Designing High Performance Shared-Address-Space and Adaptive Communication Middlewares for Next-Generation HPC Systems

Jahanzeb Maqbool Hashmi

Committee Members:

Dhabaleswar K. Panda (Advisor)
Radu Teodorescu
Feng Qin
Hari Subramoni
Danielle O. Pyun

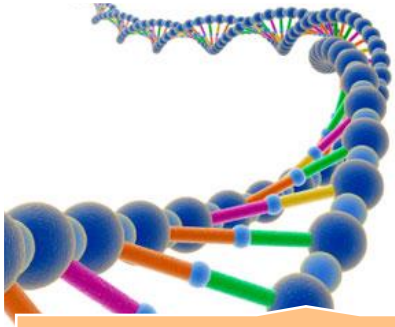
Department of Computer Science and Engineering

The Ohio State University, Columbus, USA

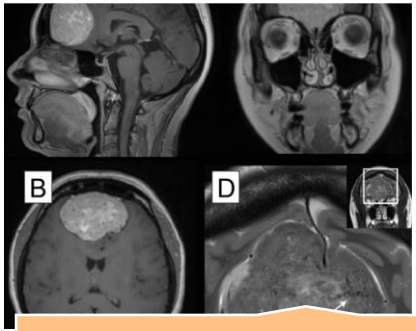
Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

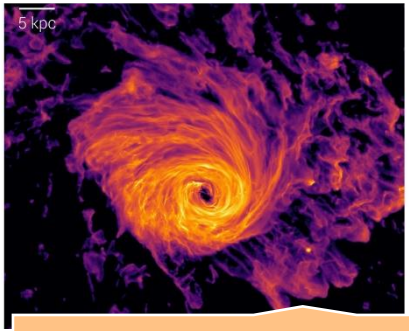
Current and Next-Generation Applications



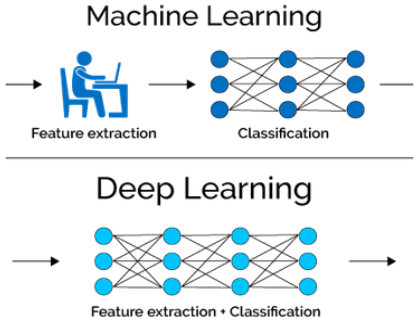
Genome Sequencing



Medical Imaging



Galaxy Formation



Distributed Deep Learning

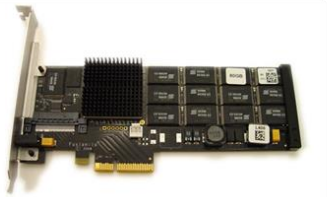
- Growth of High Performance Computing
  - Wide variety in workload
- HPC Ecosystem: Popular choice for HPC and AI
  - Scalability, Modularity, and Upgradability

Trends in Modern HPC Architecture

Multi/ Many-core Processors

High Performance Interconnects – InfiniBand, Omni-Path
<1usec latency, 100Gbps Bandwidth>

Accelerators / Coprocessors high compute density, high performance/watt



SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- High Performance Interconnects
- High Performance Storage and Compute devices
- Variety of programming models (MPI, PGAS, MPI+X)

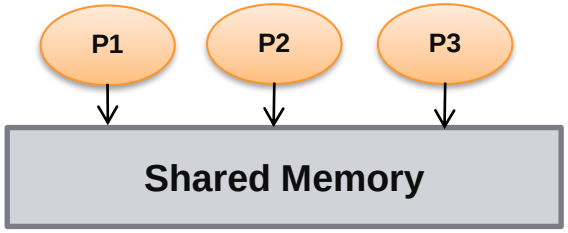
Summit

Sierra

Sunway TaihuLight

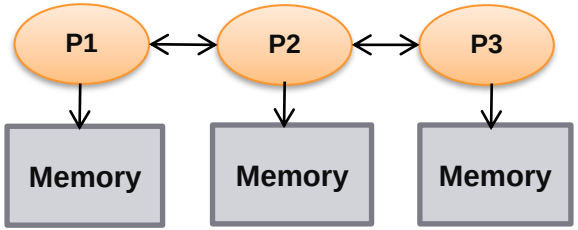
K - Computer

Parallel Programming Models Overview



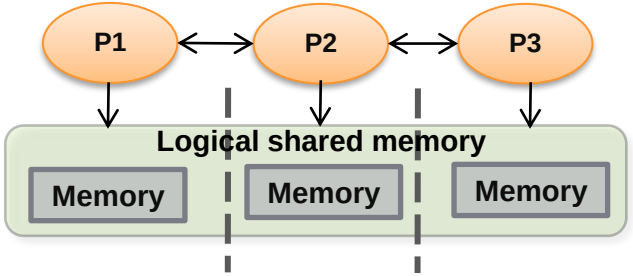
Shared Memory Model

SHMEM, DSM



Distributed Memory Model

MPI (Message Passing Interface)

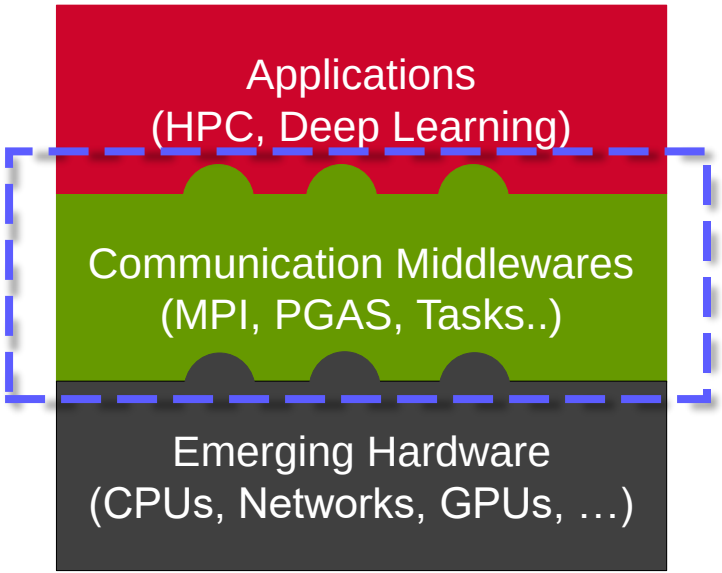


Partitioned Global Address Space (PGAS)

OpenSHMEM, UPC, UPC++, 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.
- MPI is the de-facto programming model for writing parallel applications
- MPI offers various communication primitives and data layouts
  - **Point-to-point, Collectives**, Remote Memory Access
  - **Derived Datatypes**

Role of Communication Middleware



- MPI and PGAS libraries provide a wide range of communication primitives
  - Point-to-point, Collective, One-sided
  - Blocking vs. Non-blocking
- Provide non-contiguous data-layout representations
  - MPI Derived datatypes
- Provide policies for resource mapping
  - Process-to-core mappings

- Need to consider hardware capability as well as application's needs

Challenges and Requirements

Research Challenges

- High core-density and less memory-per-core
- Emerging application requirements
  - Machine/Deep Learning
  - Scientific simulations
- Dynamic communication patterns
  - Irregular (Graphs)
  - Regular (Near-neighbor, Halo-exchange)

Design Requirements

- Efficient data-movement
- Reduced memory-footprint
- Adapt to dynamic applications and hardware topologies
- Highly Concurrent and Asynchronous communication

Broad Challenge

Designing High-performance, memory-efficient, and adaptive communication middleware for Next-generation Multi-/many-core HPC Systems

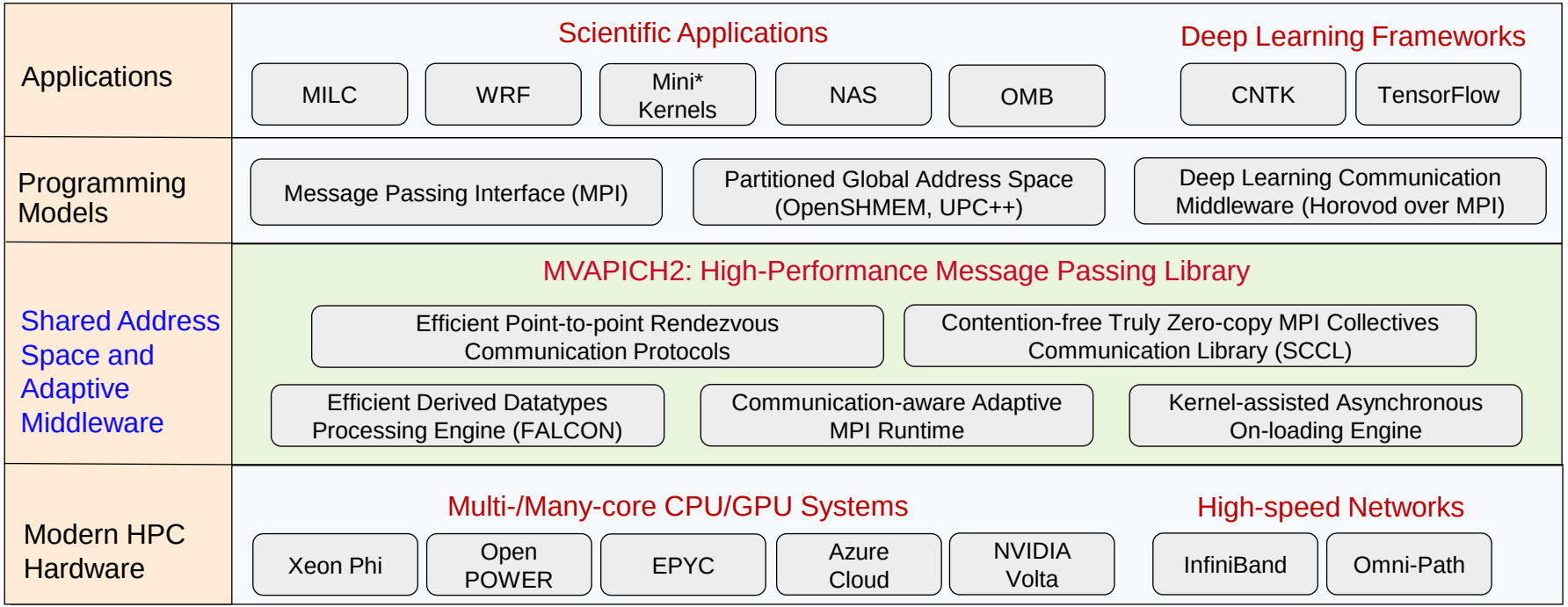
Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

Problem Statement

- Can we alleviate the bottlenecks of existing MPI intra-node designs and provide performance and memory efficient MPI primitives?
- What are the overheads associated with non-contiguous data-movement MPI and how can we alleviate these bottlenecks?
- How can we design an adaptive MPI runtime that bridges the lack of association between dynamic applications and diverse HPC systems?
- How can we leverage the high core-density of modern architectures to design asynchronous on-loading engine for MPI to achieve concurrency and overlap?
- What is the impact on real-world applications and systems?

