Accelerating k-NN Algorithm with Hybrid MPI and OpenSHMEM

Jian Lin[1], Khaled Hamidouche[1], Jie Zhang[1], Xiaoyi Lu[1], Abhinav Vishnu[2], Dhabaleswar K. Panda[1]

1. Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University, USA

2. Pacific Northwest National Laboratory

Outline

- Introduction
- Problem Statement
- Proposed Designs
- Evaluation
- Conclusion

Outline

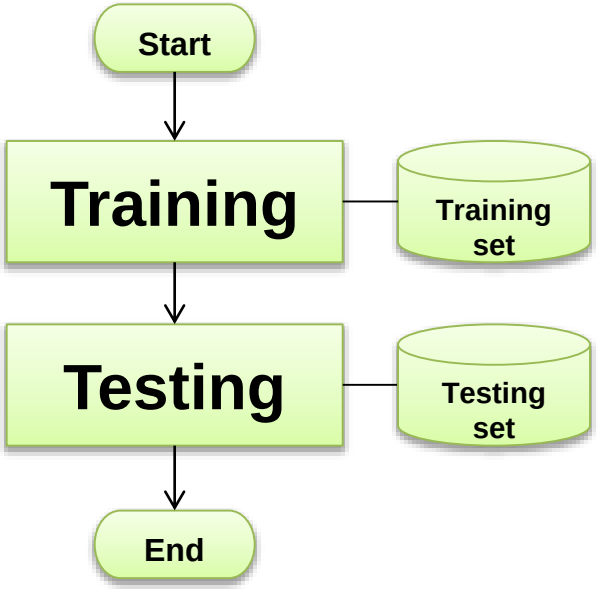
- Introduction
- Problem Statement
- Proposed Designs
- Evaluation
- Conclusion

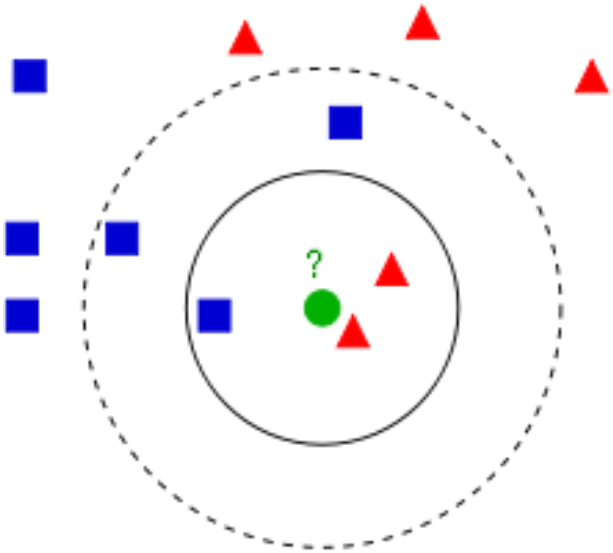
Machine Learning

- Machine learning algorithms are widely used in the Cloud and Big Data era
  - Problems/algorithms: classification, regression, association rules, structured prediction, …
  - Applications: artificial intelligence, data analysis, pattern recognition, …
  - Platforms/libraries: Apache Mahout, PNNL MaTEx, LIBSVM, Pylearn2, …
  - Supporting technologies: MPI, MapReduce, PGAS, …

k-NN Algorithm

- k-Nearest Neighbors (k-NN) algorithm is a popular supervised machine learning algorithm for classification





