MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

NETWORK-BASED COMPUTING LABORATORY

# Kernel-assisted Communication Engine for MPI on Emerging Manycore Processors


**Jahanzeb M. Hashmi, Khaled Hamidouche, Hari Subramoni and Dhabaleswar K. (DK) Panda**

**Presenter: Shashank Gugnani**

*Network-Based Computing Laboratory*
*Department of Computer Science and Engineering*
*The Ohio State University, Columbus, OH, USA*

# Agenda

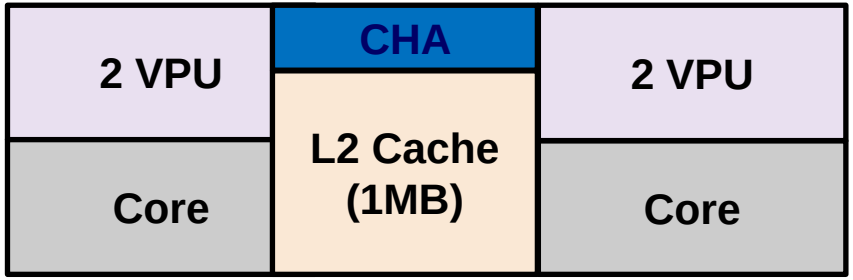
- Introduction
  - KNL Overview
  - Background
  - Motivation
- Problem Statement
- Contributions
- Design and Implementation
- Results and Discussion
- Conclusion and Future Work

# Introduction

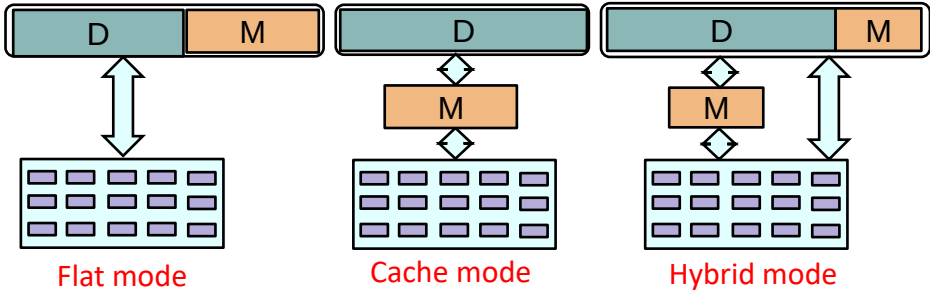
- Many-core processors such as Intel Knights Landing (KNL) are on the rise in HPC industry
- Most of the MPI runtimes have not been using multi-threading in the communication phases
  - Resources such as cores are more valuable to application needs
- Many-cores such as KNL contain lot of slower SMT cores
  - Can some of the cores be dedicated to derive communication?
- Intra-node MPI transfers mainly use memory copies

# Knights Landing Overview

- Multi-threaded cores
  - 272 threads (model 7250)
- Configurable mesh of cores
  - AlltoAll, SNC, Quadrant
- High-bandwidth memory
  - 16GB of MCDRAM
- Different MCDRAM configurations
  - Flat, cache, and hybrid
- 512-bit wide vector registers
  - AVX-512 extensions



