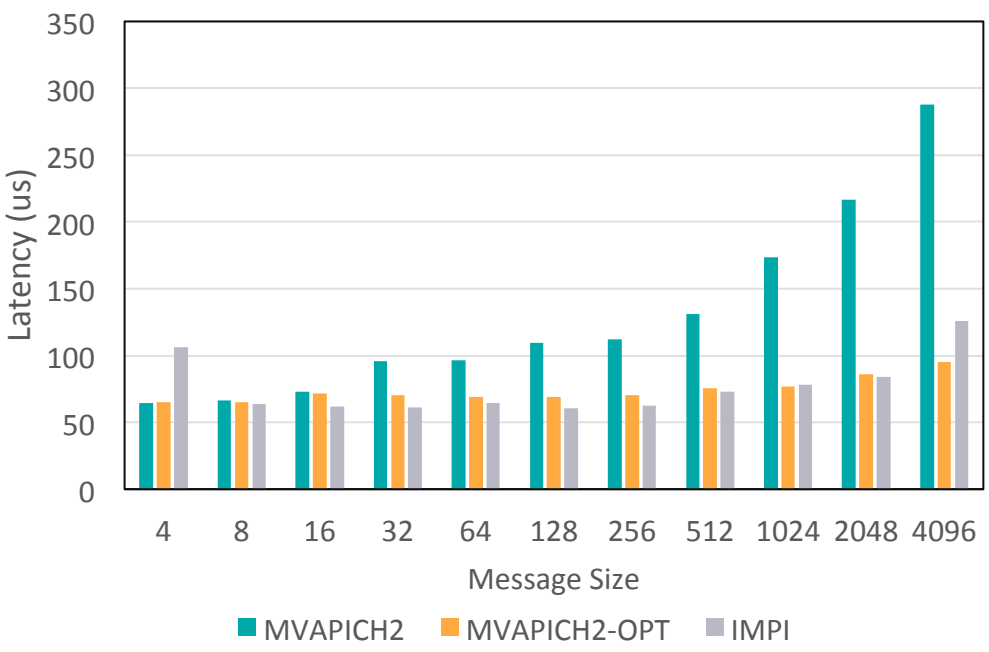
- Fast and Scalable Job Start-up
- Dynamic and Adaptive Communication Protocols and Tag Matching
- Contention-aware Designs for Intra-node Collectives
- Scalable Multi-leader Designs for Collectives
- Kernel-Assisted Communication Designs for KNL
- Efficient RMA-based Designs for Graph500 on KNL

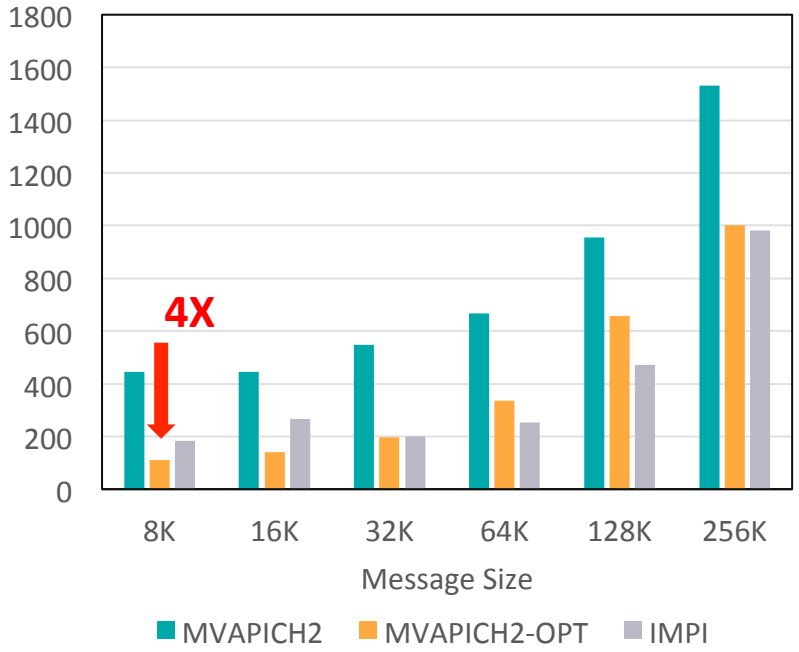
Scalable Reduction Collectives with Multi-Leaders

- Existing designs for MPI_Allreduce do not take advantage of the vast parallelism available in modern multi-/many-core processors
- Proposed a new solution for MPI_Allreduce
- DPML Take advantage of the parallelism offered by
  - Multi-/many-core architectures
  - The high throughput and high-end features offered by InfiniBand and Omni-Path

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 Stampede2 (1,024 processes)*