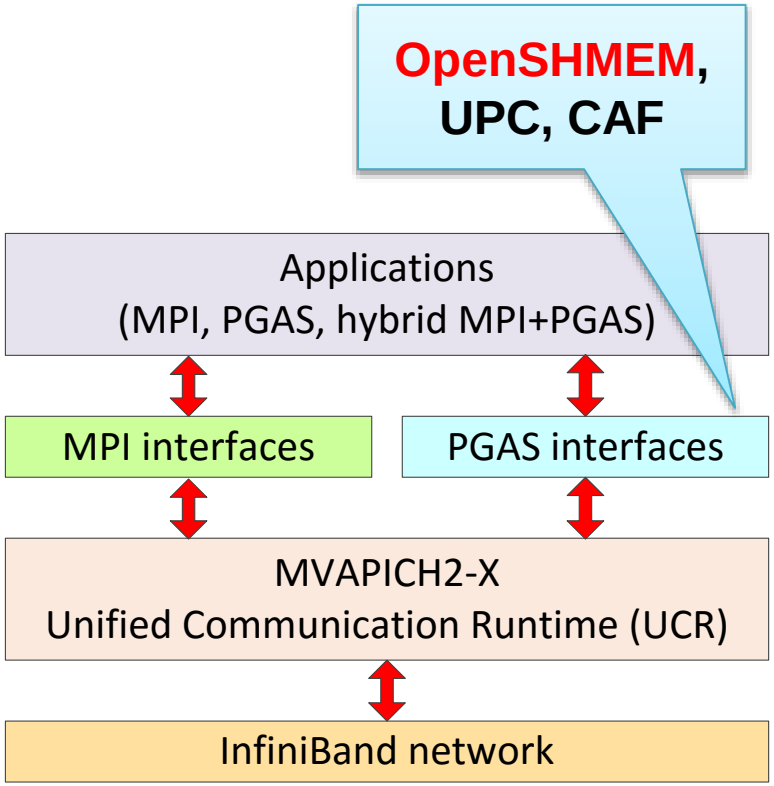
k-NN Acceleration and Optimization

- MPI-based
  - Aparicio2006, Moon2011, etc.
- MapReduce-based
  - Zhang2012, etc.
- GPU-based
  - Arefin2012, etc.
- Can PGAS model and OpenSHMEM benefit the k-NN algorithm?

MVAPICH2/MVAPICH2-X Software

- High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Support for GPGPUs and MIC
  - **Used by more than 2,425 organizations in 75 countries**
  - **More than 278,000 downloads from OSU site directly**
  - Empowering many TOP500 clusters (June 2015 ranking)
    - 8th ranked 519,640-core cluster (Stampede) at TACC
    - 11th ranked 185,344-core cluster (Pleiades) at NASA
    - 22th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - Available with software stacks of many IB, HSE, and server vendors including Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu
- Partner in the U.S. NSF-TACC Stampede System

MVAPICH2-X: Unified Communication Library for MPI and PGAS



The Basic Architecture of MVAPICH2-X

- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, CAF, MPI(+OpenMP) + OpenSHMEM/UPC /CAF
  - MPI-3 standard compliant, OpenSHMEM v1.0 compliant, UPC v1.2 (initial support for UPC v1.3) compliant, CAF in Fortran 2008 (partial support for CAF in Fortran 2015) compliant
  - Scalable inter-node communication with high performance and reduced memory footprint
  - Optimized intra-node communication using shared memory schemes
  - Optimized OpenSHMEM collectives
  - Supports different CPU binding policies
  - Flexible process manager support

Outline

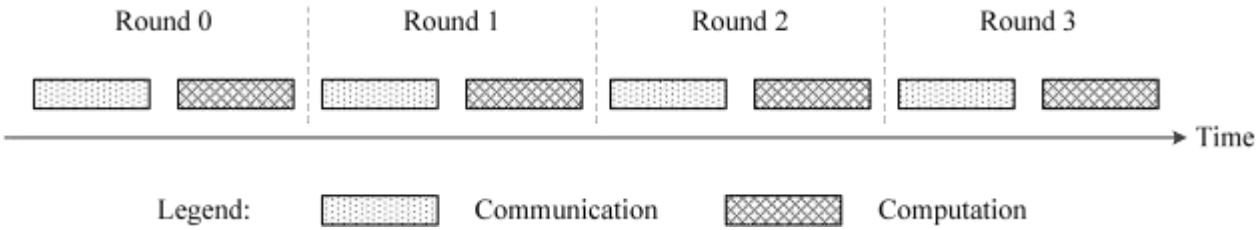
- Introduction
- Problem Statement
- Proposed Designs
- Evaluation
- Conclusion