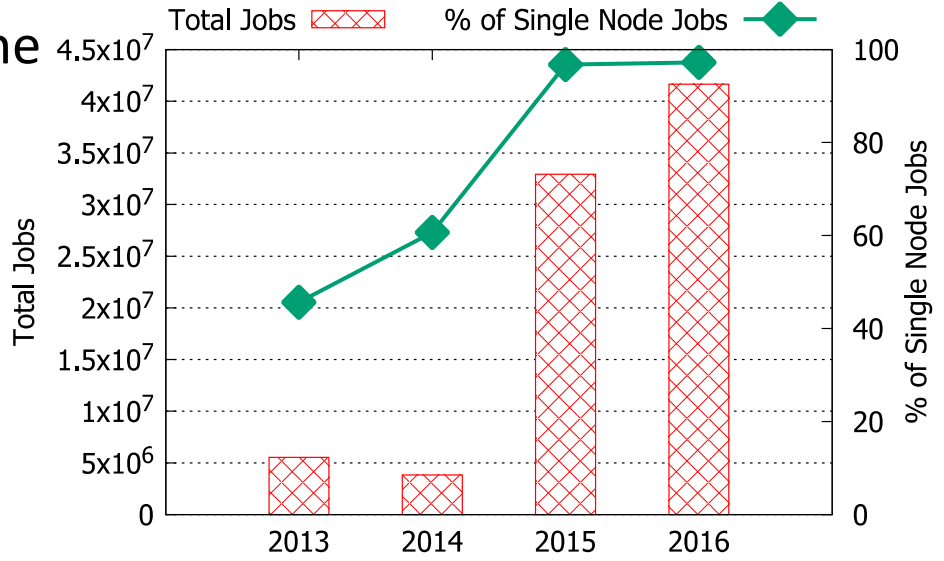

A single Tile of KNL




Different Memory Modes of KNL

# Motivation

- Due to dense multi-/many-cores, the percentage of single-node jobs are increasing
- New mechanisms for bringing concurrency to communication are needed that can achieve
  - Portability
  - Performance
  - Programming Abstraction
- Efficiently utilize many-core resources




Total number of submitted jobs and percent of single-node jobs among all the job over past four years on XSEDE clusters. Small scale jobs are the majority due to dense multi-/many-cores.

# Background

- Intra-node MPI point-to-point communication
  - Shared Memory for small messages
  - Kernel-assisted zero-copy for large messages
    - Popular designs include CMA, LiMIC, and KNEM
- Multi-threaded MPI Runtimes
  - Endpoint Proposal in MPI
    - Application needs to be redesigned (Less portability)
  - Threaded-MPI, *Balaji et al.*
    - Multiple threads to progress communication but application needs to be redesigned

# Outline

- Introduction
- **Problem Statement**
- Contributions
- Design and Implementation
- Results and Discussion
- Conclusion and Future Work

# Problem Statements

***Can we design an efficient mechanism to effectively utilize KNL resources and bring concurrency to the communication phases in MPI?***

***Can we design a communication engine that can asynchronously derive the communication in MPI?***

# Outline

- Introduction
- Problem Statement
- **Contributions**
- Design and Implementation
- Results and Discussion
- Conclusion and Future Work

# Contributions

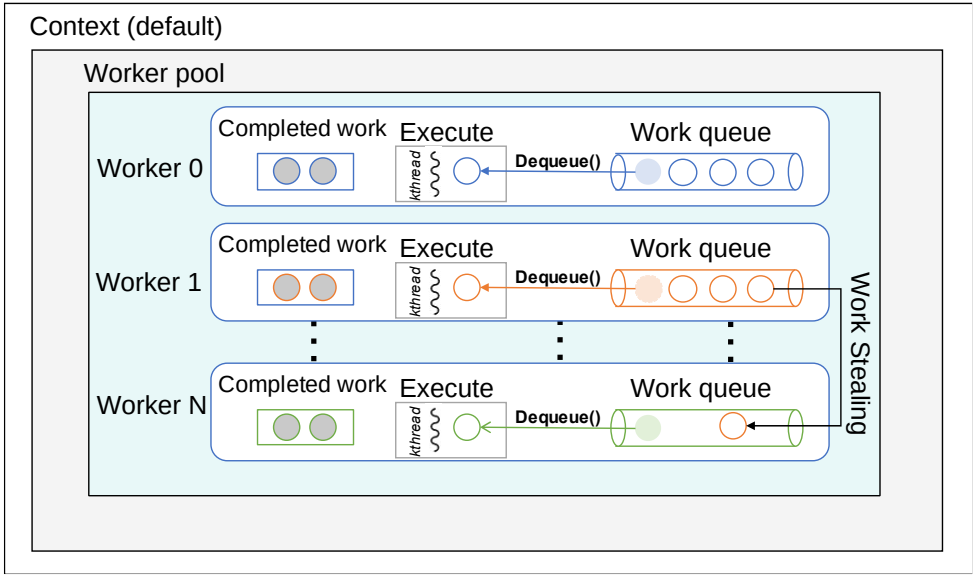
- Propose a generic, multi-threaded, and kernel-assisted ***on-load communication engine*** abstraction and its implementation as a kernel module
- Implement MPI zero-copy transfers using proposed engine based approach and compare it with the prevalent designs in modern MPI libraries
- Demonstrate the impact and the benefits of our designs on popular micro-benchmarks and applications including HPCG, and CNTK Deep Learning framework

