HiBD
High-Performance Big Data

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

HiDL
High-Performance Deep Learning

Benchmarking Parallel Python and Java Applications using OMB and MVAPICH2

10th MVAPICH User Group (MUG) Meeting '22

By

Aamir Shafi
The Ohio State University
shafi.16@osu.edu
https://cse.osu.edu/people/shafi.16

Nawras Alnaasan
The Ohio State University
alnaasan.1@osu.edu
nowlab.cse.ohio-state.edu/member/alnaasan.1/

Agenda

- **OSU MPI Micro-Benchmarks (OMB) Suite**
- Java Extensions to OMB
  - MVAPICH2-J: Java Bindings to MVAPICH2
  - Performance Evaluation using Java OMB benchmarks
- Python Extensions to OMB
  - Benchmarks Implementation
  - Performance Characterization using Python OMB benchmark
- Summary

OSU MPI Micro-Benchmarks (OMB) Suite

- OMB is a benchmarking tool that aids in measuring the performance of communication libraries on HPC systems with different configurations and hardware
- There is support for a variety of programming models and communication libraries including MPI, OpenSHMEM, UPC, and UPC++
- It provides a variety of benchmarks including point-to-point, blocking/non-blocking collectives, and one-sided communication primitives
- There is support for evaluation performance of data communication to/from NVIDIA and AMD GPUs
- The aim of this talk is to provide an overview of recent Java and Python extensions to OMB

Java and Python Extensions to OMB

- Java and Python extensions have been released as part of the OMB 6.0 release:
  - https://mvapich.cse.ohio-state.edu/benchmarks/
- Instructions for using OMB for Java:
  - User guide: https://mvapich.cse.ohio-state.edu/static/media/mvapich/README-OMB-J.txt
  - Sample run:

```
mpirun_rsh -np 2 -hostfile hosts \
  LD_PRELOAD=${MPILIB}/lib/libmpi.so java -cp $MV2J_HOME/lib/mvapich2-j.jar:. \
  -Djava.library.path=$MV2J_HOME/lib mpi.pt2pt.OSUBandwidth
```

- Instructions for using OMB for Python:
  - User guide: https://mvapich.cse.ohio-state.edu/static/media/mvapich/README-OMB-PY.txt
  -

```
mpirun -np 2 --hostfile hosts python run.py \
--benchmark latency --buffer numpy
```

Agenda

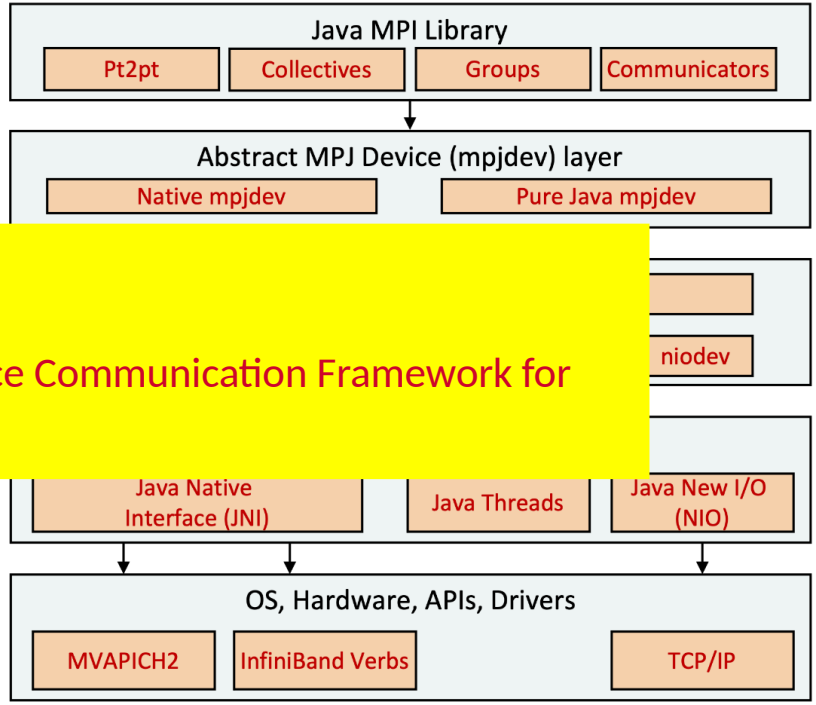
- OSU MPI Micro-Benchmarks (OMB) Suite
- **Java Extensions to OMB**
  - **MVAPICH2-J: Java Bindings to MVAPICH2**
  - **Performance Evaluation using Java OMB benchmarks**
- Python Extensions to OMB
  - Benchmarks Implementation
  - Performance Characterization using Python OMB benchmark
- Summary