MPI-based k-NN Design in MaTEx

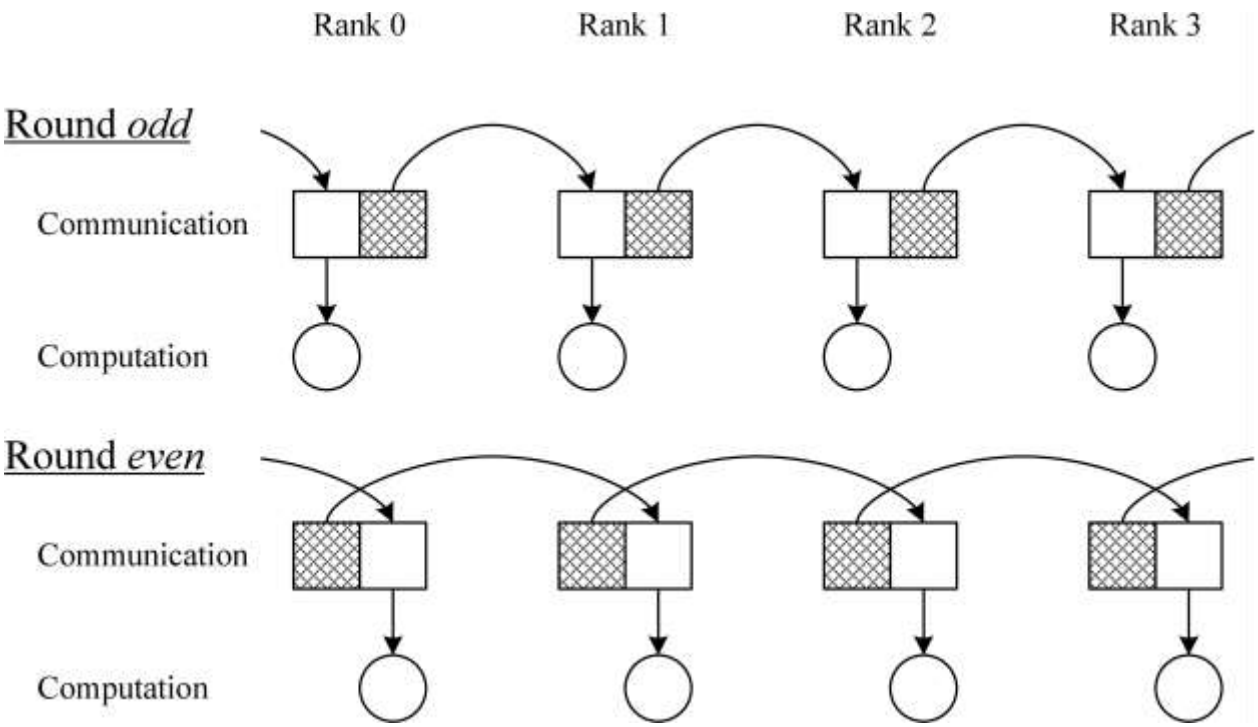
- Written in C++, using MPI for communication
- Support dataset in LIBSVM format
- Three main phases
  - **Data loading**: simple I/O operations and few broadcast communication
  - **Training**: intensive communication and computation – consumes most time and resource
  - **Testing**: local computation and few reduction communication

Main Logics in the Training Phase

The non-overlapped timing relationship between communication and computation



The data process flow based on original twin-buffer structures



Bottleneck Investigation

- Communication and computation are not overlapped
- The twin-buffer structure limits the amount of data pre-fetching
- There are some variations for both communication and computation time across different rounds

Research Objectives

- Can PGAS model and OpenSHMEM benefit the k-NN algorithm?
  - Where and how to leverage the one-sided communication semantics to improve the performance of k-NN?
  - What are the benefits of performance and scalability by introducing hybrid MPI+ OpenSHMEM designs into k-NN?