# Outline

- Introduction
- Problem Statement
- Contributions
- **Design and Implementation**
- Results and Discussion
- Conclusion and Future Work

# On-load Communication Engine: Abstraction

- Proposed on-load engine is envisioned as an entity that progresses the communication tasks on behalf of the MPI processes
- A high level abstraction of different components of the engine include:
  - ***Context***: A high-level encapsulation of engine resources
  - ***Worker-Pool***: A pool of worker threads to carry-out the work
  - ***Worker:*** A driver of the task execution
  - ***Work-Item:*** A generic task containing the definition of work
  - ***Work-Queue:*** A doubly-ended (Deque) to hold submitted tasks



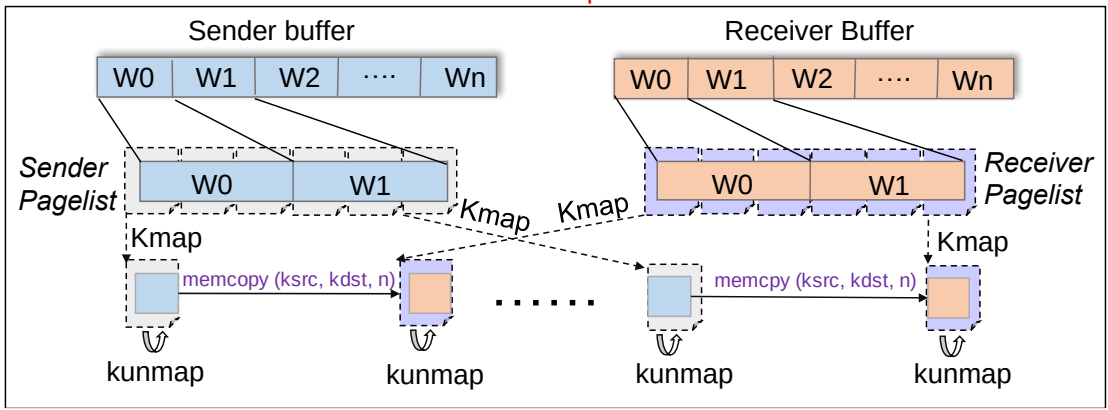

Proposed On-loading Engine Abstraction Overview

# On-load Communication Engine: Kernel Module

- Kernel threads (kthreads) are used to realize “workers” abstraction
- Efficient kernel data-structures such as *list_head, wait_queues, task_struct, spin_locks* etc. are used to implement different abstractions
- Kthreads are registered with Linux *Waitqueues*
  - MPI process wakes-up kthreads when a new request is posted
  - When task is progressed, kthread goes back to sleep
- Integration with MPI runtime
  - MPI_Send invokes engine routine to get sender process info
  - MPI_Recv, after matching, will pass sender’s info to engine via *ioctl*
  - Locality-aware mapping of kthreads

# Realizing MPI Zero-copy Operations using On-load

- Implemented MPI large-message transfer using on-load engine
  - Multiple threads carry-out the page-level copies




- MPI_Recv invokes engine routines and posts the request
  - Request contains sender's buffer address, # of workers required, and a function pointer *F() to the page-mapped memory copy implementation
- The requested workers execute *F() on a partition of user buffer
  - *Kmap()* sender and receiver pages, do memcpy(), kunmap()