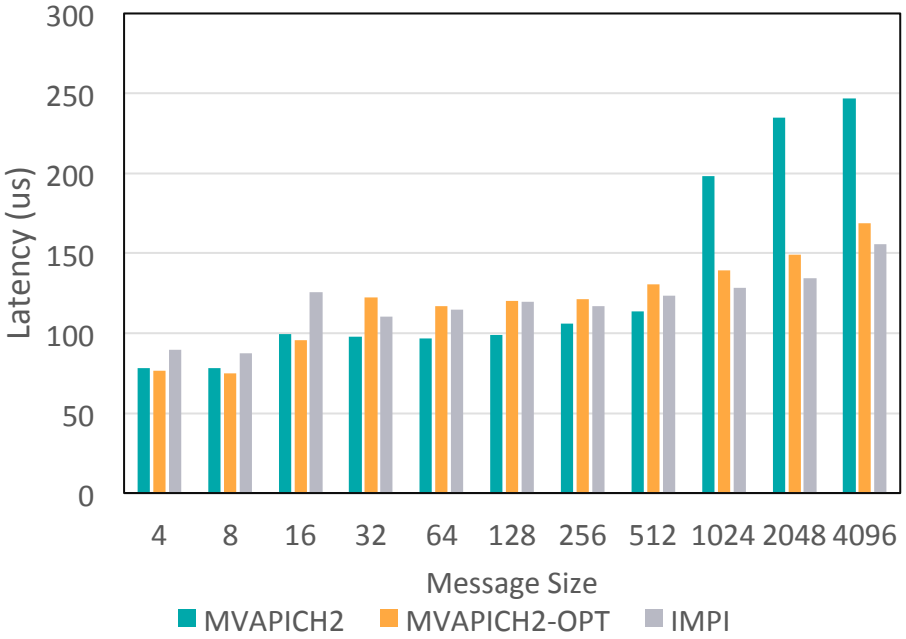
OSU Micro Benchmark with 32 PPN**

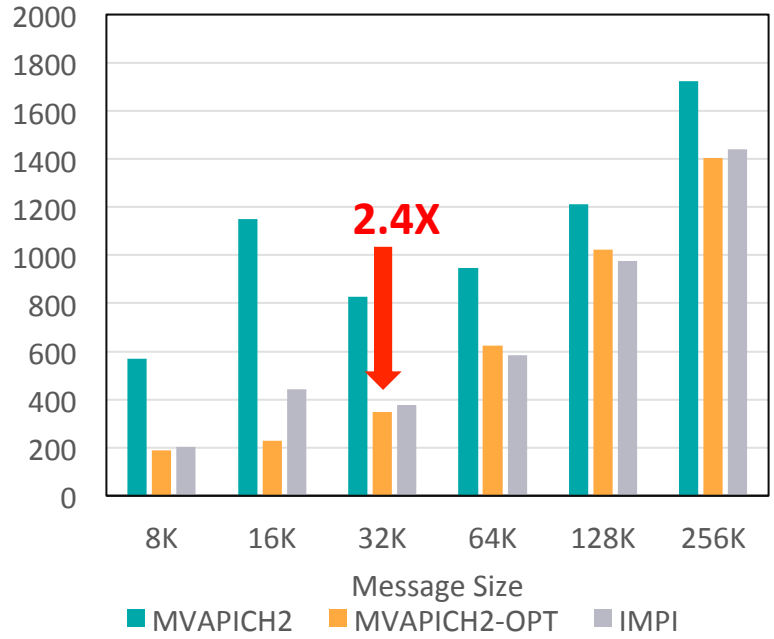
- For MPI_Allreduce latency with 8K bytes, MVAPICH2-OPT can reduce the latency by 4X

*In all our evaluations, Intel MPI 2017.1.132 and MVAPICH2 2.2 have been used

**Processes Per Node

Performance of MPI_Allreduce On Stampede2 (10,240 Processes)