Outline

- Introduction
- Problem Statement
- Proposed Designs
- Evaluation
- Conclusion

Overview of Alternative Designs

Alternative designs

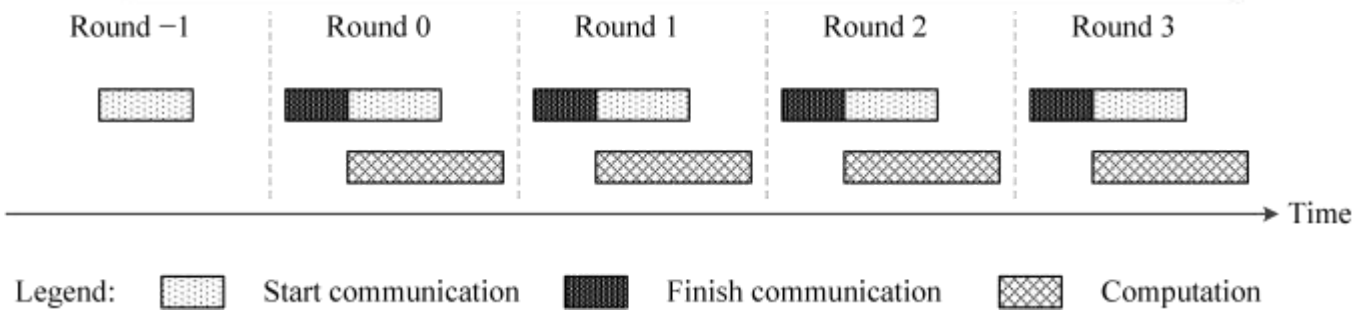
Techniques

	MPI	MPI_O	OSH_OW	OSH_OM	OSH_OC
Overlapped Data Process Flow		●	●	●	●
OpenSHMEM-based One-sided Data Transfer			●	●	●
Memcpy-based Optimization				●	
Circular-buffer Structure Design					●

Overlapped Data Process Flow

Algorithm 1: Overlapping communication for k-NN

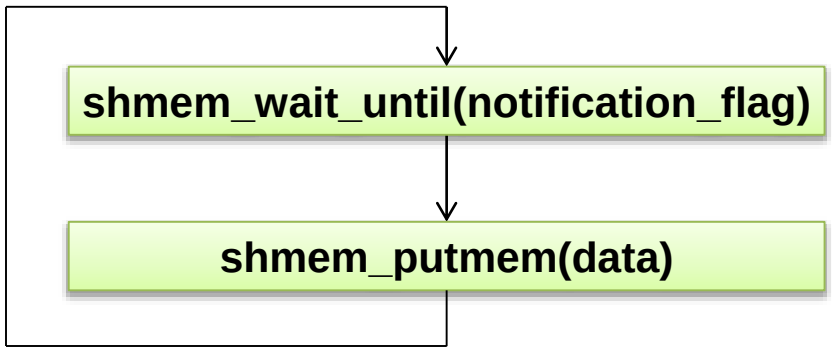
```
for round ← −1 to max_round do
    if round ≠ −1 then
        wait_for_data(local_row_ptr)
        wait_for_data(local_samples)
        row_ptr_len ← get_length(local_row_ptr)
        samples_len ← get_length(local_samples)
    end
    if round ≠ max_round then
        async_transfer_data(local_row_ptr, remote_row_ptr, row_ptr_len)
        async_transfer_data(local_samples, remote_samples, samples_len)
    end
    if round ≠ −1 then
        do_computation(local_row_ptr, row_ptr_len, local_samples, samples_len)
    end
end
```



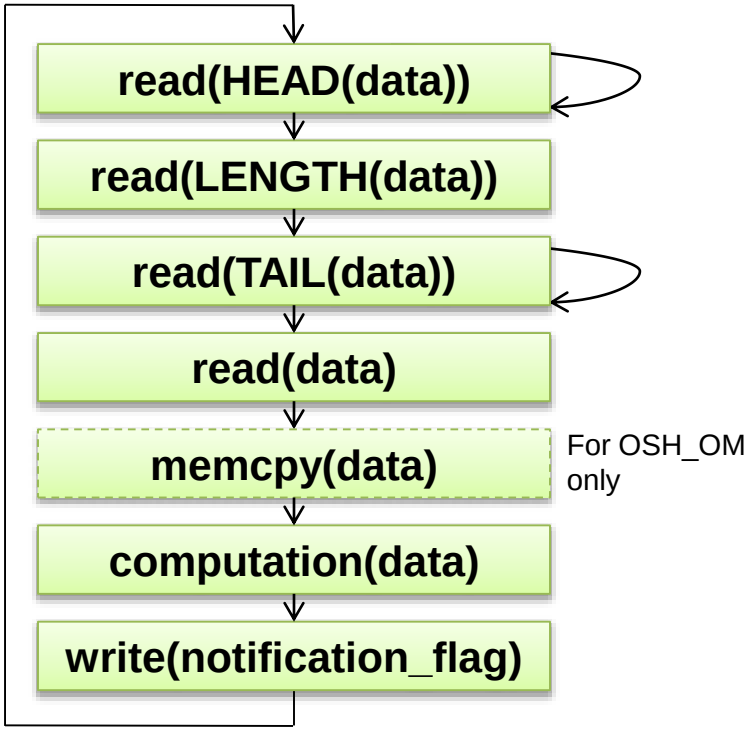
Available in: MPI_O | OSH_OW | OSH_OM | OSH_OC