Research Framework



Primary Publications

1. **J. Hashmi**, C. Chu, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda, FALCON-X: Zero-copy MPI Derived Datatype Processing on Modern CPU and GPU Architectures, *accepted (with minor-revision) to the Journal of Parallel and Distributed Computing*, (**JPDC '20**)

2. **J. Hashmi**, S. Xu, B. Ramesh, M. Bayatpour, H. Subramoni, and D. K. Panda, Machine-agnostic and Communication-aware Designs for MPI on Emerging Architectures, *in Proceeding of the 34th IEEE Intl' Parallel and Distributed Processing Symposium* (**IPDPS '20**)

3. **J. Hashmi**, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda, FALCON: Efficient Designs for Zero-copy MPI Datatype Processing on Emerging Architectures, *in Proceeding of the 33rd IEEE Intl' Parallel and Distributed Processing Symposium* (**IPDPS '19**), **Best Paper Finalist**

4. **J. Hashmi**, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda, Design and Characterization of Shared Address Space MPI Collectives on Modern Architectures, *in Proceeding of the 19th IEEE/ACM Intl' Symposium on Cluster, Cloud, and Grid Computing* (**CCGrid '19**)

5. **J. Hashmi**, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda, Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores, *in Proceedings of the 32nd IEEE Intl' Parallel and Distributed Processing Symposium* (**IPDPS '18**)

6. **J. Hashmi**, M. Li, H. Subramoni, and D. K. Panda, Designing High-Performance and Scalable Collectives for the Many-core Era: The MVAPICH2 Approach, *Intel eXtreme Performance User's Group* (**IXPUG '18**)

Primary Publications (Cont'd)

7. **J. Hashmi**, K. Hamidouche, H. Subramoni, and D. K. Panda, Kernel-assisted Communication Engine for MPI on Emerging Manycore Processors, *in Proceedings of the 24th IEEE Intl' Conference on High Performance Computing, Data, and Analytics* (**HiPC '17**)

8. **J. Hashmi**, M. Li, H. Subramoni, and D. K. Panda, Exploiting and Evaluating OpenSHMEM on KNL Architecture, *in Proceedings of the Fourth Workshop on OpenSHMEM and Related Technologies* (**OpenSHMEM'17**)

9. **J. Hashmi**, M. Li, H. Subramoni, and D. K. Panda, Performance of PGAS Models on KNL: A Comprehensive Study with MAPICH2-X, *at Intel eXtreme Performance User's Group* (**IXPUG '17**)

10. **J. Hashmi**, K. Hamidouche, and D. K. Panda, Enabling Performance Efficient Runtime Support for Hybrid MPI+UPC++ Programming Models, *in Proceedings of the 18th IEEE Intl' Conference on High Performance Computing and Communications* (**HPCC '16**)

Secondary Publications

1. M. Bayatpour, **J. Hashmi**, S. Chakraborty, K. Kandadi Suresh, M. Ghazimirsaeed, H. Subramoni, and D. K. Panda, Communication-Aware Hardware-Assisted MPI Overlap Engine, *in Proceeding of the 2020 Intl' Supercomputing Conference* (**ISC '20**)
2. K. Kandadi Suresh, B. Ramesh, M. Ghazimirsaeed, M. Bayatpour, **J. Hashmi**, H. Subramoni, and D. K. Panda, Performance Characterization of Network Mechanisms for Non-Contiguous Data Transfers in MPI, *in Proceeding of the Annual SNACS Workshop, held in conjunction with the 34th IEEE Intl' Parallel and Distributed Processing Symposium* (**SNACS @ IPDPS '20**)
3. C. Chu, **J. Hashmi**, K. Shafie Khorassani, H. Subramoni, and D. K. Panda, High-Performance Adaptive MPI Derived Datatype Communication for Modern Multi-GPU Systems, *in Proceedings of the 26th IEEE Intl' Conference on High Performance Computing, Data, Analytics, and Data Science* (**HiPC '19**)
4. X. Xu, **J. Hashmi**, S. Chakraborty, H. Subramoni, and D. K. Panda, Design and Evaluation of Shared Memory Communication Benchmarks on Emerging Architectures using MVAPICH2, *in Proceeding of the 3rd Annual IPDRM Workshop, held in conjunction with Supercomputing 2019* (**IPDRM @ SC '19**)
5. A. Ruhela, B. Ramesh, S. Chakraborty, H. Subramoni, **J. Hashmi**, and D. K. Panda, Leveraging Network-level Parallelism with Multiple Process-Endpoints for MPI Broadcast, *in Proceeding of the 3rd Annual IPDRM Workshop, held in conjunction with Supercomputing 2019* (**IPDRM @SC '19**)

Secondary Publications (Cont'd)

6. M. Bayatpour, **J. Hashmi**, S. Chakraborty, H. Subramoni, and D. K. Panda, SALaR: Scalable and Adaptive Designs for Large Message Reduction Collectives, *in Proceedings of the 2018 IEEE Intl' Conference on Cluster Computing* (**CLUSTER '18**), **Best Paper Award**

7. S. Chakraborty, M. Bayatpour, **J. Hashmi**, H. Subramoni, and D. K. Panda, Cooperative Rendezvous Protocols for Improved Performance and Overlap*, in Proceedings of the 2018 Intl' Conference for High Performance Computing, Networking, Storage and Analysis* (**SC '18**), **Best Student Paper Finalist**

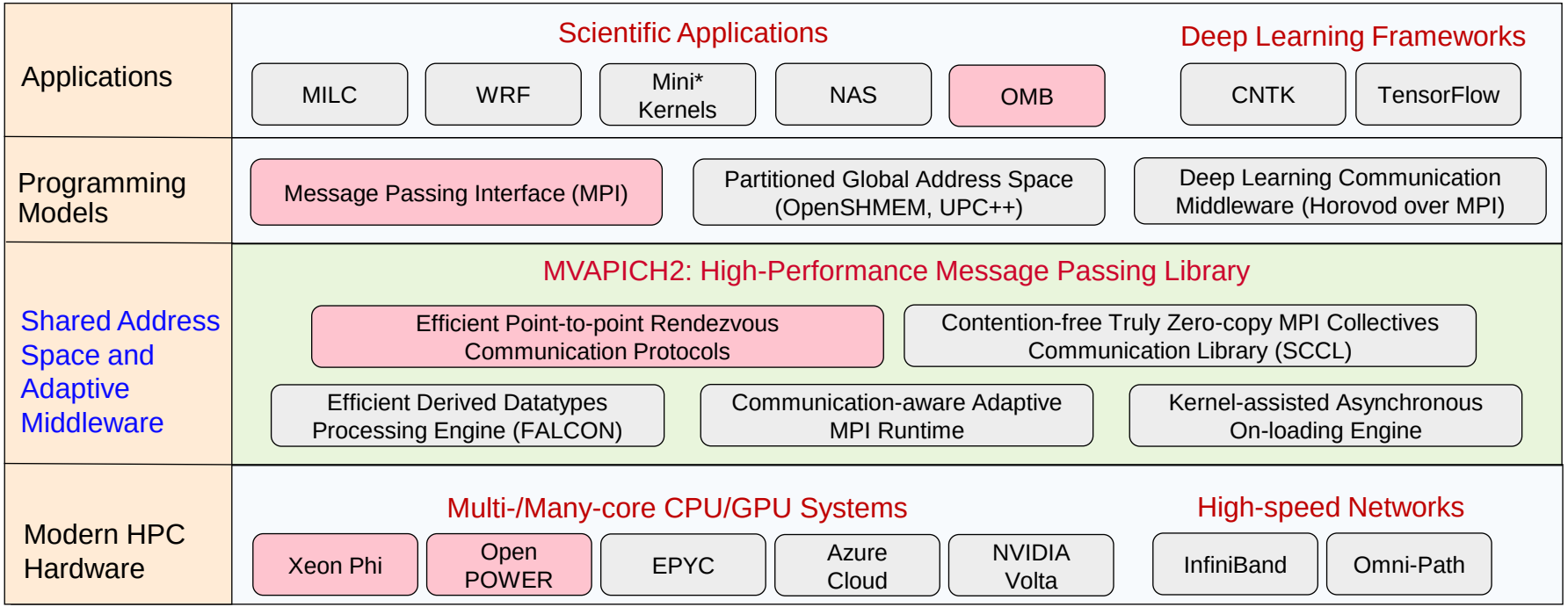
8. A. Awan, K. Hamidouche, **J. Hashmi**, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters, *in Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (**PPoPP '17**)

9. C. Chu, X. Lu, A. Awan, H. Subramoni, **J. Hashmi**, B. Elton, and D. K. Panda, Efficient and Scalable Multi-Source Streaming Broadcast on GPU Clusters for Deep Learning, *in Proceedings of the 2017 Intl' Conference on Parallel Processing* (**ICPP '17**)

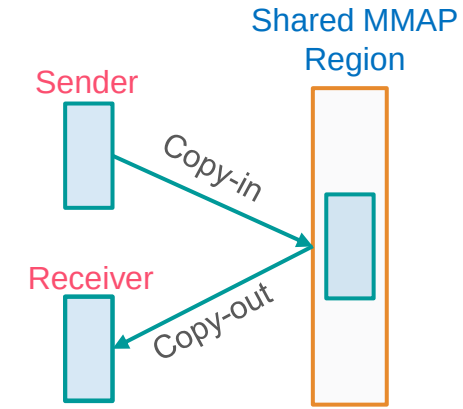
Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

Research Framework

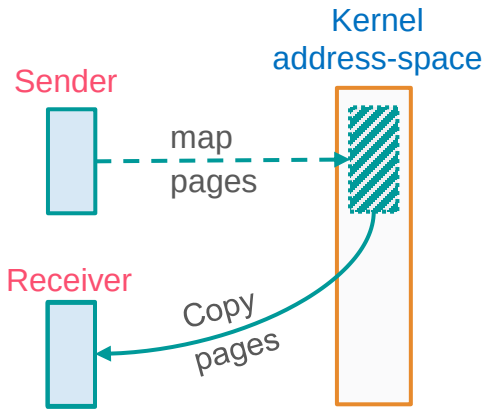


Challenges in Intra-node Point-to-point Communication in MPI



Shared Memory (POSIX)

Requires two copies
No system call overhead
Better for Small Messages

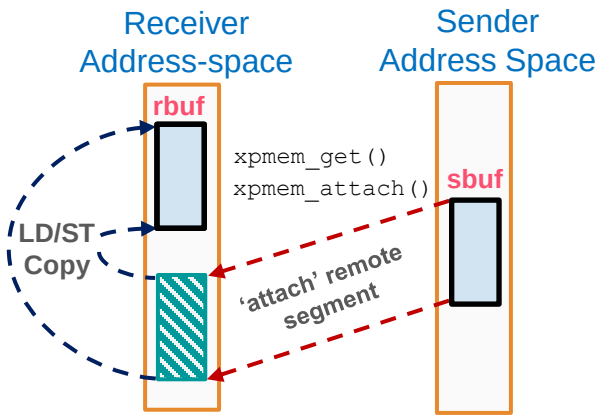


Kernel-mapping (CMA/LiMIC/KNEM)