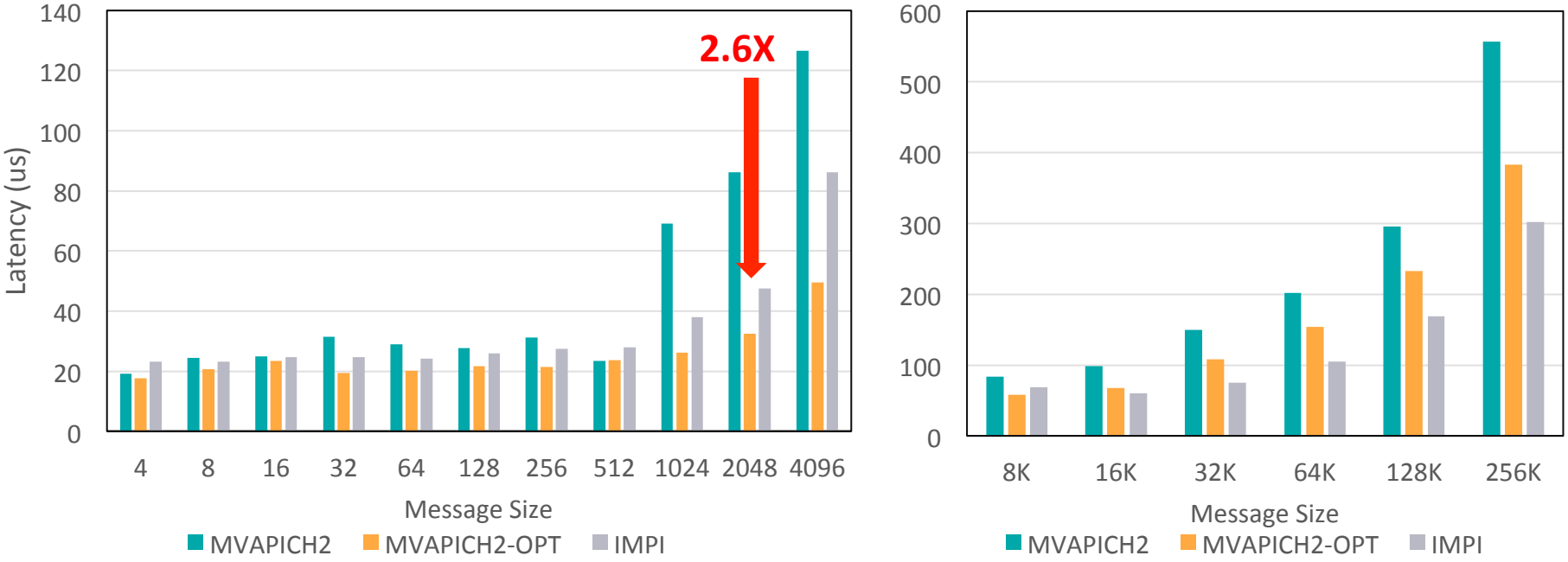




OSU Micro Benchmark 64 PPN

- For MPI_Allreduce latency with 32K bytes, MVAPICH2-OPT can reduce the latency by 2.4X

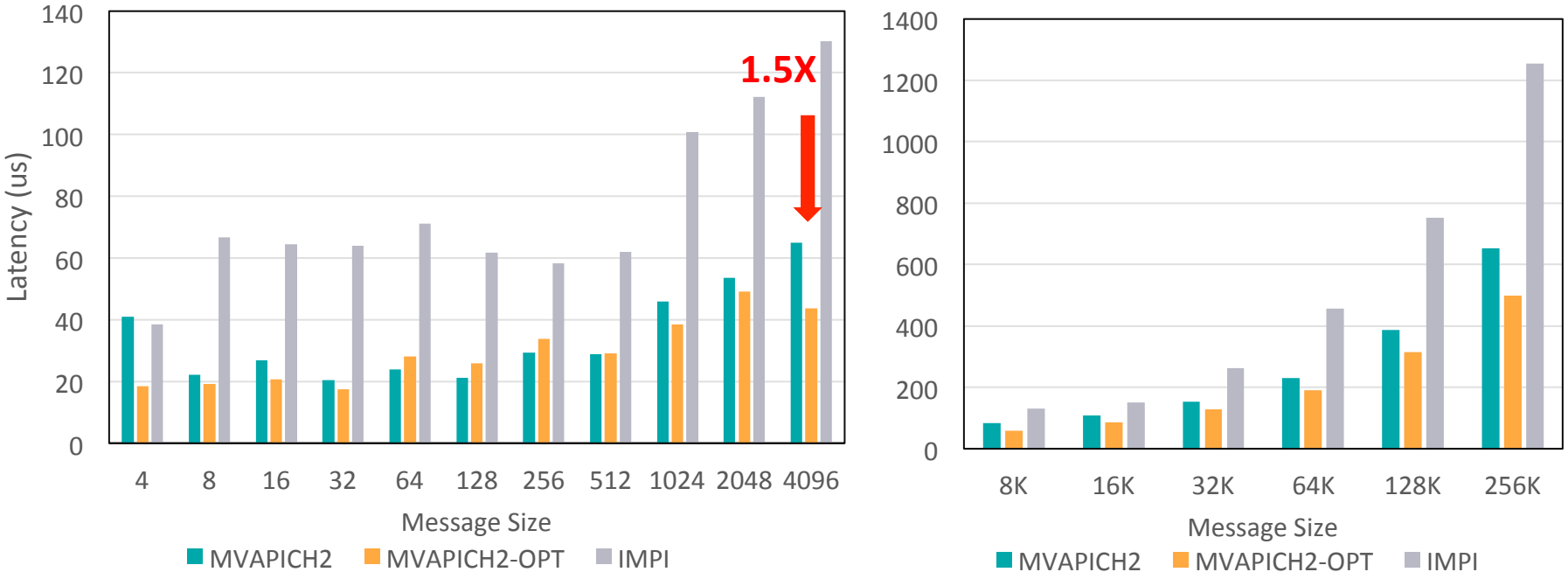
Performance of MPI_Allreduce On Bridges (1,024 Processes)