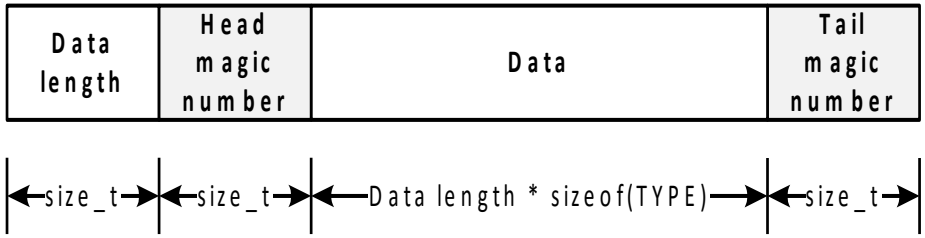
OpenSHMEM-based One-sided Data Transfer

Source process



Destination process

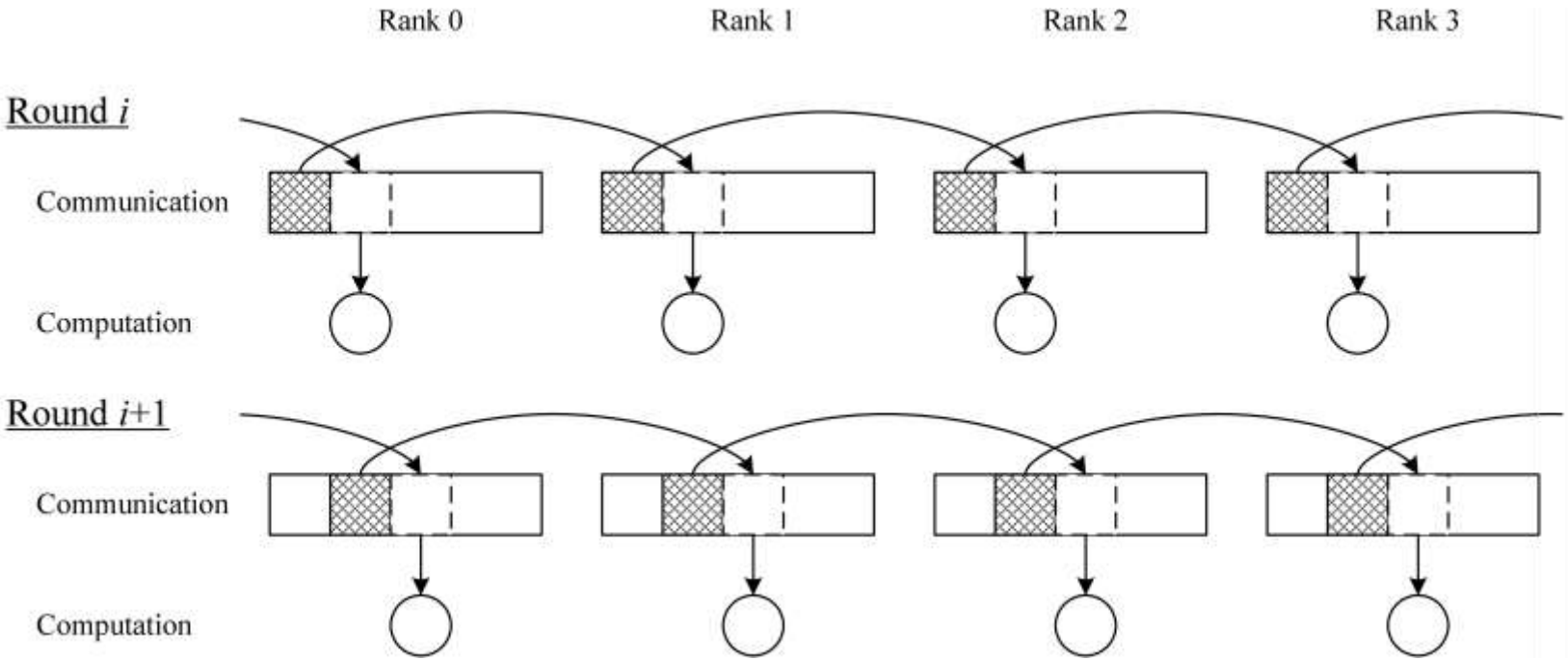




Available in: OSH_OW OSH_OM OSH_OC

Circular-buffer Structure Design

The data process flow based on proposed circular-buffer structures



Available in: OSH_OC

Outline

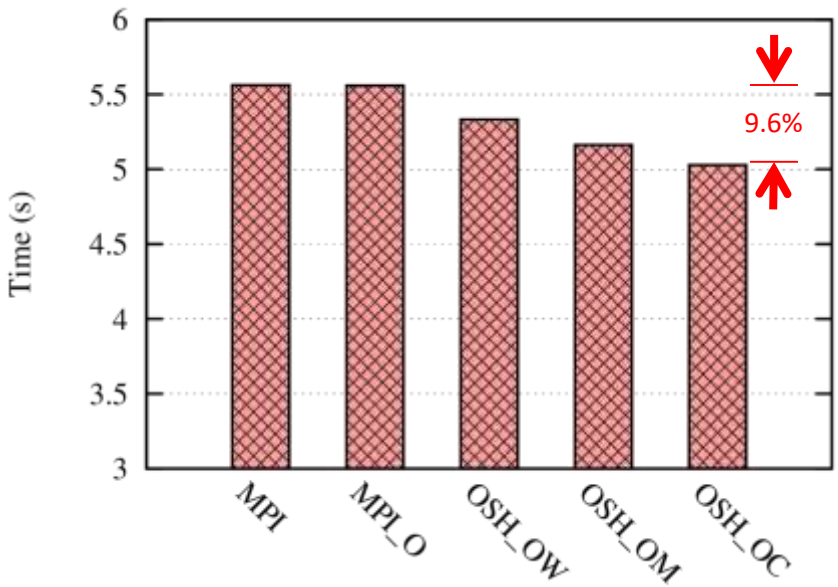
- Introduction
- Problem Statement