MVAPICH2-J: Java Bindings to MVAPICH2

- We have recently added Java bindings to the MVAPICH2 library:
  - Allows writing HPC applications in the Java programming language
- The library currently implements a subset of the MPI API:
  - Our bindings follow the same API as Open MPI Java bindings
- MVAPICH
  - blocki
  - blocki
  - blocki
- Motivation:
  - Enhance communication infrastructure of BigData frameworks, written in Scala/Java, using MPI
- MVAPICH2-J 2.3.7 is recently released:
  - Userguide: https://mvapich.cse.ohio-state.edu/userguide/mv2j/

Talk Announcement:

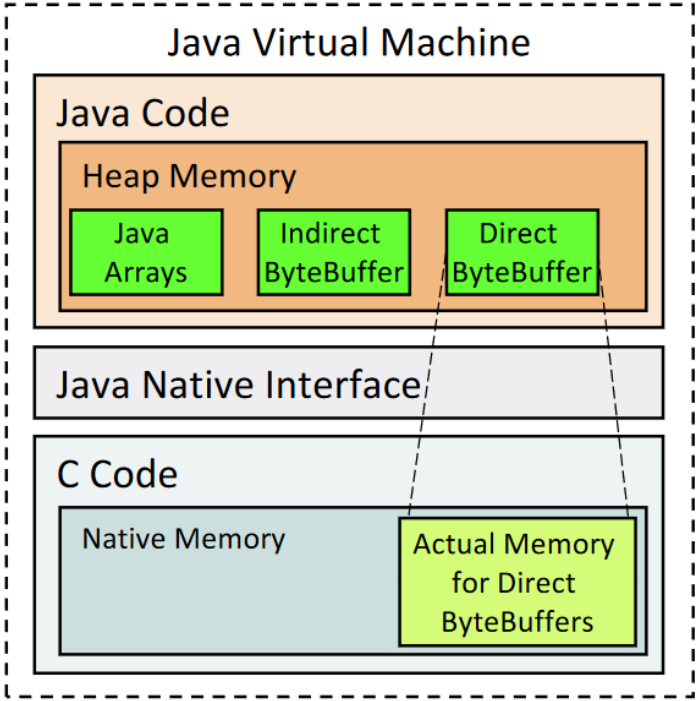
Wed, 3:00 – 3:30, MPI4Spark: A High-Performance Communication Framework for Spark using MPI



K. Al Attar, A. Shafi, H. Subramoni, D. Panda, Towards Java-based HPC using the MVAPICH2 Library: Early Experiences, HIPS '22 (IPDPSW), May 2022.