OSU Micro Benchmark with 16 PPN

- For MPI_Allreduce latency with 2K bytes, MVAPICH2-OPT can reduce the latency by 2.6X

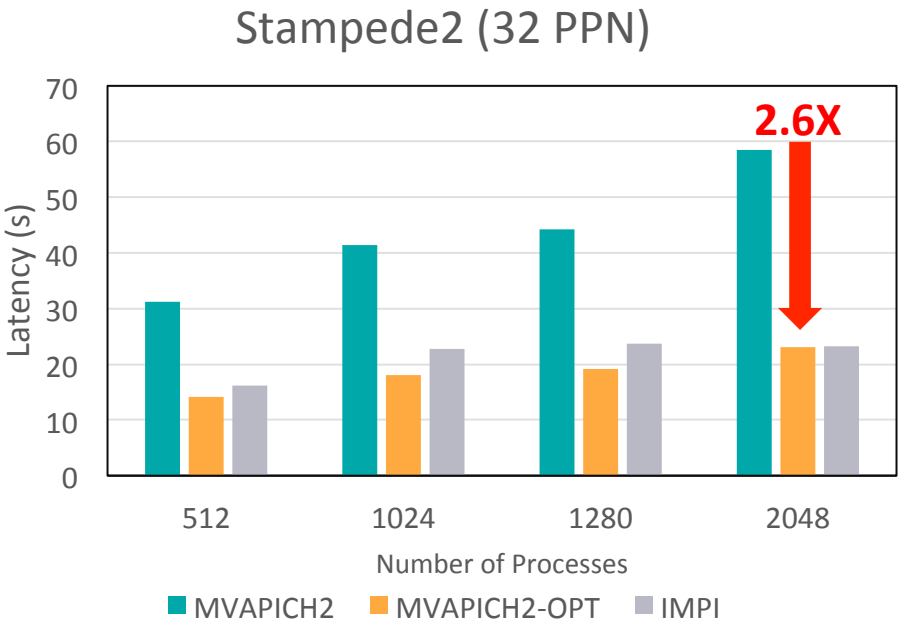
Performance of MPI_Allreduce On Bridges (1,792 Processes)

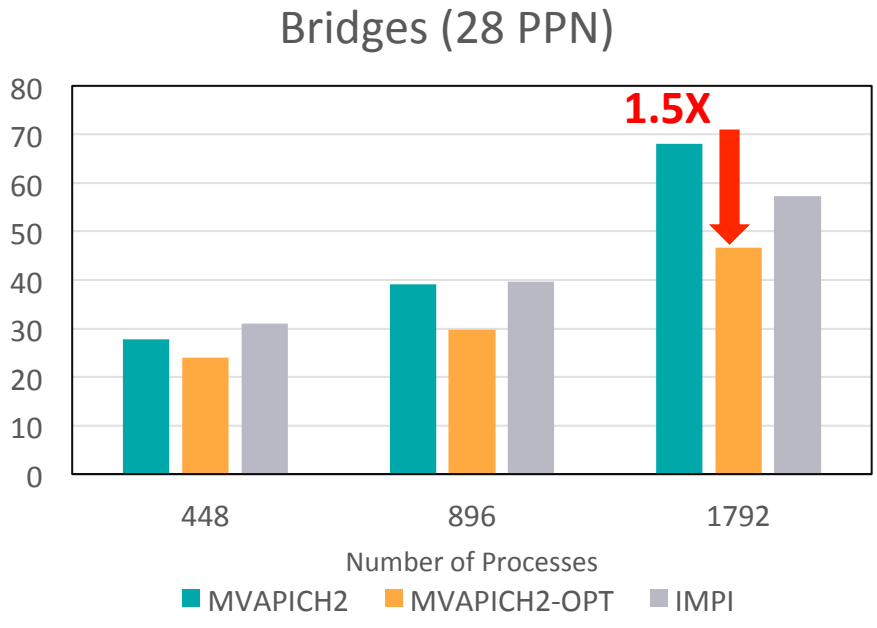


OSU Micro Benchmark with 28 PPN

- For MPI_Allreduce latency with 4K bytes, MVAPICH2-OPT can reduce the latency by 1.5X