System call overhead
Lack of Load/store access
single (a.k.a “zero”) copy
Better for Large Messages

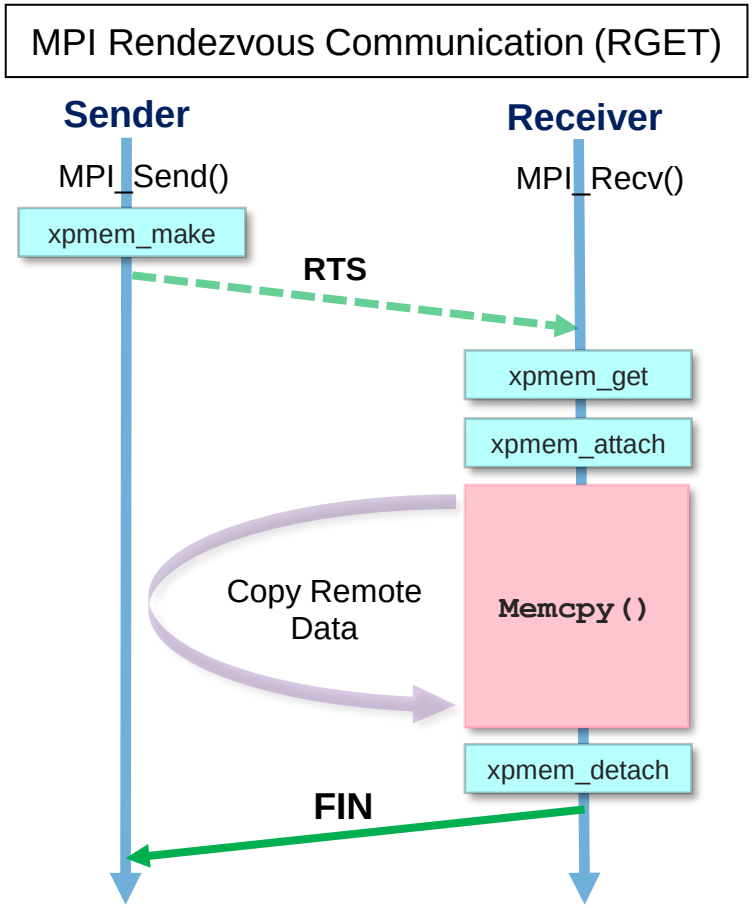
We require over-head free, user-space, load/store based inter-process communication mechanism, also called “**Shared Address Space**” communication

Shared Address Space MPI Communication using XPMEM



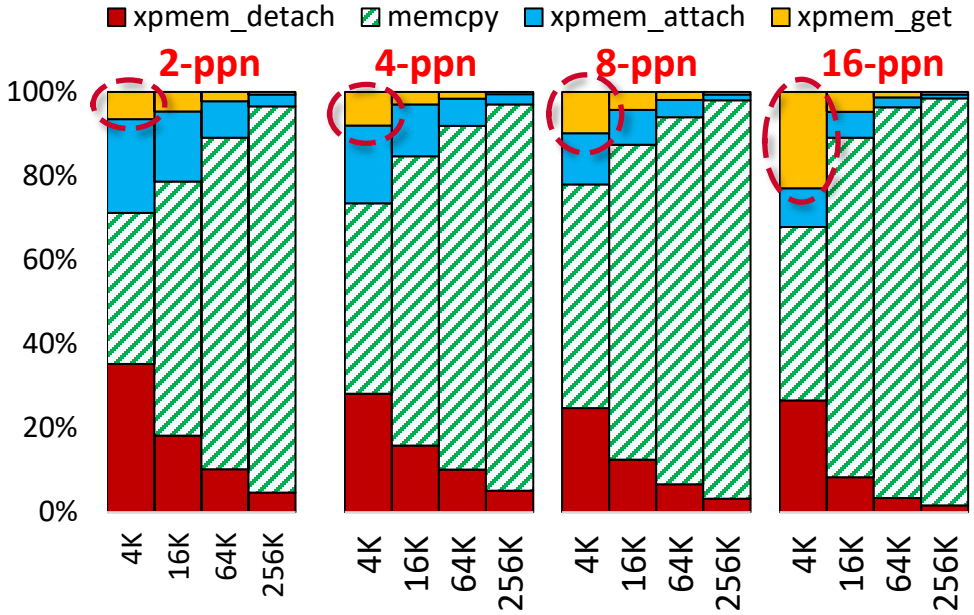
Cross-partition Memory (XPMEM)

- Kernel Module with user-space API
- Allows a process to "*attach*" to the virtual memory segment of a remote process



Limitations of Shared Address Space Communication

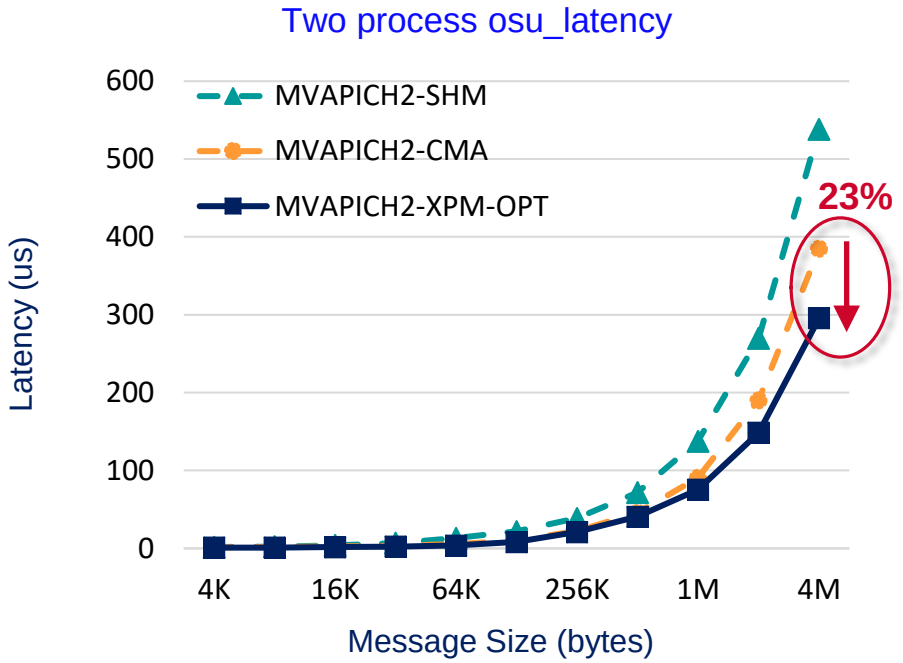
- Experiment:
  - XPMEM based one-to-all latency benchmark
  - All processes read from a root process' buffer
- Key observation:
  - Up to **65%** time spent in XPMEM registration for short message (4K)
  - **Lock contention** with increasing PPN
  - Vanilla shared address space design is suboptimal for MPI communication



Relative costs of XPMEM API functions for different PPN using one-to-all communication benchmark on a single dual-socket Broadwell node with 14 cores.

Proposed Registration Cache for XPMEM based Communication

- Broad Idea:
  - Memory de-registration is delayed
  - Maintain a cache of remote *attached* pages
  - Detach pages only in *MPI_Finalize()* or when capacity-miss occurs (FIFO)
  - MPI calls on the same buffers always hit!
- A new problem?
  - Multiple calls to malloc/free on remote buffers
- Solution:
  - Interception of malloc/free calls to invalidate remote mappings



Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores

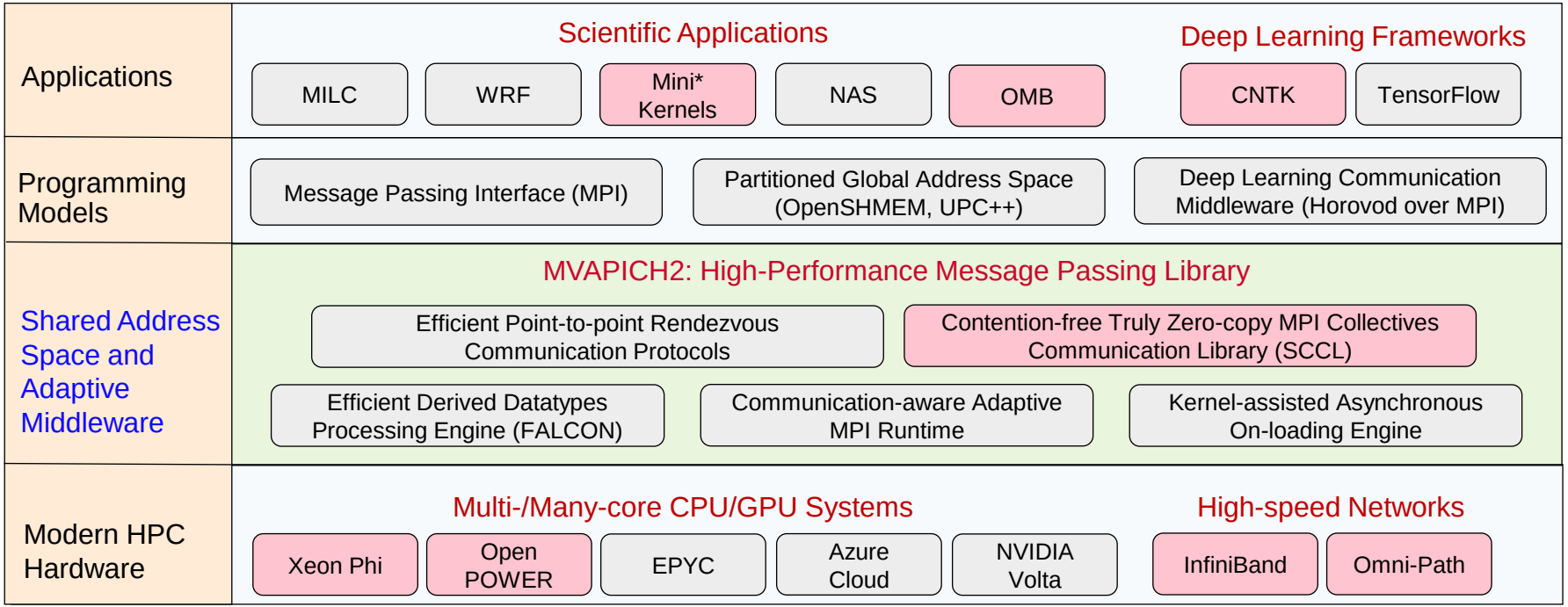
J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda

32nd IEEE IPDPS '18

Overview

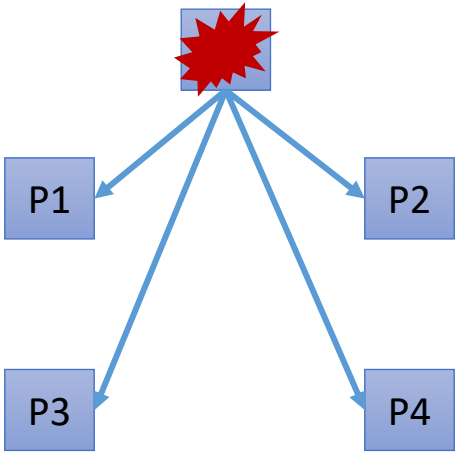
- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