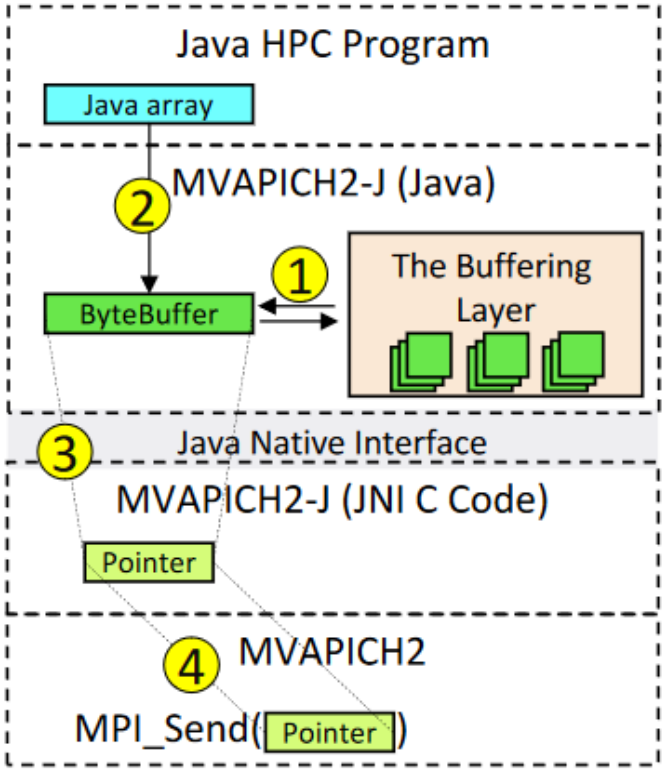
Java Arrays vs. Indirect/Direct ByteBuffers

- Communication Java arrays incur significant overhead in Java MPI libraries:
  - Direct ByteBuffers provide an attractive alternative
- Java arrays and indirect ByteBuffers or Buffers in general are allocated differently when compared to direct ByteBuffers
- Direct ByteBuffers, though are costly to create and destroy, only exist as pointers inside of the heap memory
  - Direct ByteBuffers are allocated inside of native memory and are not subject to garbage collection
- This difference is what makes direct ByteBuffers useful as their pointers can be directly passed to the JNI layer