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

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Many-core Era

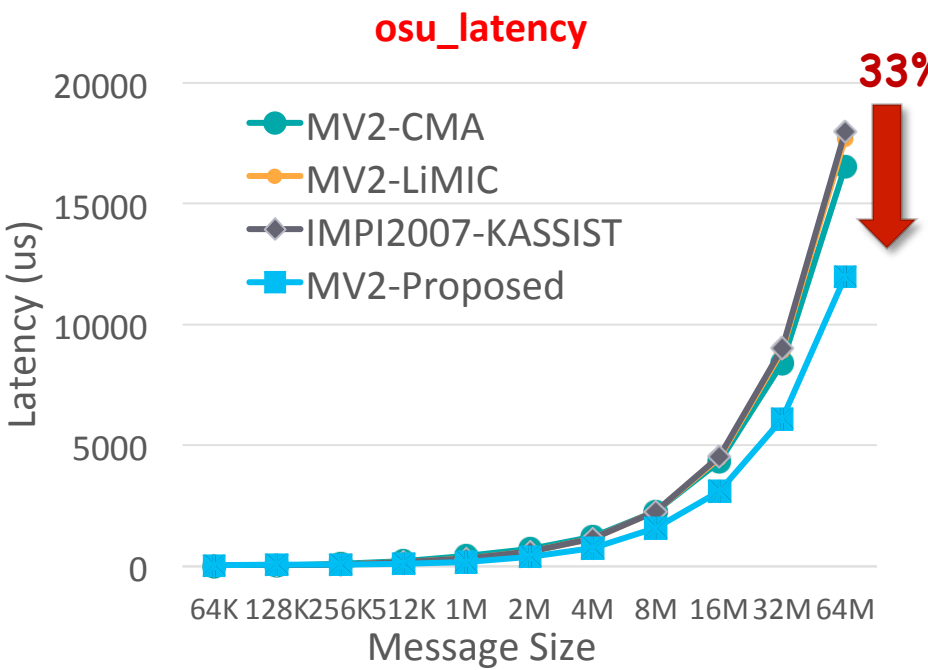
- Fast and Scalable Job Start-up
- Dynamic and Adaptive Communication Protocols and Tag Matching
- Contention-aware Designs for Intra-node Collectives
- Scalable Multi-leader Designs for Collectives
- Kernel-Assisted Communication Designs for KNL
- Efficient RMA-based Designs for Graph500 on KNL

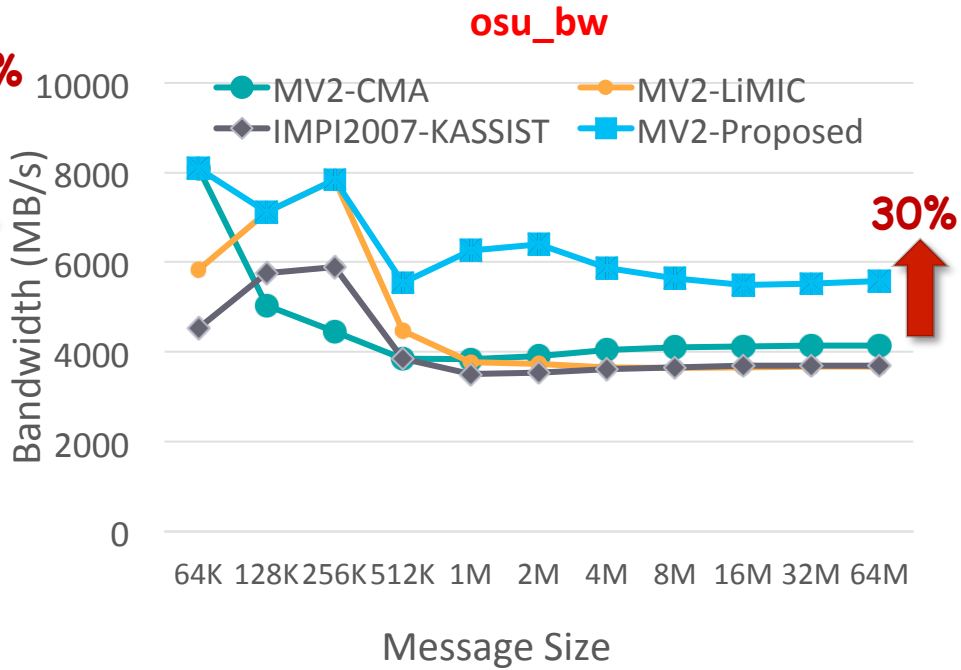
Kernel-assisted Communication Designs for KNL

- Proposed kernel-assisted on-loading communication engine for many-cores with high bandwidth memories
  - Exploits high concurrency and MCDRAM offered by KNL
- Implemented as a Linux Kernel Module with MPI as a high-level runtime
- Applicable to other programming models such as PGAS, Task-based etc.
- Provides portability, performance, and applicability to runtime as well as applications in a transparent manner
- Low latency and high throughput
  - Medium to large messages
  - Optimized for Deep Learning workloads

J. Hashmi, K. Hamidouche, H. Subramoni, D. Panda, Kernel-assisted Communication Engine for MPI on Emerging Many-core Processors, Int'l Conference on High-Performance Computing, Data, and Analytics (HiPC), Dec 2017.