# 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 (Nov '17 ranking)
    - **1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China**
    - 12th, 368,928 cores (Stampede2) at TACC
    - 17th, 241,108-core (Pleiades) at NASA
    - 48th, 76,032-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 |

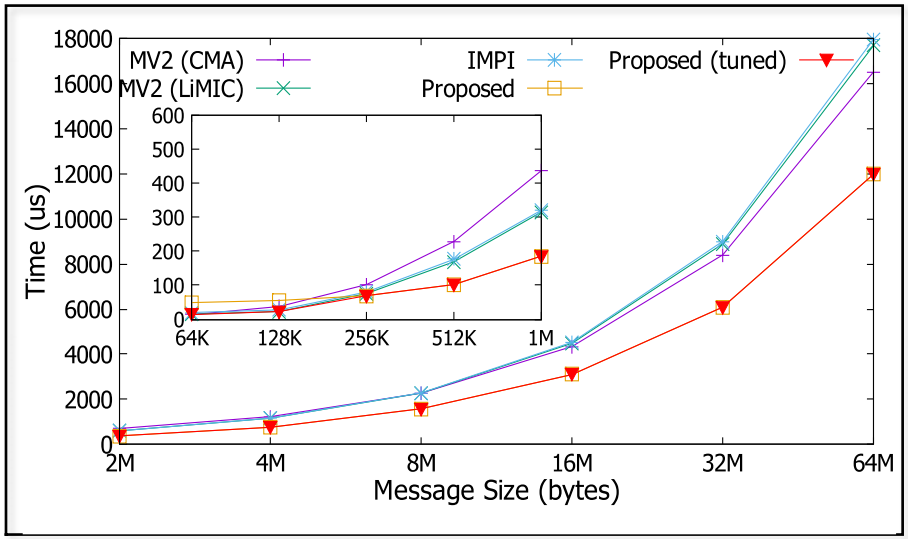
# Outline

- Introduction
- Problem Statement
- Contributions
- Design and Implementation
- **Results and Discussion**
- Conclusion and Future Work

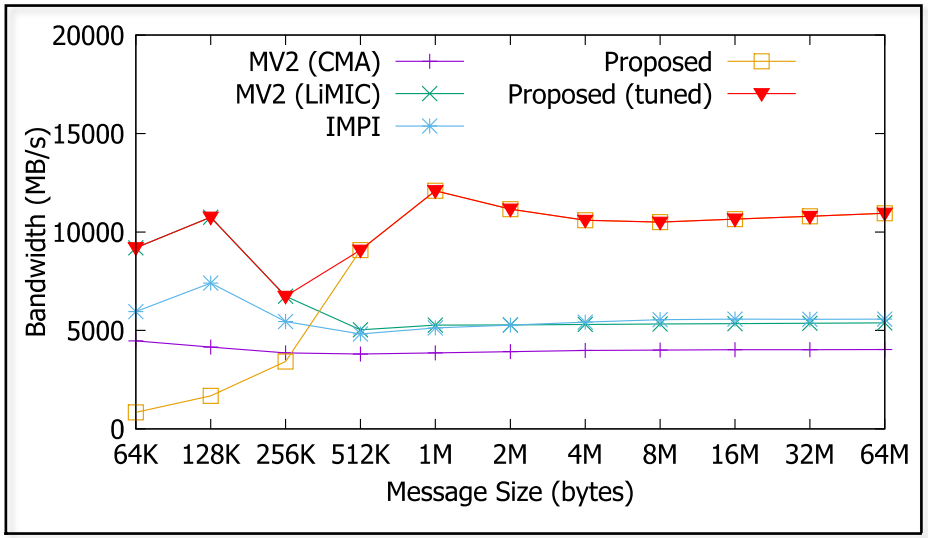
# Evaluation Benchmarks and Testbed

- Benchmarks and applications
  - OSU Microbenchmark v5.3.2
  - High Performance Conjugate Gradient (HPCG) Kernel
  - Microsoft CNTK Deep Learning Framework
- Intel Xeon Phi KNL Cluster @ CSE, OSU
  - KNL 7250 (1.4GHz) with 16GB MCDRAM and 96GB DDR
- Comparative Designs
  - MVAPICH2-X 2.3a with CMA and LiMIC support
  - IntelMPI 2017 with CMA support enabled