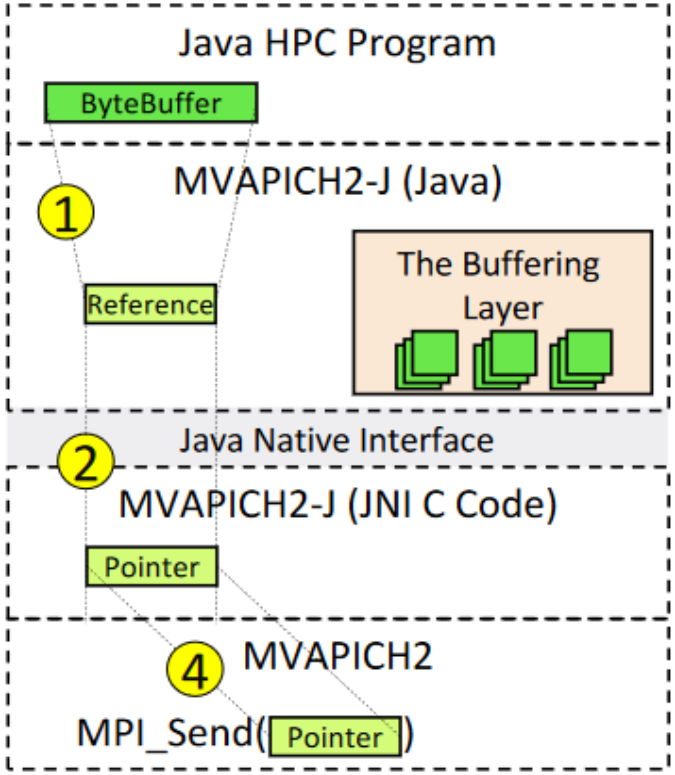


*The layout of Direct/Non-direct ByteBuffers and Java Arrays in the JVM

Communication of Java arrays vs. direct ByteBuffers



Communicating Java Arrays



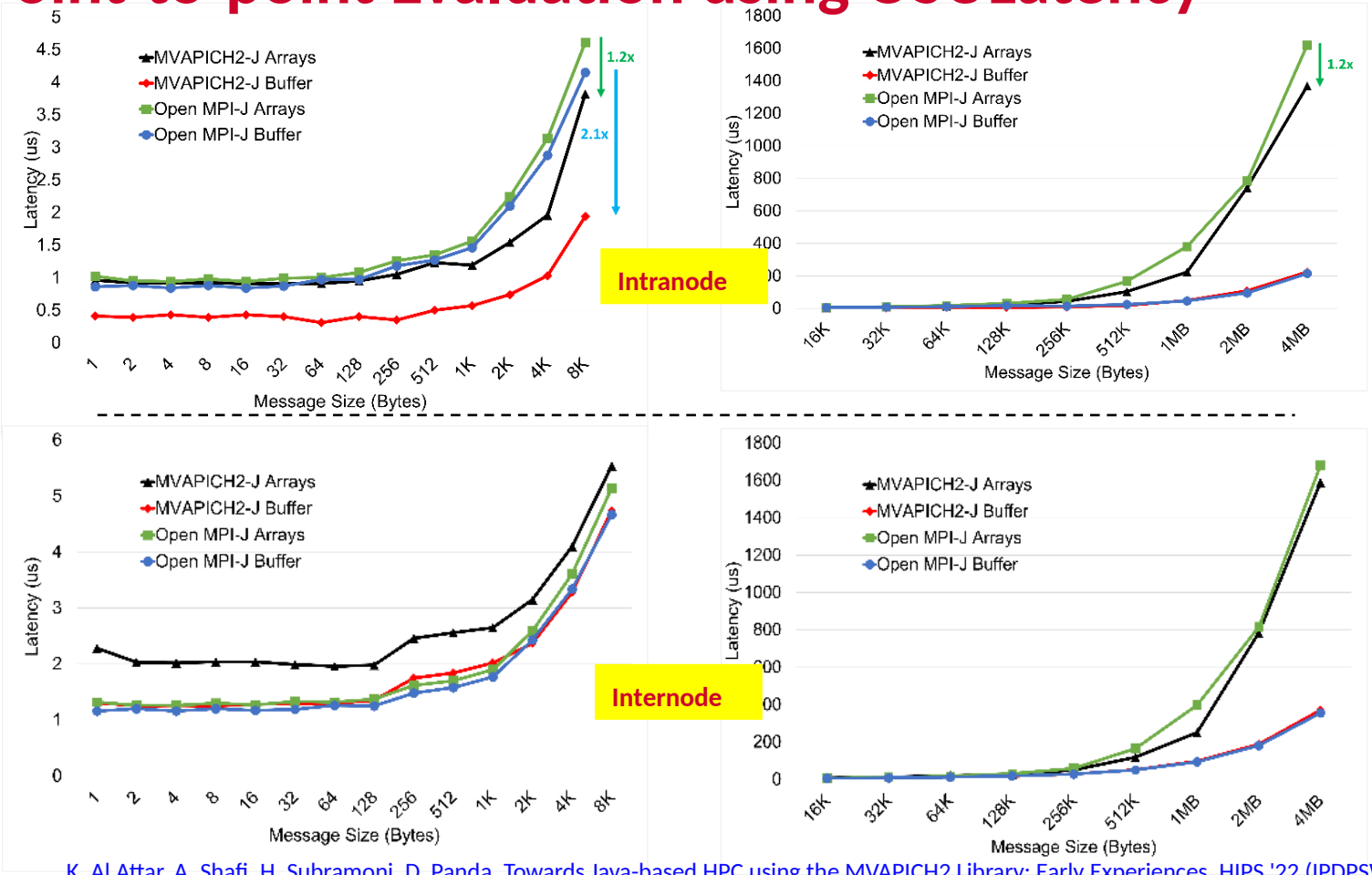
Communicating Direct ByteBuffers

Java extensions to OMB

- Java extensions to OMB include:
  - point-to-point primitives (latency, bandwidth, and bi-bandwidth)
  - vectored and blocking collective communication primitives (latency)