Research Framework



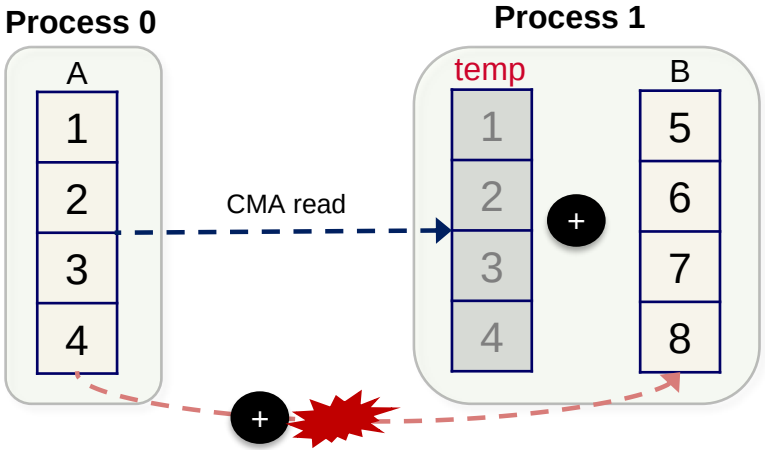
Problems with State-of-the-art Zero-copy Collectives

Contention Problem



- One-to-all or all-to-one patterns
  - CMA Broadcast, CMA Scatter, etc.
- Kernel-level contention
  - CMA relies on get_user_pages()
  - Page table lock at target process

Reduction Problem



- No load/store access via CMA
- Remote data has to be brought to local memory before operation

[1] 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

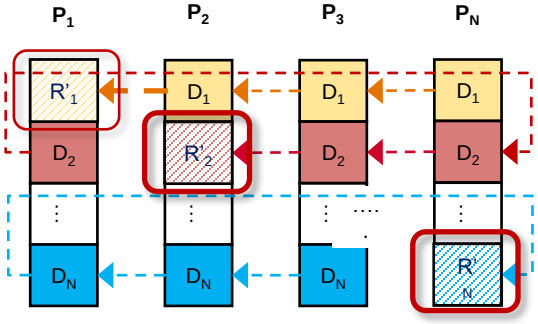
Proposed Shared Address Space Collective Communication Library (SCCL)

- Step-1:
  - Ranks exchange buffer information
  - Tuple of `<vaddr, len, segid>`
- Step-2:
  - Peer ranks map remote memory segments
  - Add entry to cache if not found
- Step-3:
  - Intra-node barrier to enforce ordering
- Step-4:
  - Plug our proposed XPMEM based collective implementation routines e.g.,
  - `MV2_XPMEM_Direct_<collective-name>`
  - Implemented all MPI collectives

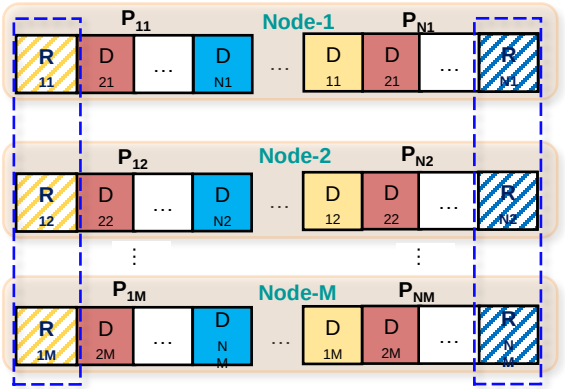
```
/* Share vaddr with peer ranks */
Exchange_buffer_addresses();                    ▷ Step-1
  /* Create remote buffers mapping */;          ▷ Step-2
foreach rem_rank in SMP rank list do
    if rank ≠ local then
        Dreg_entry d;
        /* Find in local registration cache */
        d ← AVL_lookup(rem_rank, rbuf, len);
        if found then
            return d;
        else
            /* create remote page mappings */
            d ← XPMEM_Attach(rbuf, len);
            /* Cache dreg entry in local tree */
            AVL_insert(d, avl_roots[rem_rank]);
            return d;
        end
    end
end
synchronize();                                  ▷ Step-3
/* Call direct Load/Store based algorithm */
MV2_XPMEM_Direct_coll*(...);                    ▷ Step-4
```

High-level Overview of XPMEM base Direct MPI Collectives Implementation

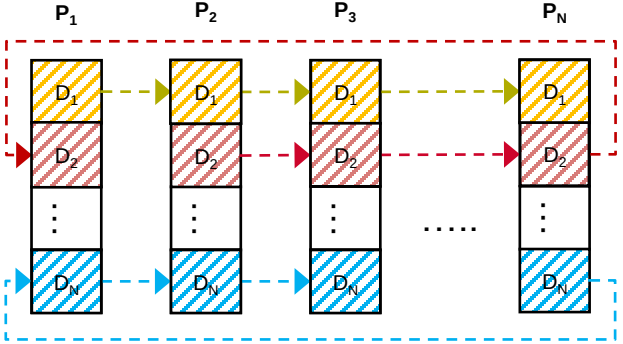
Proposed Contention-free and Zero-copy MPI_Allreduce



Step-1: Parallel Intra-node Partitioned Reduce



Step-2: Multi-root Inter-node Allreduce



Step-3: Parallel Intra-node Partitioned Broadcast

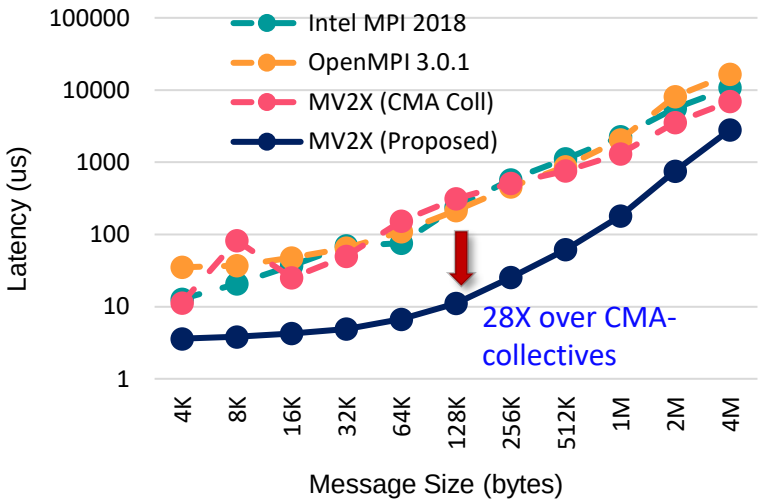
Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores

J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda

32nd IEEE IPDPS '18

Performance of Contention-free Zero-copy Collectives

Benefits of "Contention-free" MPI_Scatter on OMB

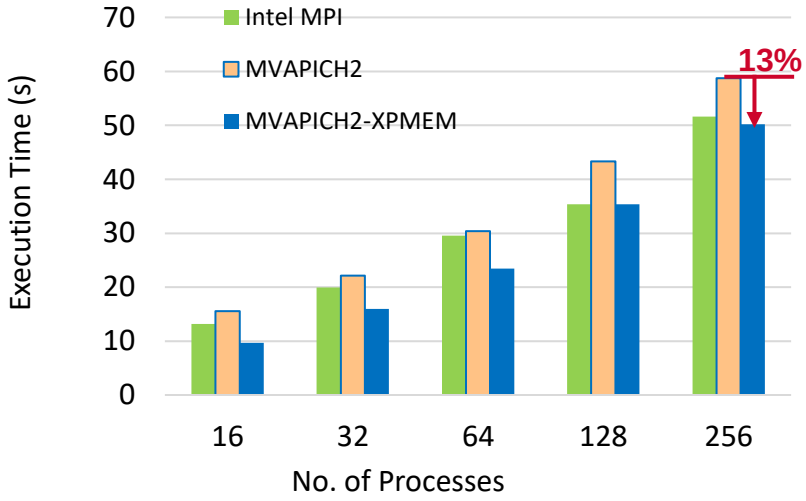


Design and Characterization of Shared Address Space MPI Collectives on Modern Architectures

J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda

19th IEEE CCGRID '19

Benefits of "Truly Zero-copy Allreduce" on MiniAMR



Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores

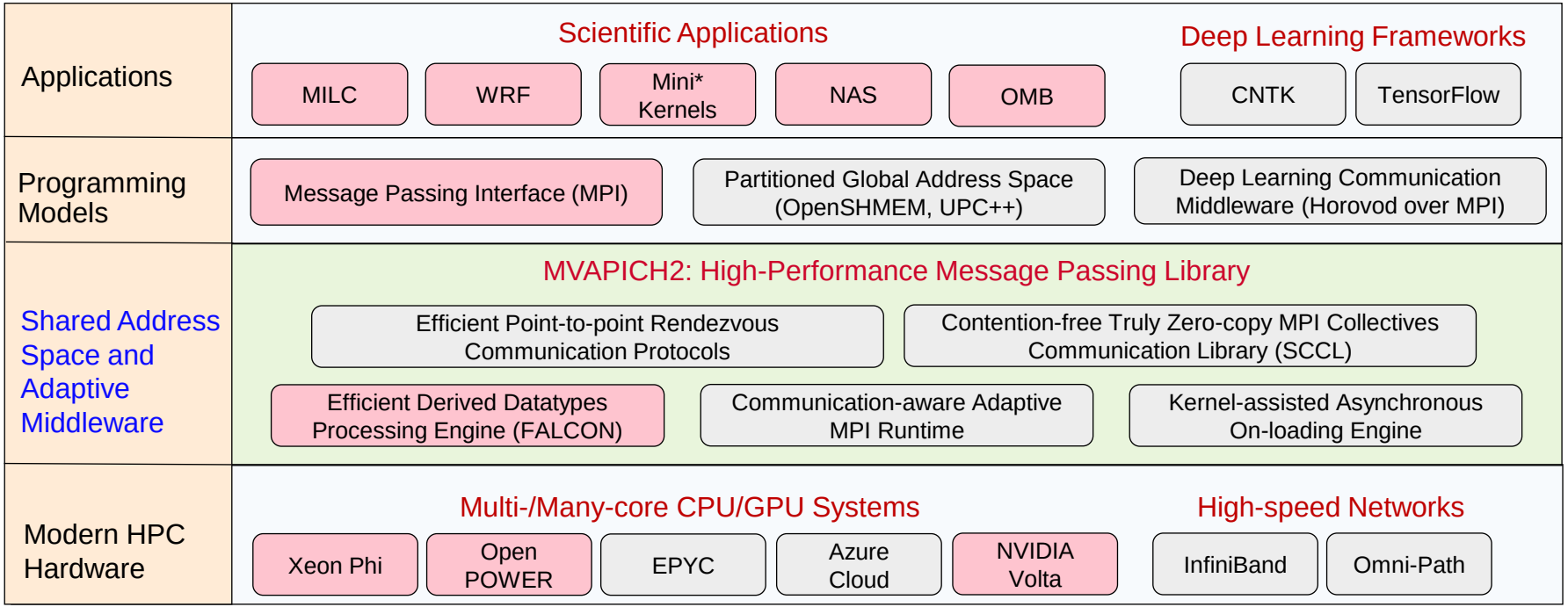
J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda

32nd IEEE IPDPS '18

Overview

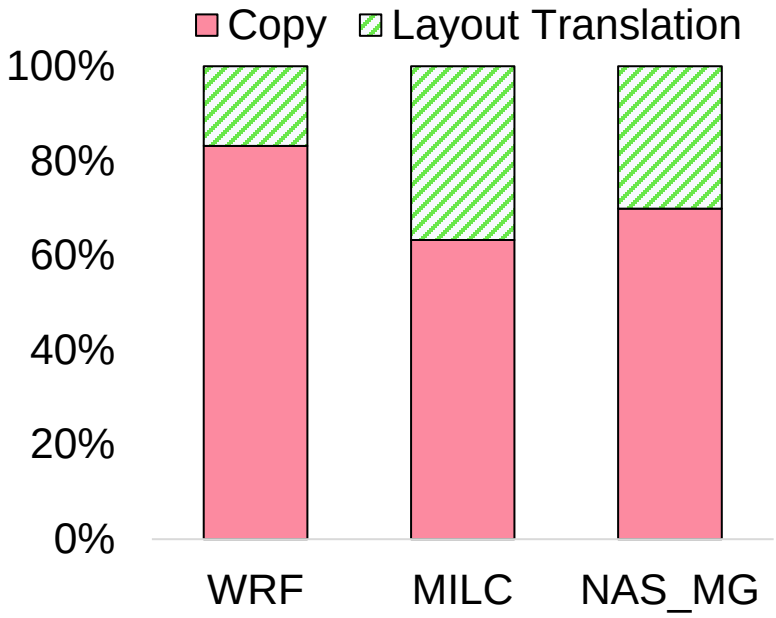
- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

Research Framework



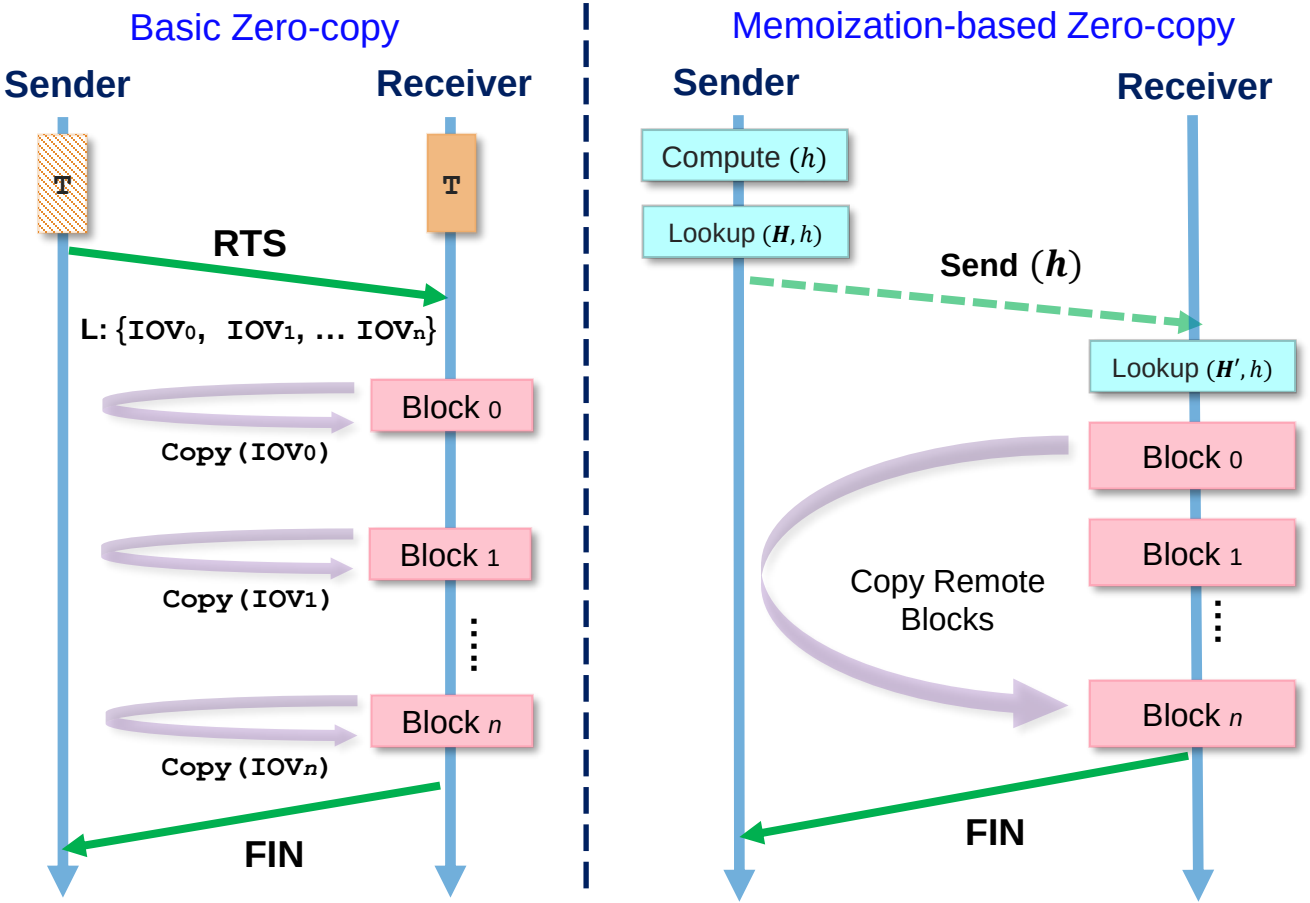
Limitations of Existing Pack/Unpack based Designs

- Design Challenges
  - Flattening the layout into list of I/O vectors (Layout Translation)
  - Significantly slow for nested datatypes
  - **Pack/Unpack** requires two copies
    - 2X overhead for large messages!!
- Proposed Solution:
  - **FALCON — FA**st and **L**ow-overhead **Zero-copy** MPI datatype processing **CO**mmunication e**N**gine
  - Amortized layout translation
  - Zero-copy non-contiguous data movement



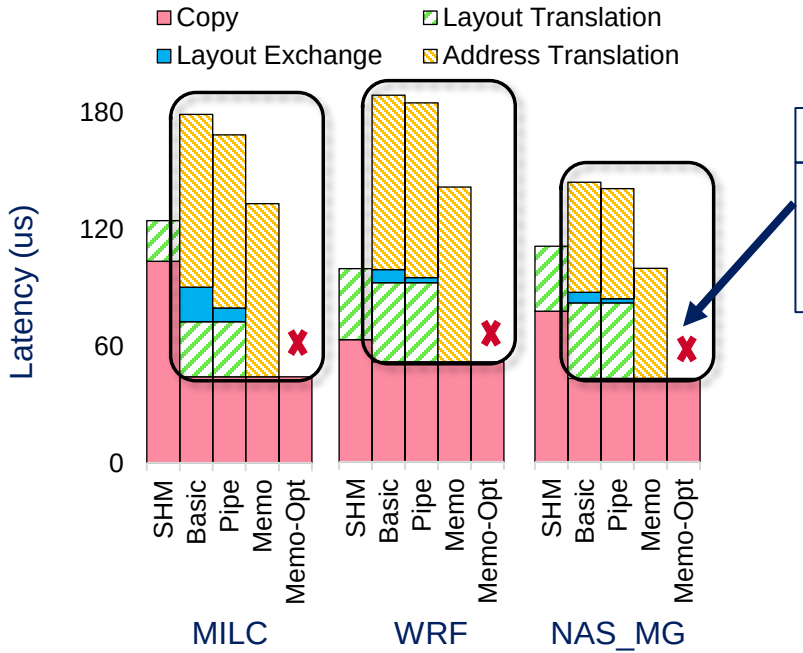
Cost breakdown of existing Pack/Unpack designs on Broadwell

Proposed Designs: FALCON



- **Avoiding Remote Virtual Address Translation**
  - Attached segments are cached
- **Communication pattern as input to the Hash function**
  - Request object has enough information
  - *<Datatype, Count, Destination Rank, Tag, Communicator>*

Impact of Optimized Memoization based Zero-copy Design



Optimized Memoization Design (Final)

- ~~Layout exchange overheads~~
- ~~Layout translation overheads~~
- ~~Address translation overheads~~

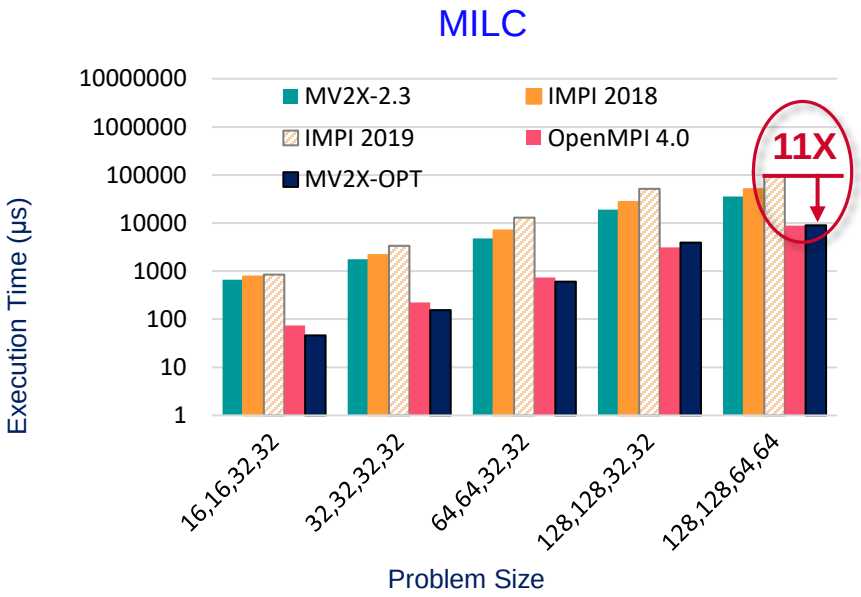
No additional Overhead!!