# Microbenchmark Evaluations (Single-pair Pt2Pt Test)



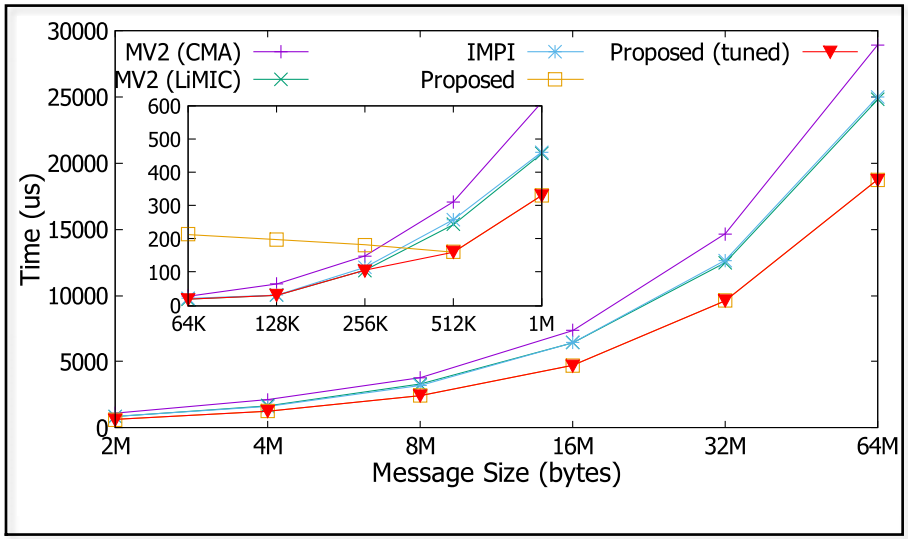

2-process Latency on MCDRAM



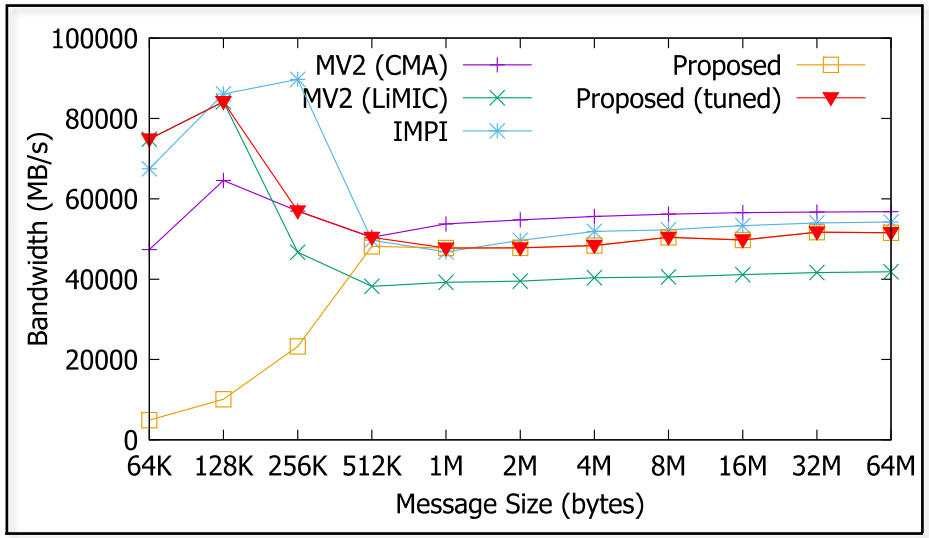

2-process Bi-Bandwidth on MCDRAM

- Single-pair latency and bi-directional bandwidth test using 2 processes
- For large message latency, up to 34% improvement over existing designs
- For bandwidth, we observe up to 75% improvement from 1M to 64M message ranges when using proposed designs