Point-to-Point	Collectives	Vectored Collectives
OSULatency	OSUAllgather	OSUAllgatherv
OSUBandwidth	OSUAlltoall	OSUAlltoallv
OSUBiBandwidth	OSUGather	OSUGatherv
- OSUBandwidthOMPI	OSUScatter	OSUScatterv
- OSUBiBandwidthOMPI	OSUReduce	
	OSUAllReduce	
	OSUBcast	
	OSUBarrier	
	OSUReduceScatter	

Point-to-point Evaluation using OSULatency

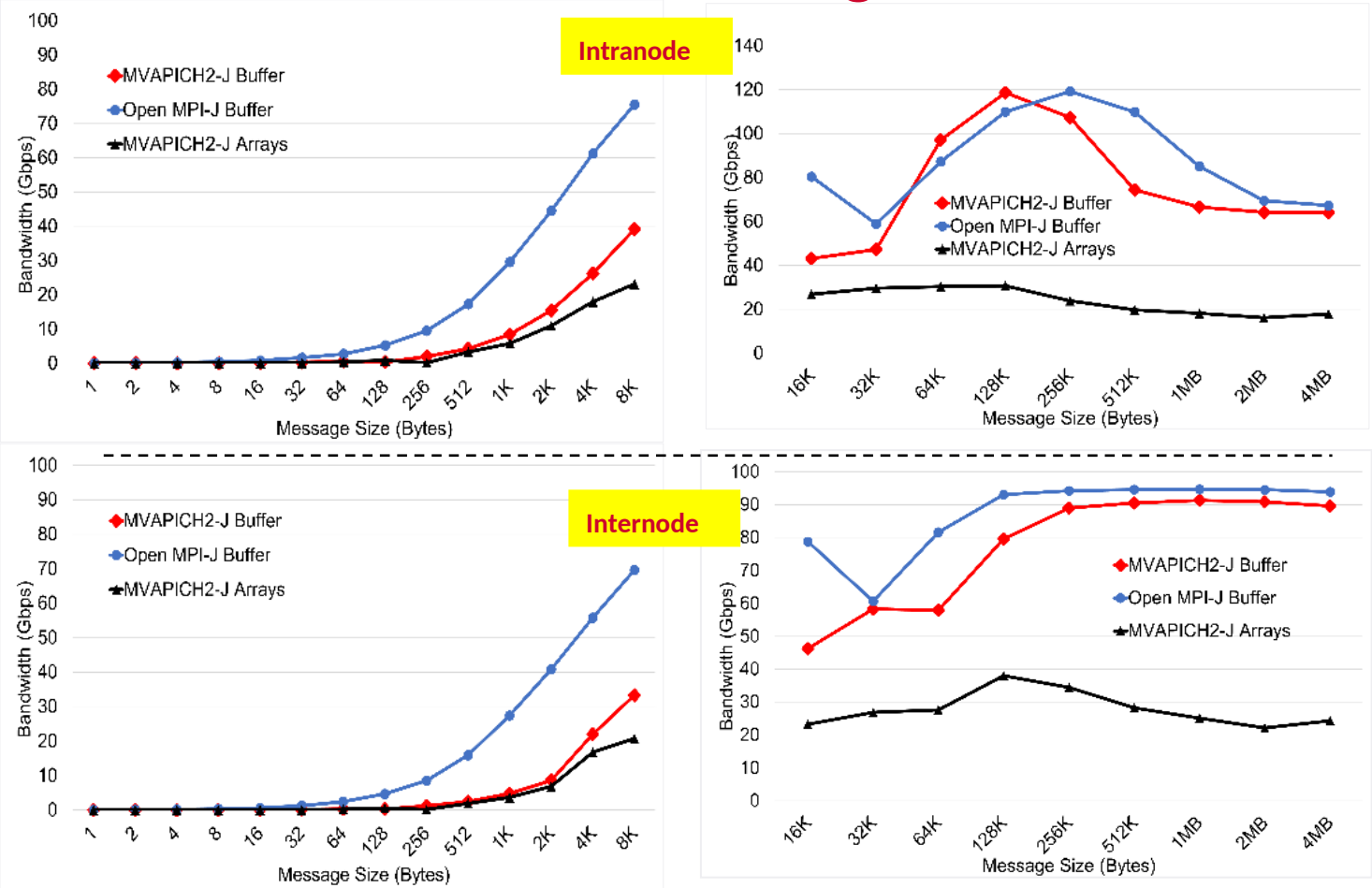


Configuration:

- Cluster: TACC Frontera
- Cores/Node: 56
- CPU: Intel Cascade Lake (Xeon Platinum 8280)
- Memory: 192 GB
- Clock-speed: 2.7 GHz
- Network: Mellanox (HDR-100)
- MVAPICH2-X v2.3.6
- Open MPI v4.1.2
- UCX 1.13.0
- Open MPI Java Bindings
- MVAPICH2-J 2.3.7

K. Al Attar, A. Shafi, H. Subramoni, D. Panda, Towards Java-based HPC using the MVAPICH2 Library: Early Experiences, HIPS '22 (IPDPSW), May 2022.

Point-to-point Evaluation using OSUBandwidth

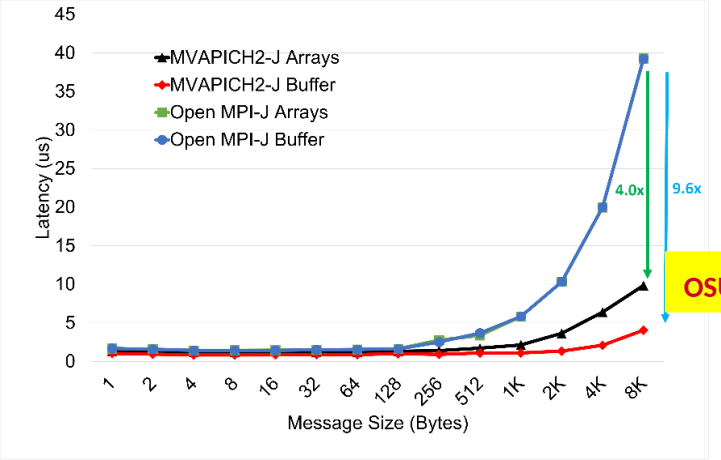


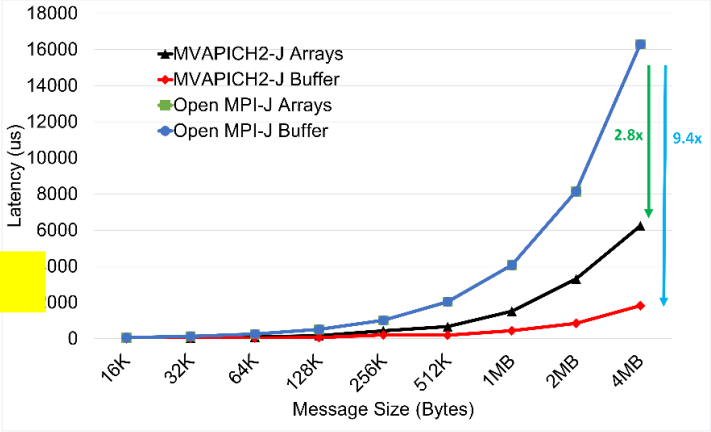
Configuration:

- Cluster: TACC Frontera
- Cores/Node: 56
- CPU: Intel Cascade Lake (Xeon Platinum 8280)
- Memory: 192 GB
- Clock-speed: 2.7 GHz
- Network: Mellanox (HDR-100)
- MVAPICH2-X v2.3.6
- Open MPI v4.1.2
- UCX 1.13.0
- Open MPI Java Bindings
- MVAPICH2-J 2.3.7