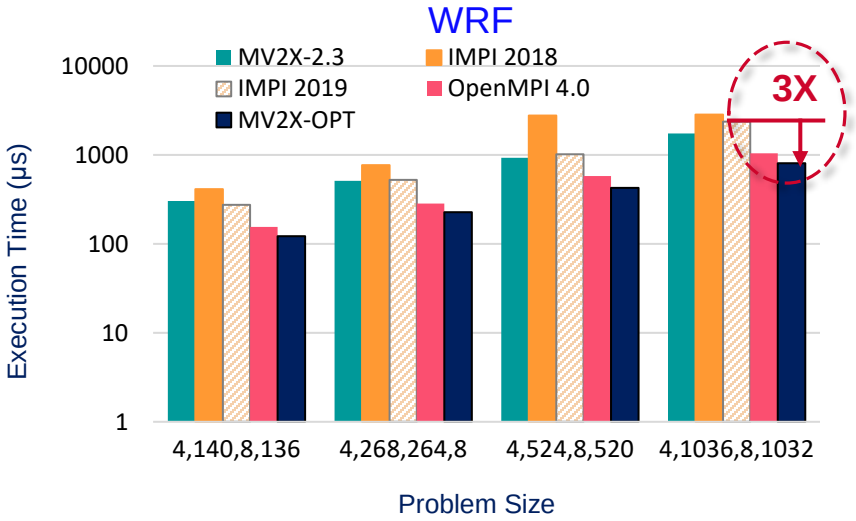
FALCON: Efficient Designs for Zero-copy MPI Datatype Processing on Emerging Architectures

J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda,

IEEE IPDPS '19, **Best Paper Finalist**

Performance Results on Application Kernels (Broadwell)





- On MILC, for Problem-B (768-KB), up to **11X** over IMPI 2019
- On Broadwell, up to **2.1X** and **3X** improved latency over MVAPICH2-X and Intel MPI 2019

FALCON: Efficient Designs for Zero-copy MPI Datatype Processing on Emerging Architectures

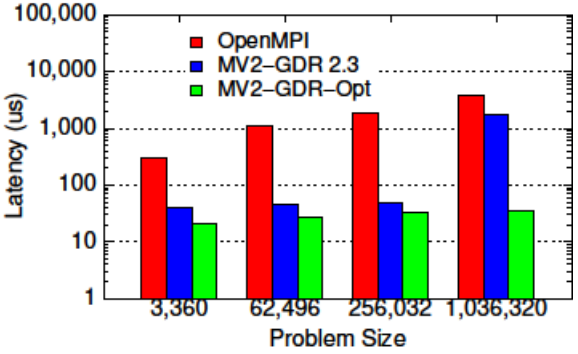
J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda,

IEEE IPDPS '19, **Best Paper Finalist**

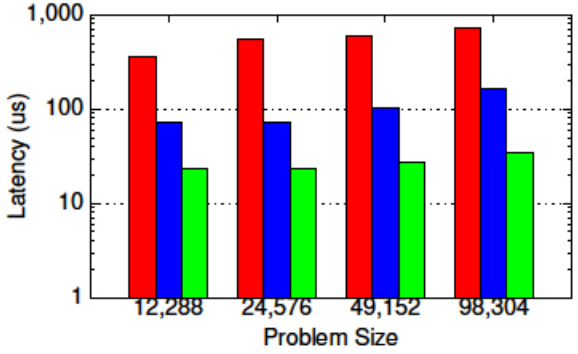
Extending FALCON for Dense Multi-GPU Systems (FALCON-X)

- Common issues in CUDA-aware MPI for Non-contiguous datatypes
- **FALCON-X**: Efficient MPI derived datatype for dense multi-GPU systems e.g., DGX-2
  - Extends the ideas of FALCON for GPU systems using CUDA IPC

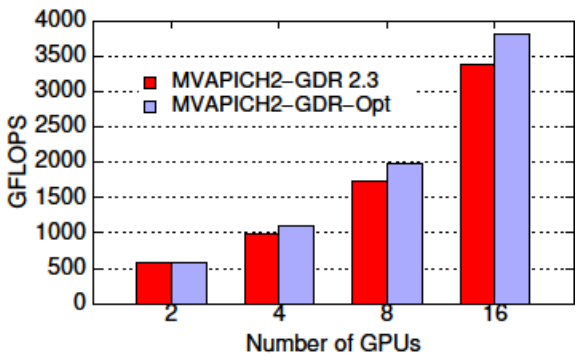
NAS_MG kernel on 2x Volta GPUs



MILC kernel on 2x Volta GPUs



Scaling 2D-stencil on DGX-2



FALCON-X: Zero-copy MPI Derived Datatype Processing on Modern CPU and GPU Architectures

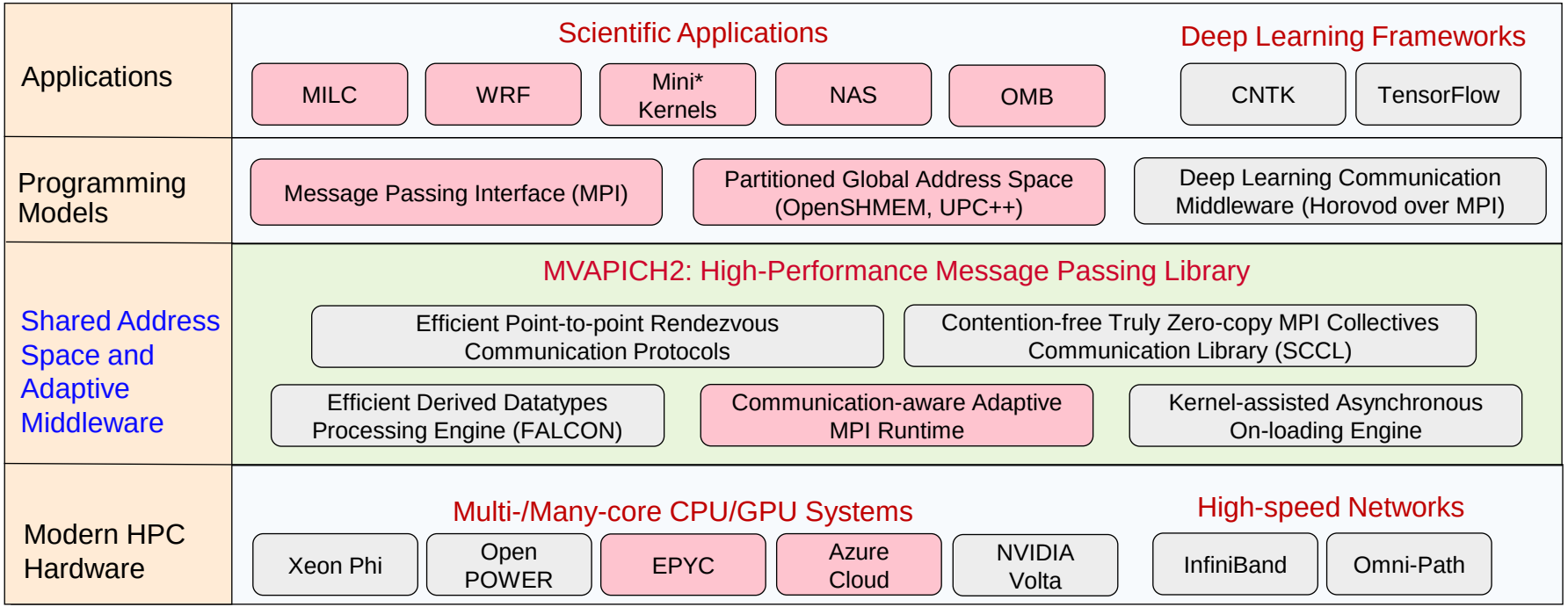
J. Hashmi, C. Chu, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. K. Panda,

Journal of Parallel and Distributed Computing **(JPDC '20)**

Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

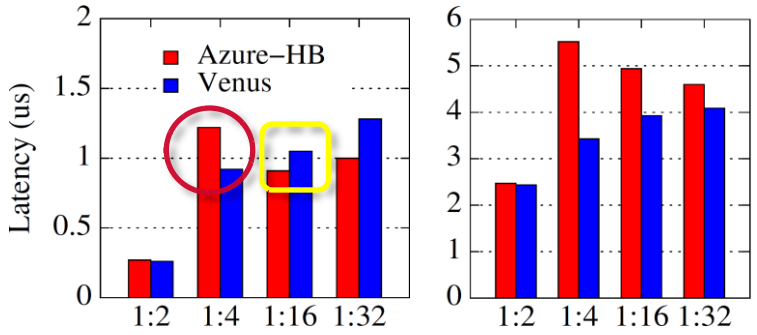
Research Framework



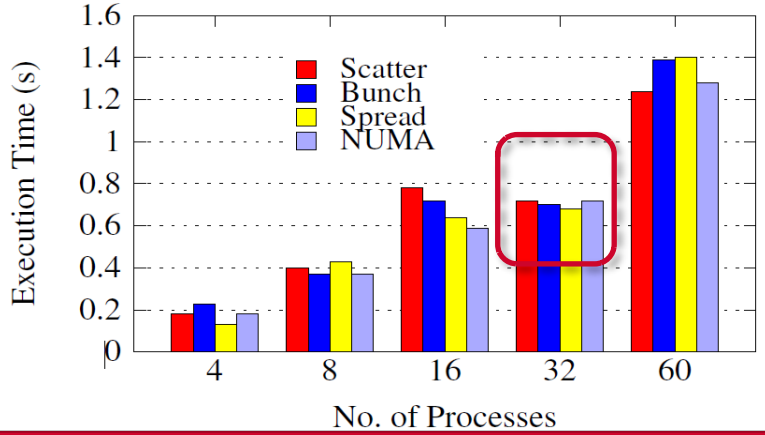
Challenges with Diverse Architectures and Applications

AMD EPYC 7551 Architecture

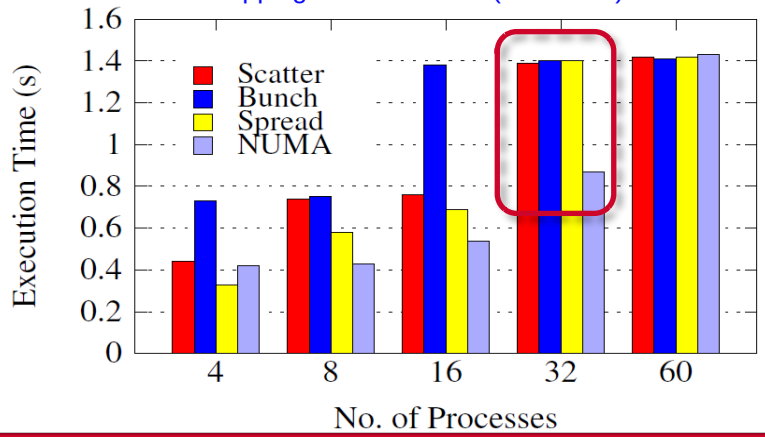
Latency test with various Rank-to-core placements with small (left) and large (right) messages



MPI Mappings on MiniAMR (Azure-HB)



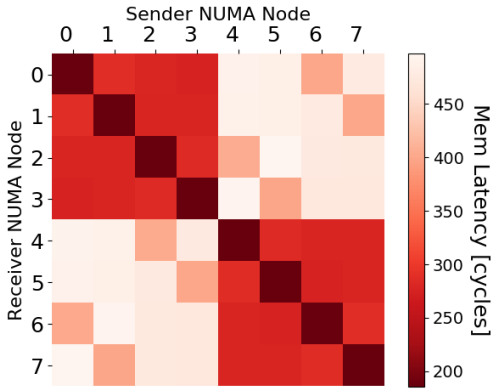
MPI Mappings on MiniGhost (Azure-HB)