- Proposed Designs
- Evaluation
- Conclusion

Experiment Setup

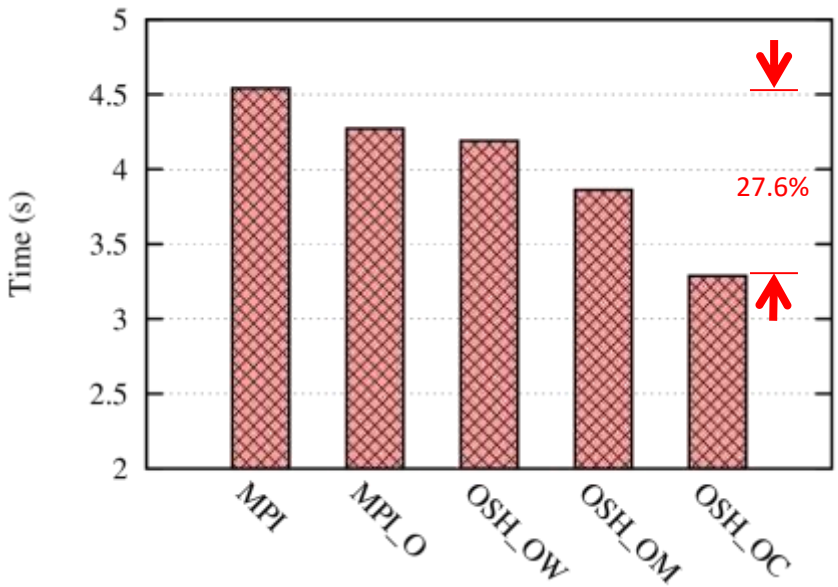
- Hardware: RI Cluster @ CSE, OSU
  - Intel Xeon dual 8 core sockets (2.53GHz) with 12GB RAM
  - Mellanox QDR ConnectX HCAs (32 Gbps data rate) with PCI-Ex Gen2 interfaces
- Software stack
  - RHEL 6.5
  - MLNX_OFED 2.2
  - MVAPICH2-X 2.1
- Benchmark
  - KDD Cup 2010 (8,407,752 records, 2 classes, k=5)