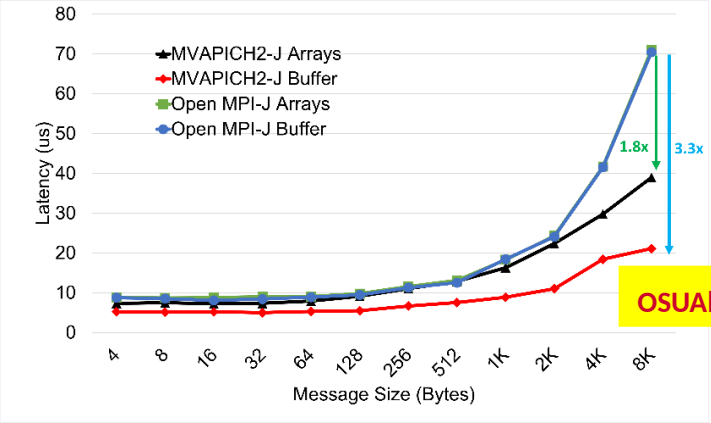
Ibid.

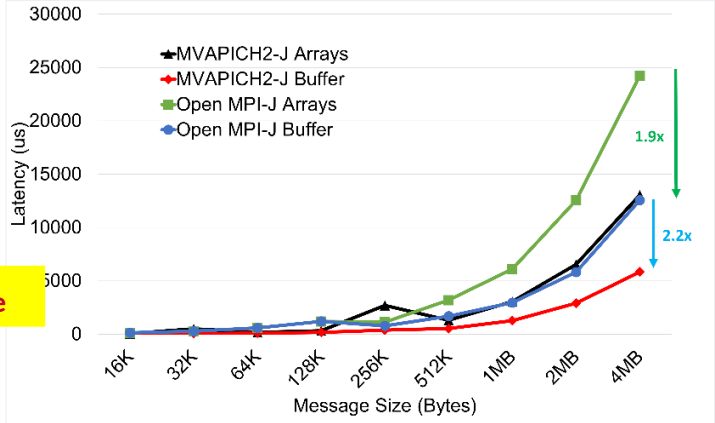
Collectives Evaluation





OSUBcast





OSUAllreduce

Configuration:

- Cluster: TACC Frontera
- Cores/Node: 56
- CPU: Intel Cascade Lake (Xeon Platinum 8280)
- Memory: 192 GB
- Clock-speed: 2.7 GHz
- Network: Mellanox (HDR-100)
- MVAPICH2-X v2.3.6
- Open MPI v4.1.2
- UCX 1.13.0
- Open MPI Java Bindings
- MVAPICH2-J 2.3.7

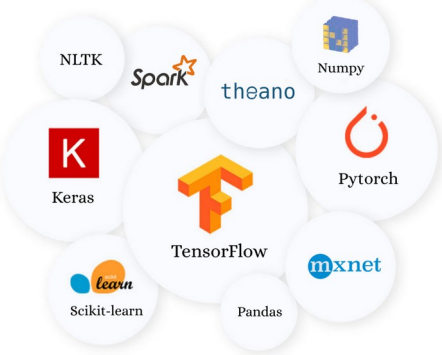
Ibid.

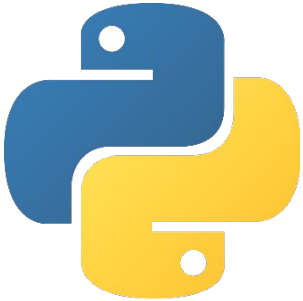
Agenda

- OSU MPI Micro-Benchmarks (OMB) Suite
- Java Extensions to OMB
  - MVAPICH2-J: Java Bindings to MVAPICH2
  - Performance Evaluation using Java OMB benchmarks
- **Python Extensions to OMB**
  - **Benchmarks Implementation**
  - **Performance Characterization using Python OMB benchmark**
- Summary

High Performance Computing with Python

- Python has become a dominant programming language for emerging areas like Machine Learning (ML), Deep Learning (DL), and Data Science (DS).
- Python has a rapidly growing community and support for prominent scientific libraries and frameworks with a flexible and simplified syntax.
- ML, DL, and DS applications are computationally intensive tasks that can be accelerated by harnessing the compute power offered by HPC.