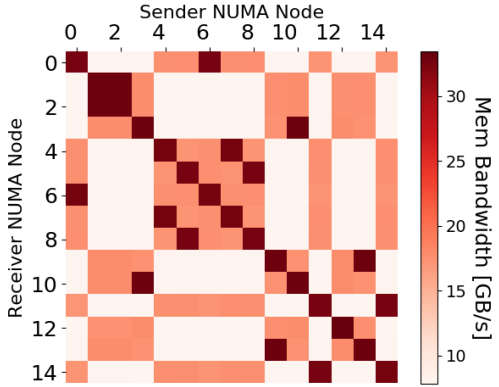
Proposed Design for Constructing Machine Topologies

- Low-level benchmarking approach to construct physical to virtual resource graph
  - Use `MCTOP`[2] for cache-line level measurements
  - Works regardless of the native and VM systems

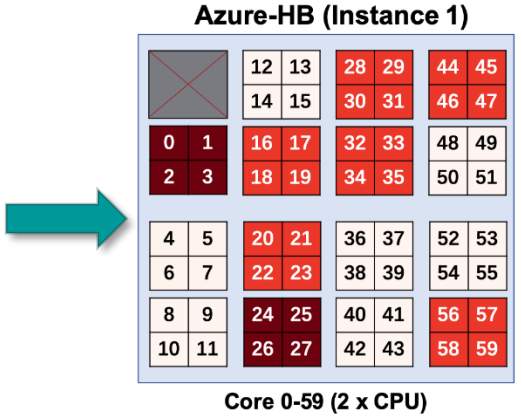
NUMA-NUMA Bandwidth (Native AMD EPYC)



NUMA-NUMA Bandwidth (Azure AMD VM)



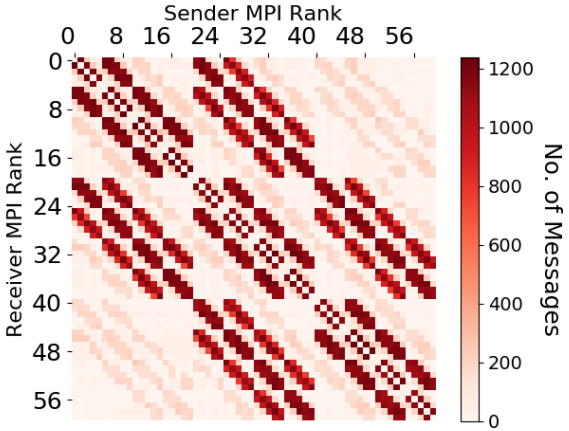
Generated Topology Graph



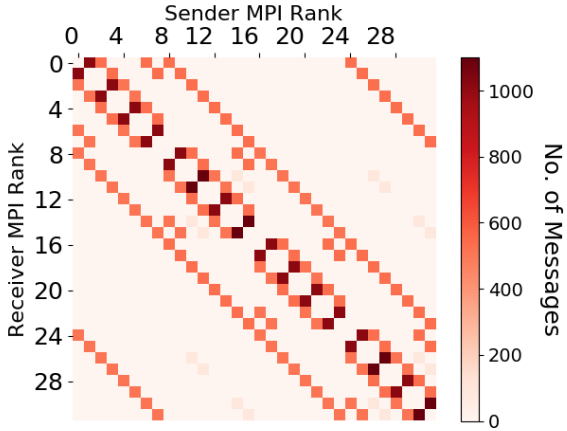
[2] "Abstracting Multi-Core Topologies with MCTOP", Georgios et al. EuroSys'17

Proposed Efficient MPI Rank to Core Mappings

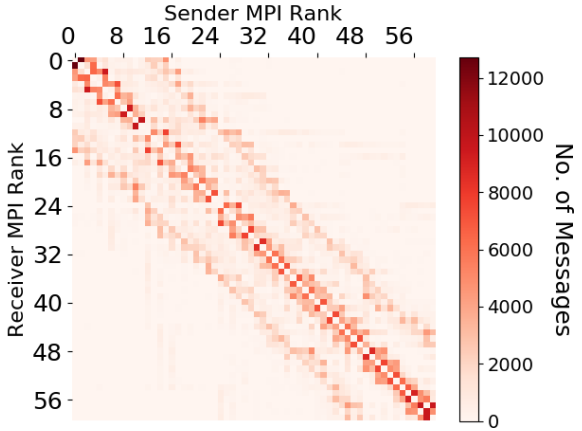
Communication Pattern (AMG)



Communication Pattern (NAS_MG)

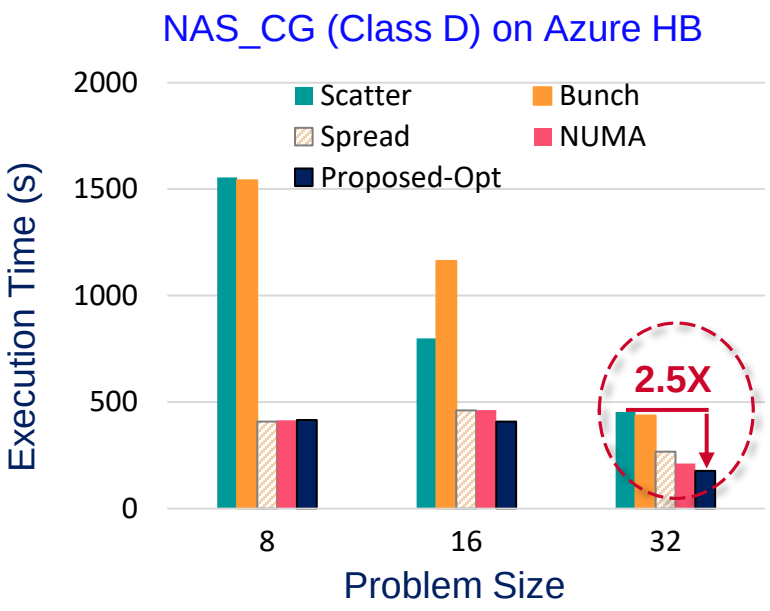


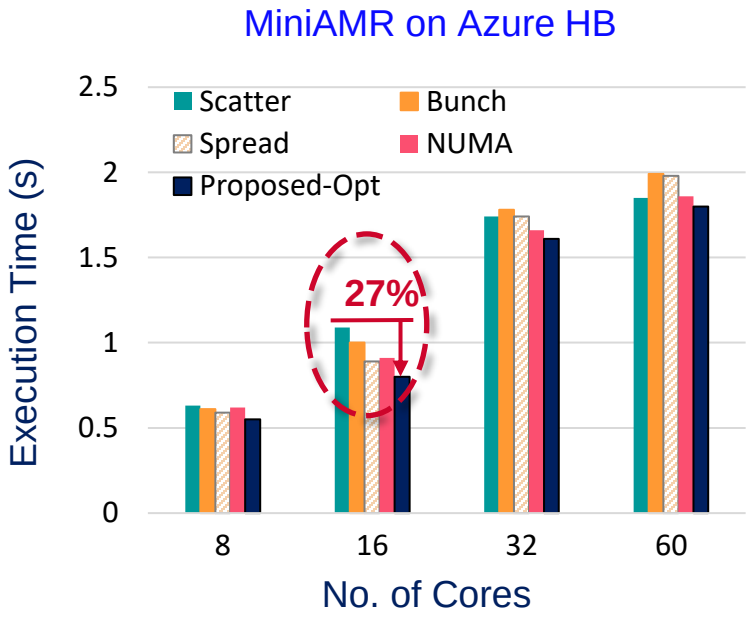
Communication Pattern (MiniAMR)



- **Efficient virtual to physical resource mapping algorithm**
  - Step 1: Construct topology graph (G) using MCTOP
  - Step 2: Adaptively generate communication graph (G') using MPI_T interception (on-the-fly)
  - Step 3: Greedy style algorithms to map high-cost edges in G' on to low-latency/high-bandwidth edges in G

Application-level Benefits of Adaptive Designs





Machine-agnostic and Communication-aware Designs for MPI on Emerging Architectures

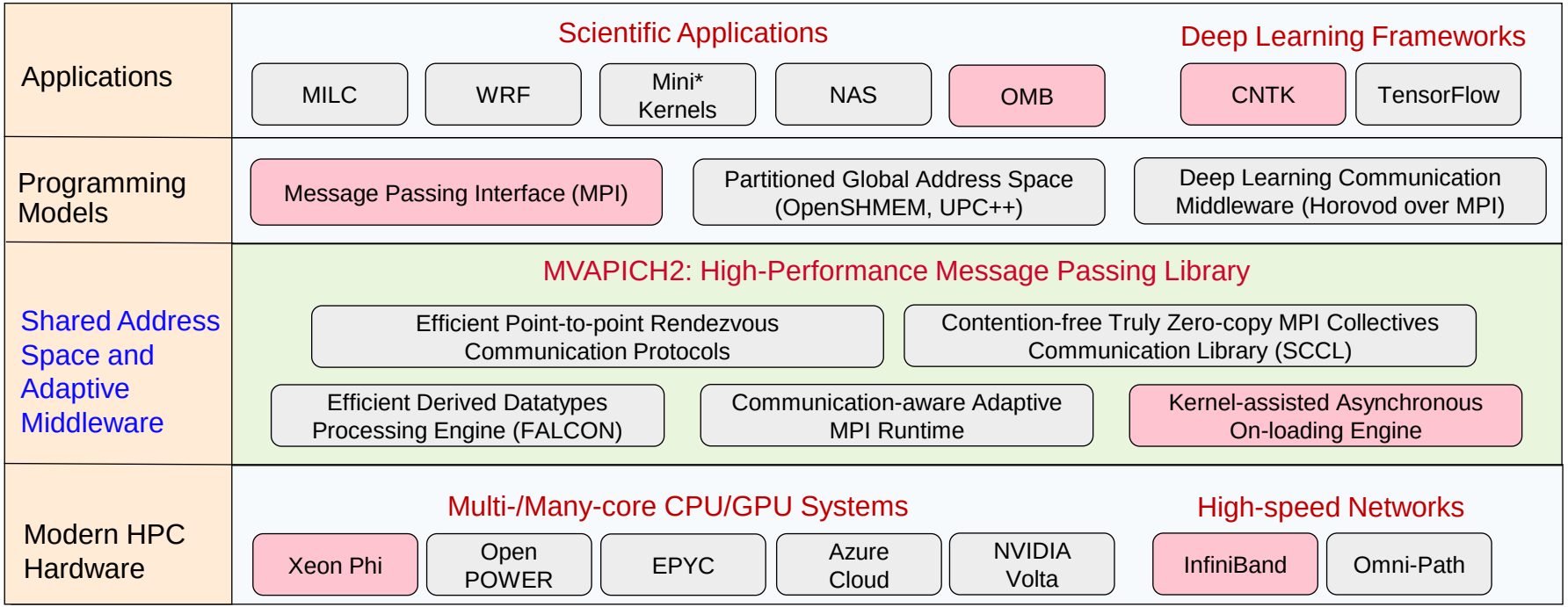
J. Hashmi, S. Xu, B. Ramesh, M. Bayatpour, H. Subramoni, and D. K. Panda