Performance Tests on 128-256 cores

KDD-XS workload on 128 cores



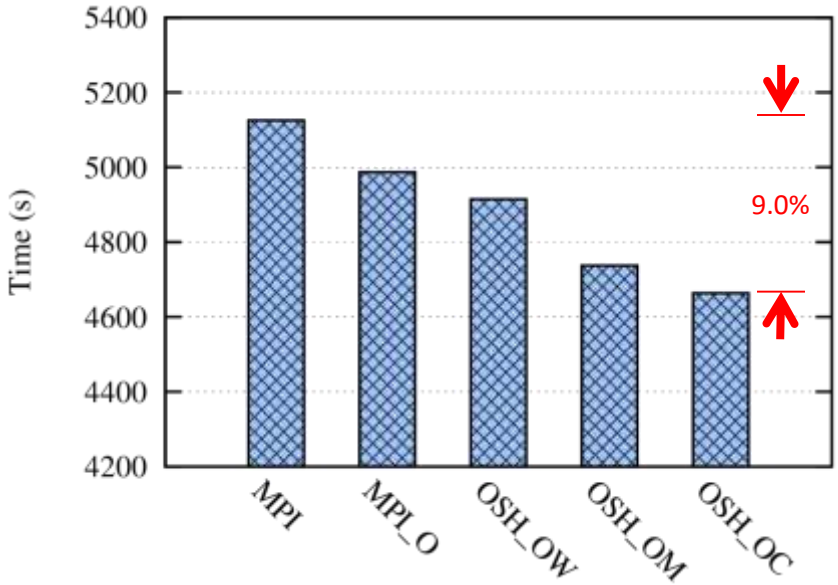
KDD-XS workload on 256 cores



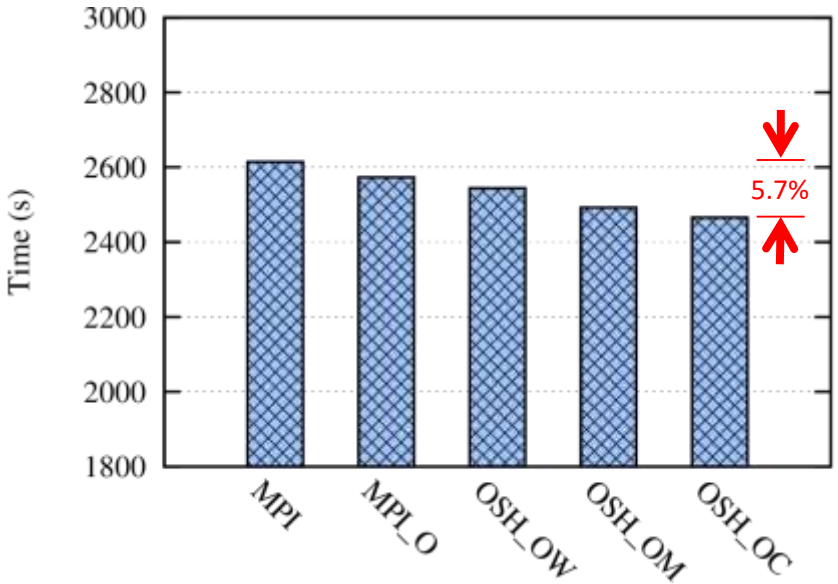
- Truncated KDD with 100,000 records (30MB)
- Example: 256-core, comparing with the original MPI design: MPI_O can save 5.9% time, and OSH_OC can save 27.6% time