# Microbenchmark Evaluations (Multi-Pair Pt2Pt Test)



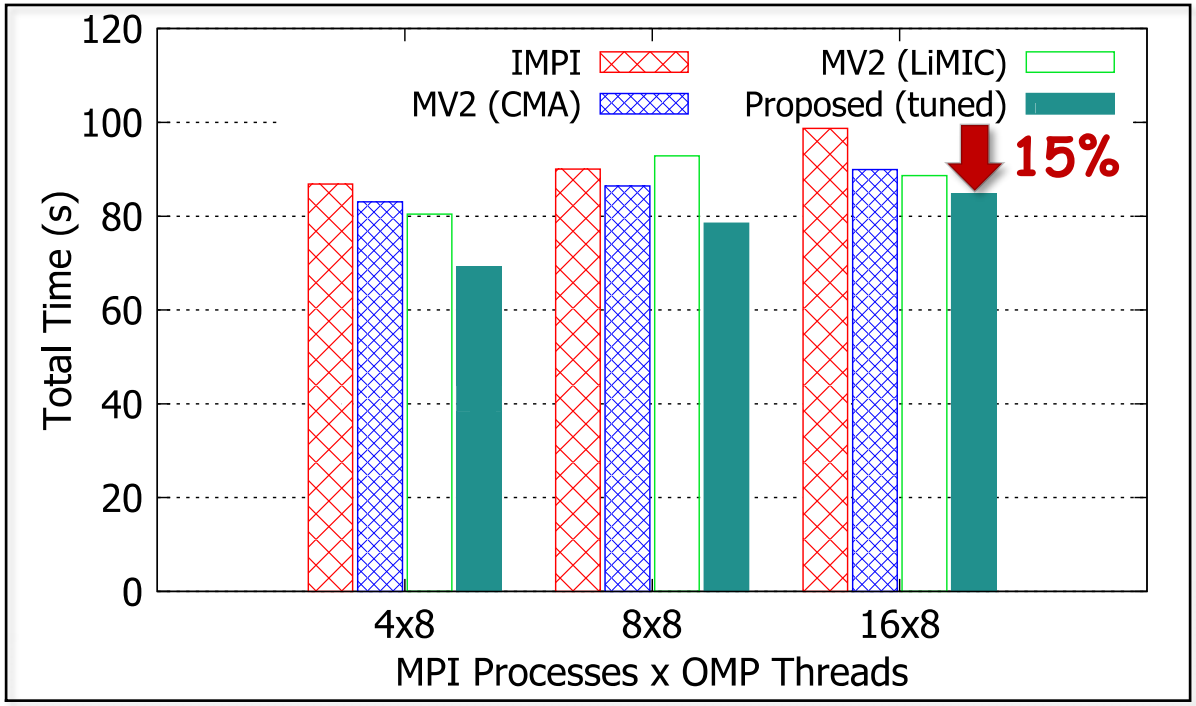

Multi-pair Latency on MCDRAM




Multi-pair Bandwidth on MCDRAM

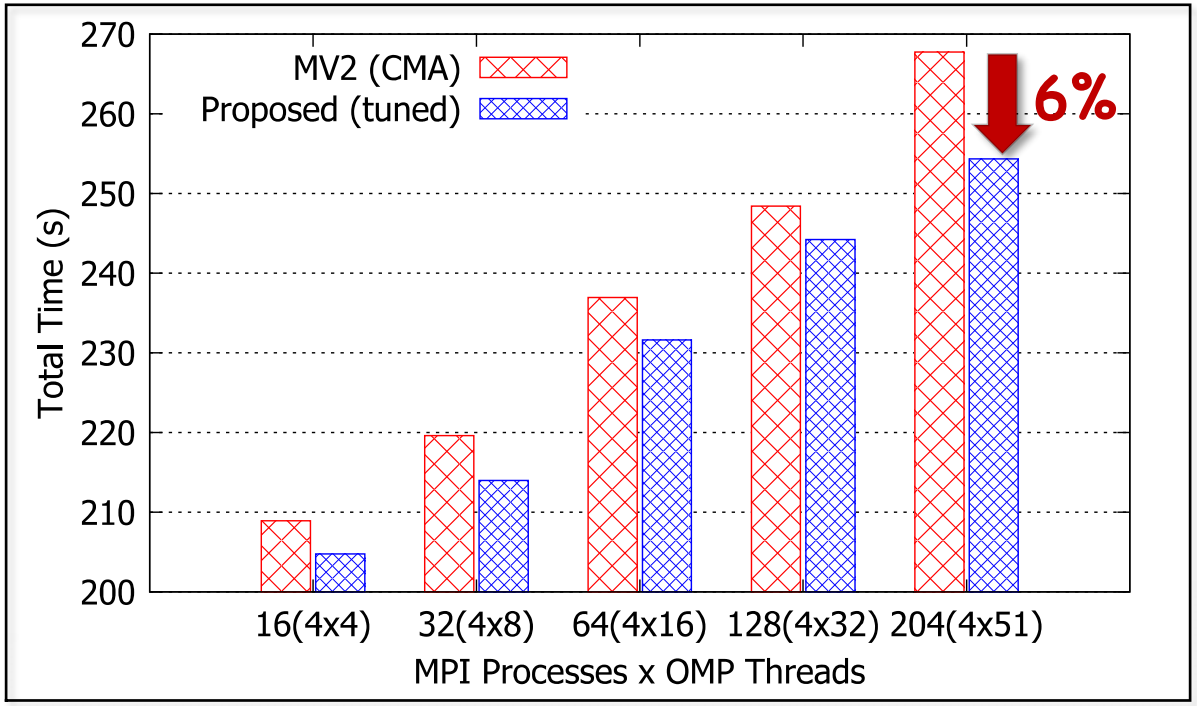
- Multi-pair latency and bandwidth test using 32 processes and 16 pairs
- Compared with MVAPICH2-CMA, MVAPICH2-LiMIC, and Intel-MPI 2017
- For large messages, up to 34% improvement over Intel-MPI, and 38% improvement over MVAPICH2-CMA is observed