34th IEEE IPDPS '20

Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

Research Framework

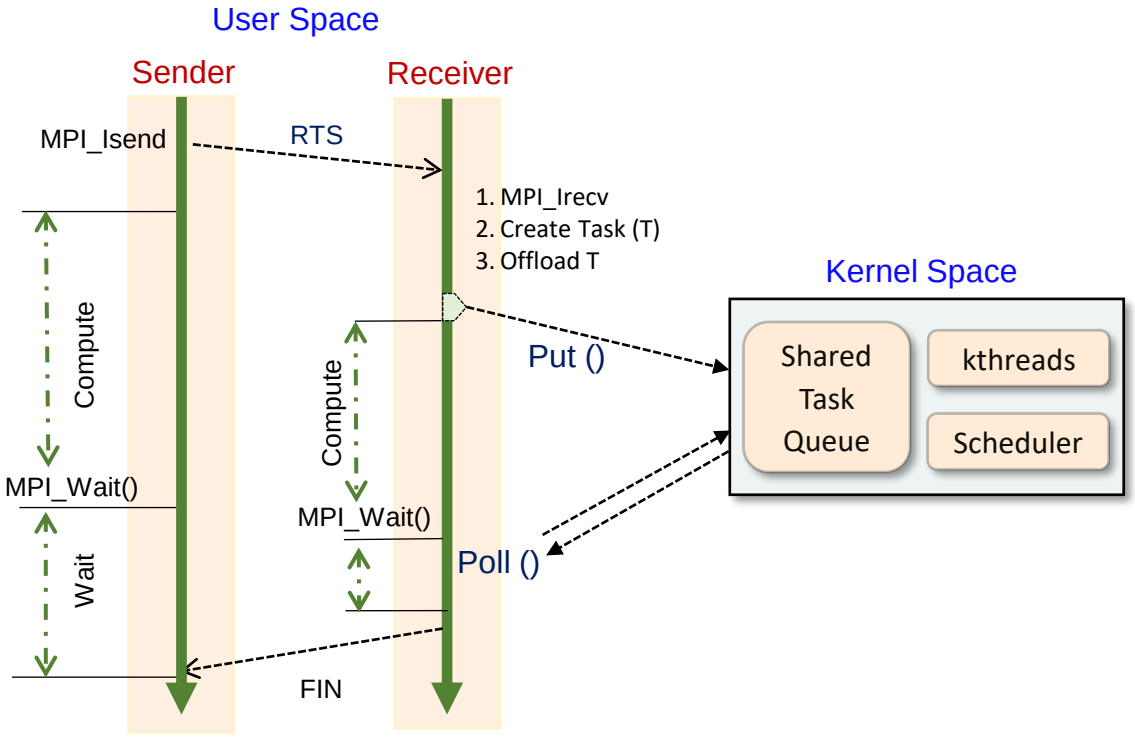


Kernel-assisted MPI Communication on Many-cores

- Intra-node data-movement is blocking memory copies
- Many-cores such as Knights Landing (KNL) have lots of lightweight cores
- Broad Questions
  - Can we dedicate some cores to derive MPI communication?
- Proposed Approach
  - Kernel-assisted Shared Communication On-loading Engine
  - Concurrent and asynchronous progression of communication

Designing Kernel-assisted Communication On-loading Engine

- Two design components
  - API abstraction
  - Kernel module
- High-level API
  - Used by MPI runtime to delegate tasks
  - Integration with MVAPICH2
- Kernel module
  - Task scheduling
  - multi-threading,
  - Signaling



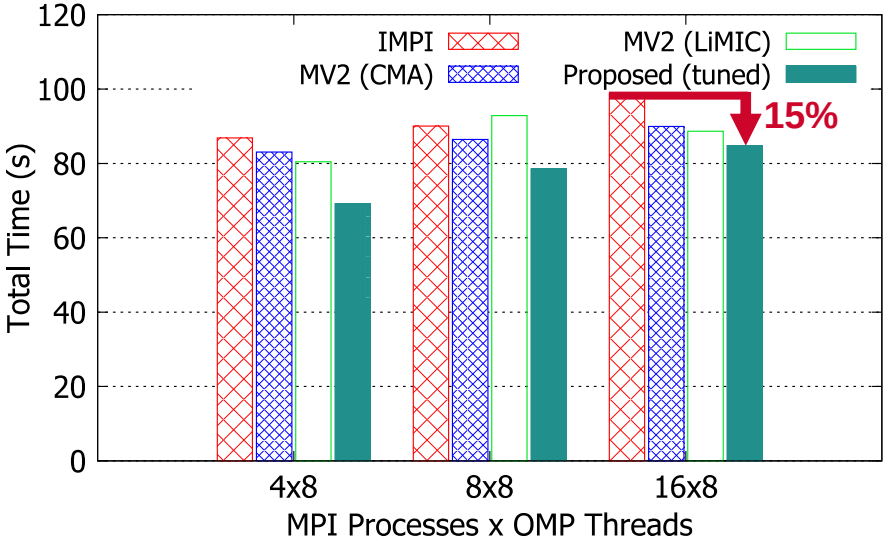
Kernel-assisted Communication Engine for MPI on Emerging Manycore Processors

J. Hashmi, K. Hamidouche, H. Subramoni, and D. K. Panda

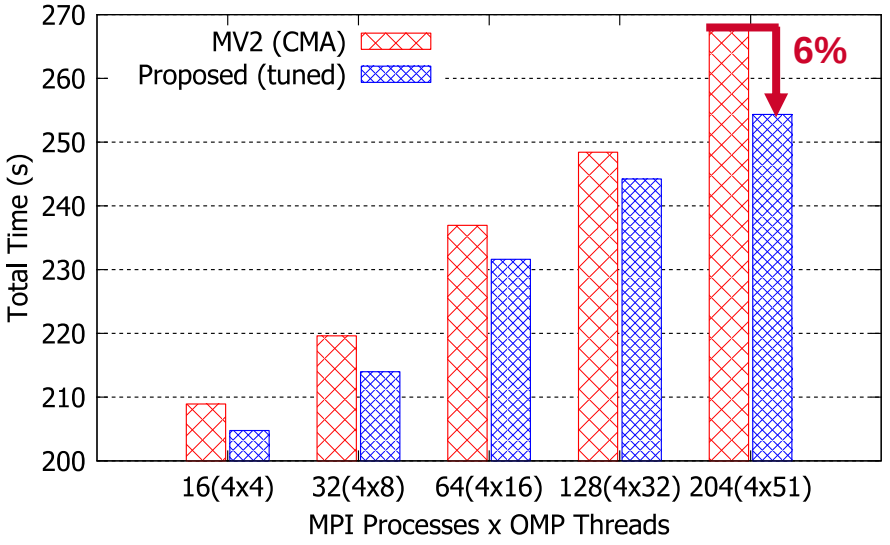
24th IEEE HiPC '17

Application Evaluations on KNL

HPCG



CNTK (MLP+MNIST) Training



- HPCG with MPI+OpenMP running 8 OpenMP threads per MPI process.
  - Main benefits come from DDOT, MG, and DDOT Allreduce phases of HPCG
  - Overall execution time is reduced by **15%** over Intel MPI 2017
- CNTK Multi-level Perceptron (MLP) feed-forward neural network using MNIST dataset

Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

Future Research Directions

- Applicability to Hybrid (MPI+X) Programming Models
  - Hybrid MPI+OpenMP programming models becoming popular
  - Can we exploit shared-address-space based designs for MPI+X models?
- Self-governed Communication Runtimes
  - Communication middlewares (MPI, SHMEM) offer hundreds of tuning parameters
  - Can we design a self-tuning runtime that adapts to communication requirements?

Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

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-3.1), 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 3,075 organizations in 89 countries**
  - **More than 721,000 (> 0.7 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (November '19 ranking)
    - 3rd ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China
    - 5th, 448, 448 cores (Frontera) at TACC
    - 8th, 391,680 cores (ABCI) in Japan
    - 14th, 570,020 cores (Nurion) in South Korea and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
  - **http://mvapich.cse.ohio-state.edu**
- Empowering Top500 systems for over a decade

Partner in the 5th ranked TACC Frontera System

Broader Impact on HPC and AI Community

- Designs made available to the HPC community via MVAPICH2-X releases
- **Adoption by other HPC stacks**
  - Shared address space communication (adopted by MPICH)
  - Efficient Datatype processing (adopted by UCX)
- Proposed work is empowering several of Top500 supercomputers
  - TACC Stampede2, Frontera
  - OSC Owens, Pitzer

Version	Release Date	Included Features
MVAPICH2-X 2.2b	03/30/2016	- Efficient support for UPC++ model via UCR
MVAPICH2-X 2.3rc1	09/21/2018	- XPMEM point-to-point protocols - Truly Zero-copy reductions for InfiniBand channel
MVAPICH2-X 2.3rc2	04/02/2019	- Contention-free collectives using XPMEM - XPMEM collectives for Omni-Path networks
MVAPICH2-X 2.3rc3	03/03/2020	- XPMEM based collectives for PGAS runtimes e.g., OpenSHMEM, UPC, and UPC++
Upcoming*	TBD*	- Efficient datatype processing for CPU/GPU - Adaptive rank-to-core mapping strategies

Overview

- Introduction
- Problem Statement
- Detailed Designs and Results
  - Shared Address Space MPI Point-to-point Communication
  - Shared Address Space MPI Collective Communication Library (SCCL)
  - Efficient Zero-copy Derived Datatype Processing Engine (FALCON)
  - Adaptive MPI Communication Runtime
  - Kernel-assisted Communication On-loading
- Future Research Directions
- Broader Impact on HPC and AI Community
- Conclusion

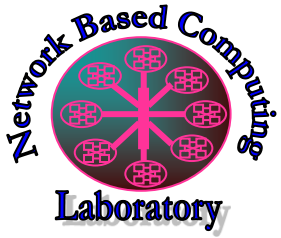
Conclusion

- High core-density architectures are building next-generation ultra-scale systems
- Proposed work optimizes MPI communication for emerging multi-/many-core HPC systems via novel design approaches
- Extends the state-of-the-art in distributed communication middlewares
- Significant impact on the community in transition to next-generation multi-/many-cores
  - Adoption by other communication libraries (MPICH) and runtime (UCX)
- Broader outreach through MVAPICH2 public releases
  - Empowering top supercomputers (TACC Frontera, SDSC Comet, OSC Owens)
- Proposed work defines the direction for future work on applicability to other programming models and heterogeneous systems

Thank You!

hashmi.29@osu.edu

web.cse.ohio-state.edu/~hashmi.29



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/