Performance Tests on 512-1024 cores

KDD workload on 512 cores



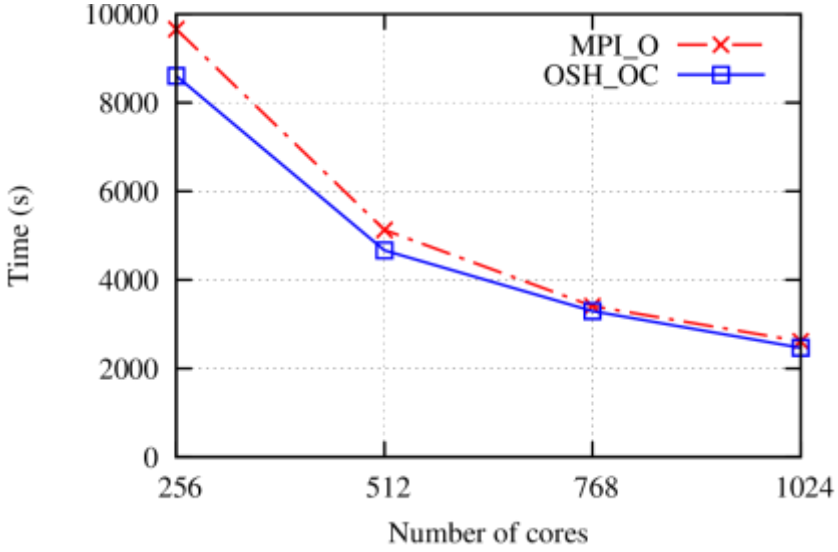
KDD workload on 1024 cores



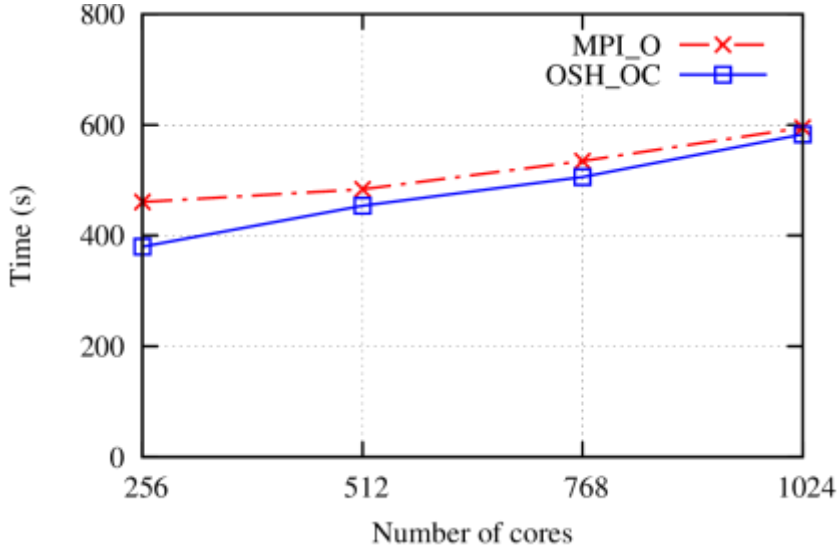
- KDD with 8,407,752 records (2.5GB)
- Example: 512-core, comparing with the original MPI design: MPI_O, OSH_OW, OSH_OM and OSH_OC can save 2.7%, 4.1%, 7.6%, and 9.0% time, respectively

Scalability Tests

Strong scalability test



Weak scalability test



- Truncated KDD (1,000,000 records per 256 cores)
- The MPI+OpenSHMEM design has good strong scalability, and does not break the scalability

Outline

- Introduction
- Problem Statement
- Proposed Designs
- Evaluation
- Conclusion

Conclusion

- OpenSHMEM can benefit k-NN algorithm!
  - Save up to 9.0% time for training typical KDD workload over 512 cores
  - Save up to 27.6% time for workload with balanced communication and computation
  - Keep good strong scalability as the original design does
  - Using hybrid MPI+OpenSHMEM designs with MVAPICH2-X to overlap communication and computation, improve memory management, and hide performance variation

Future Work

- Further decouple communication and computation by adjusting data placement
- Accelerate data loading by efficient data distribution algorithm
- Propose a portable version for different OpenSHMEM runtime
- Accelerate other popular algorithms from machine learning and deep learning with the hybrid MPI+PGAS technology

Thank you!

{linjia,hamidouc,zhanjie,luxi,panda}@cse.ohio-state.edu,
abhinav.vishnu@pnnl.gov

Network-Based Computing Laboratory

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

MVAPICH Web Page

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