# Application Evaluations (HPCG)



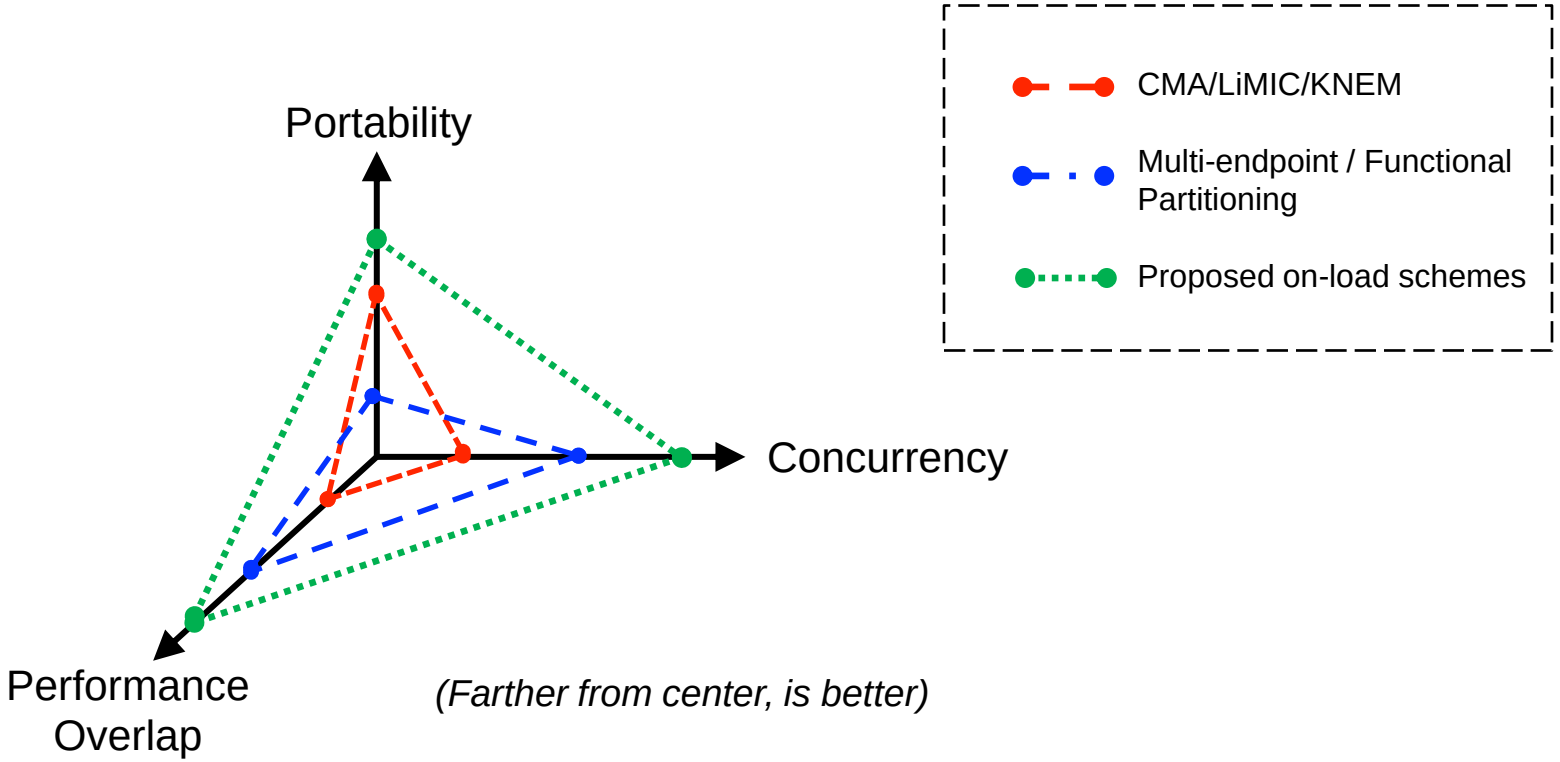

- Hybrid MPI+OpenMP evaluation with 8 OpenMP threads per MPI process.
- Main benefits achieved from **DDOT, MG, and DDOT *All Reduce*** phases of HPCG
- Overll execution time is reduced by **15%** when using proposed designs

# Application Evaluations (CNTK)




- CNTK Multi-level Perceptron (MLP) feed-forward neural network using MNIST dataset
- Different variation of MPI+OpenMP threads are used
- Noticeable improvement in total training time of MLP is observed with proposed design

# Performance Results Summary



# Outline

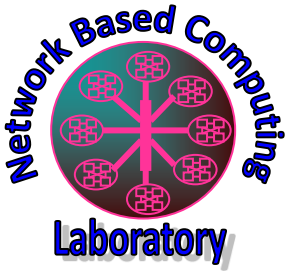
- Introduction
- Problem Statement
- Contributions
- Design and Implementation
- Results and Discussion
- **Conclusion and Future Work**

# Conclusion and Future Work

- Proposed and designed an on-load engine abstraction that achieves concurrency, portability, and programmability
- Realized this abstraction as a multi-threaded kernel-assisted engine
- Compared our proposed designs with state-of-the-art intra-node communication designs employed by modern MPI libraries
  - 2.5X better latency performance than MVAPICH2-CMA and MVAPICH2-LiMIC
  - 2X improvement over Intel MPI
- Observe up to 15% improvement in execution time of hybrid HPCG application
- Significant improvement over existing zero-copy based approaches for MLP training using CNTK deep learning framework
- We plan to extend this abstraction for inter-node communication
- We also plan to show its applicability to other programming models e.g., PGAS
- Proposed Designs will be available in future releases of MVAPICH2

# Thank You!

{hashmi.29, hamidouche.2, subramoni.1, panda.2}@osu.edu




MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Network-Based Computing Laboratory

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

MVAPICH Web Page

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