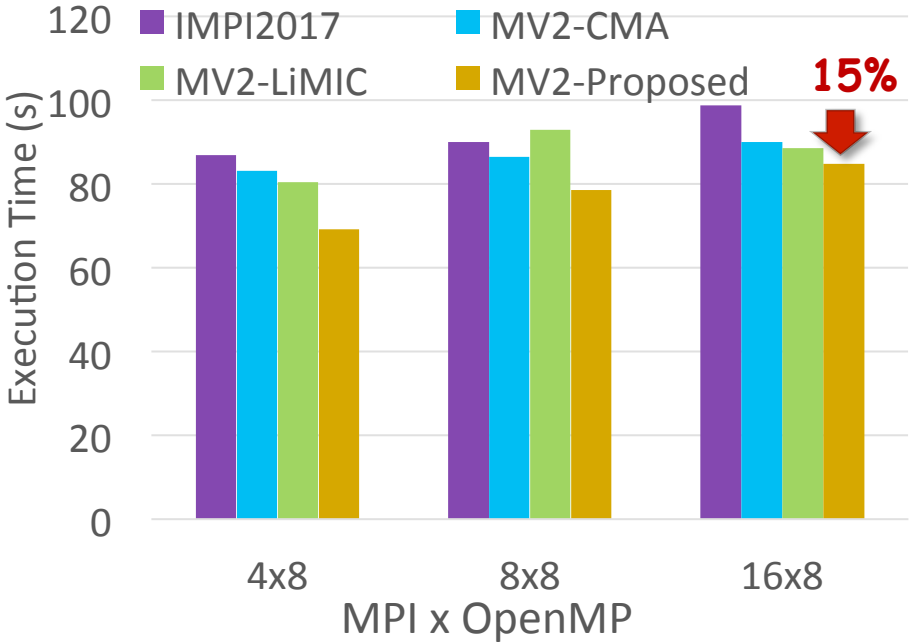
Microbenchmark Evaluation



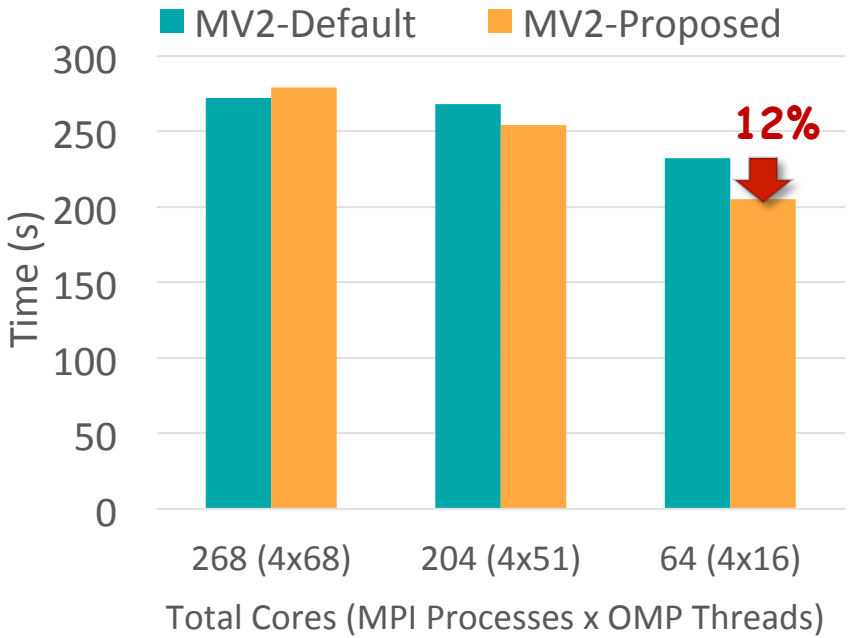


- Proposed design exploits KNL cores and MCRAM to accelerate large message transfers
- Two process latency is improved by up to 33% and bandwidth by 30%

Application Evaluations



HPCG weak-scaling execution time. [nx, ny, nz]=[104, 104,104]



CNTK: MLP Training Time using MNIST (Batch-size:64)

- 15% improvement over Intel MPI 2017 for weak-scaling HPCG benchmark
- Proposed design improved MLP training time on CNTK framework by 12%

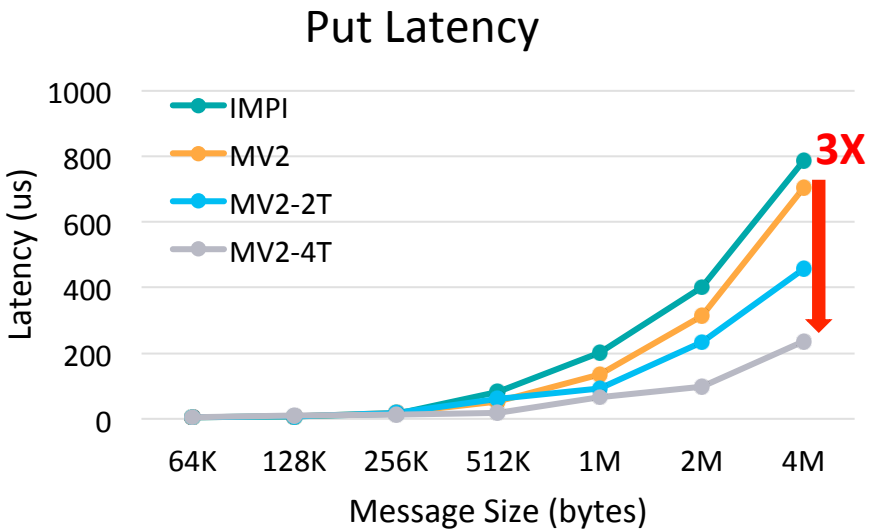
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Many-core Era

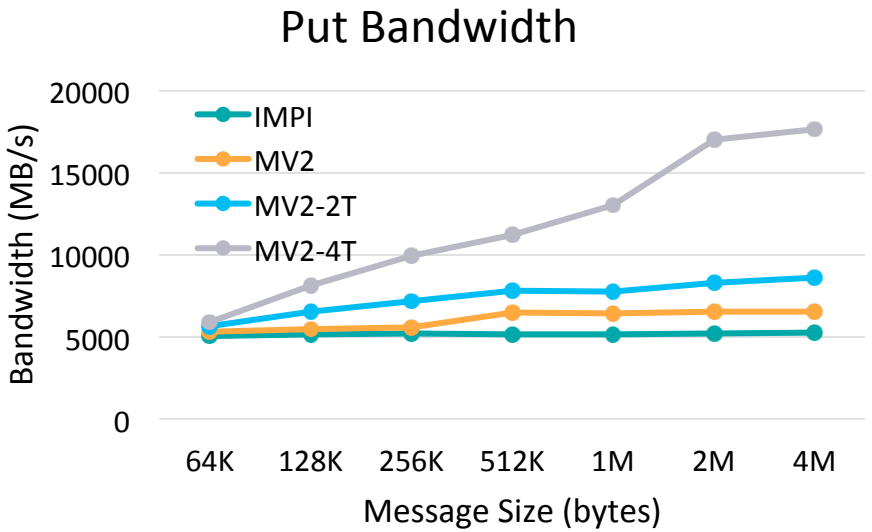
- Fast and Scalable Job Start-up
- Dynamic and Adaptive Communication Protocols and Tag Matching
- Contention-aware Designs for Intra-node Collectives
- Scalable Multi-leader Designs for Collectives
- Kernel-Assisted Communication Designs for KNL
- Efficient RMA-based Designs for Graph500 on KNL

Multi-threading and Lock-Free MPI RMA Based Graph500 on KNL

- Propose multi-threading and lock-free designs in MPI runtime as well as Graph500 to take advantage of
  - Large number of CPU hardware threads
  - High-Bandwidth Memory (HBM)

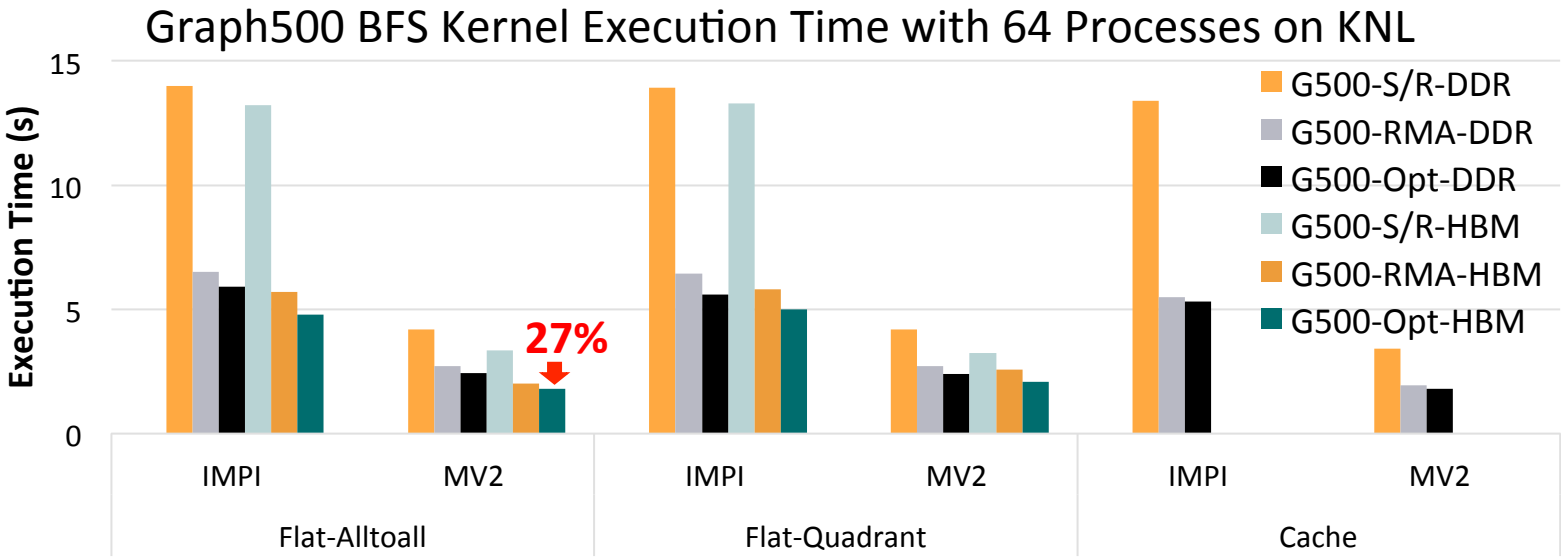
MPI Level One-sided Put Latency & Bandwidth





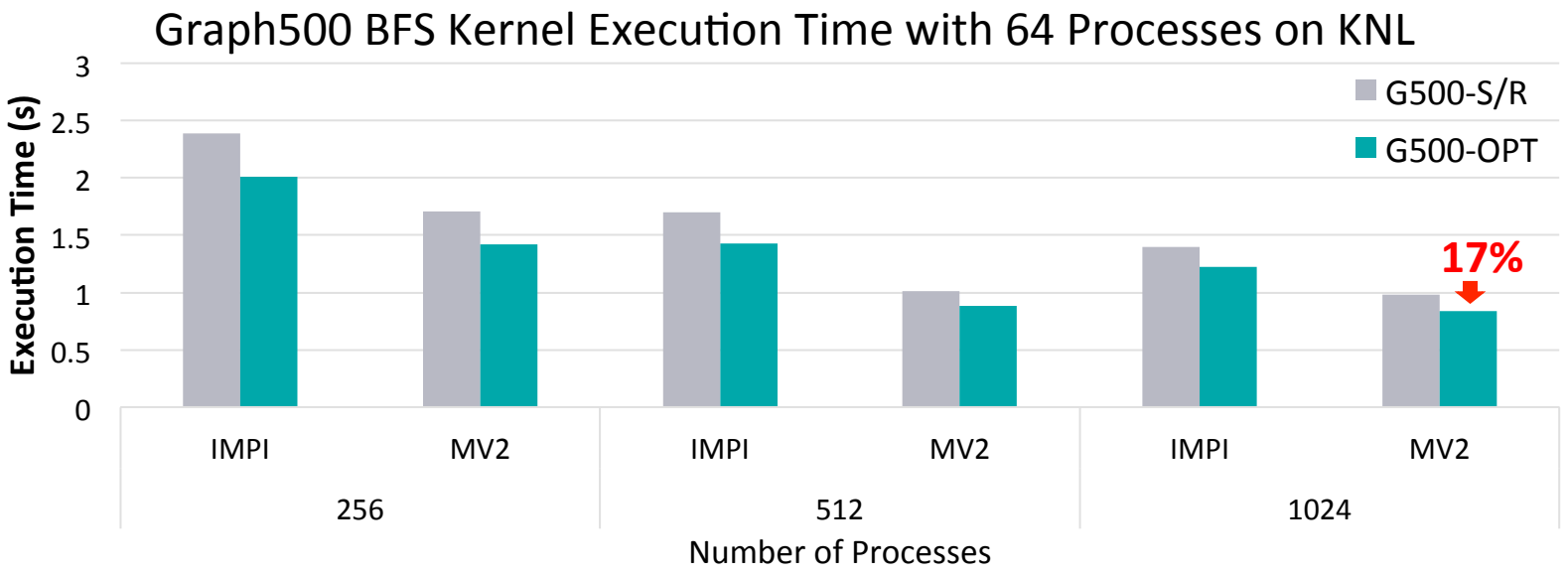
- For Put latency with 4M bytes, MV2-4T can reduce the latency by 3X

Graph500 BFS Kernel Performance Evaluation



- With Flat-Alltoall mode, G500-OPT-HBM scheme could reduce the kernel execution time by 27% compared with the G500-OPT-DDR

Graph500 BFS Kernel Scale-Out Evaluation



- With 1,024 processes, G500-OPT scheme can reduce the kernel execution time by 17% compared with the G500-S/R scheme

Concluding Remarks

- Many-core nodes will be the foundation blocks for emerging Exascale systems
- Communication mechanisms and runtimes need to be re-designed to take advantage of the availability of large number of cores
- Presented a set of novel designs and demonstrated the performance benefits
- The new designs will be available in upcoming MVAPICH2 libraries

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 Post-Docs

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

Past Research Scientist

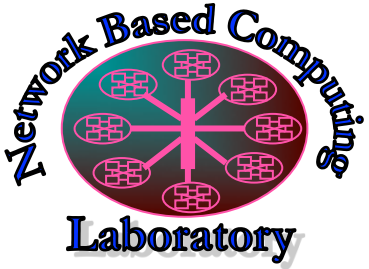
- K. Hamidouche
- S. Sur

Past Programmers

- D. Bureddy
- J. Perkins

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 Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/