Courtesy: https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c

Why Python?

Flexible and simplified syntax.

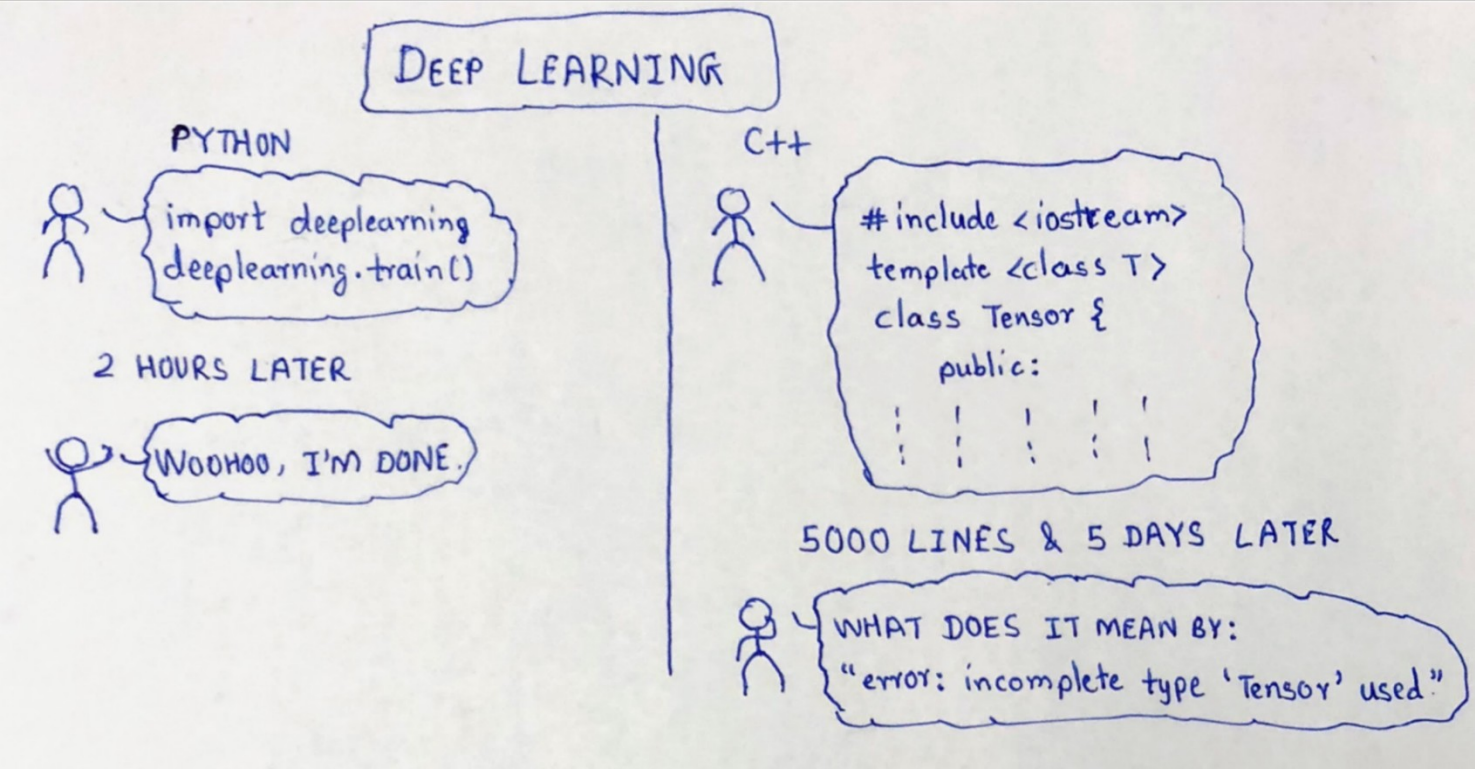
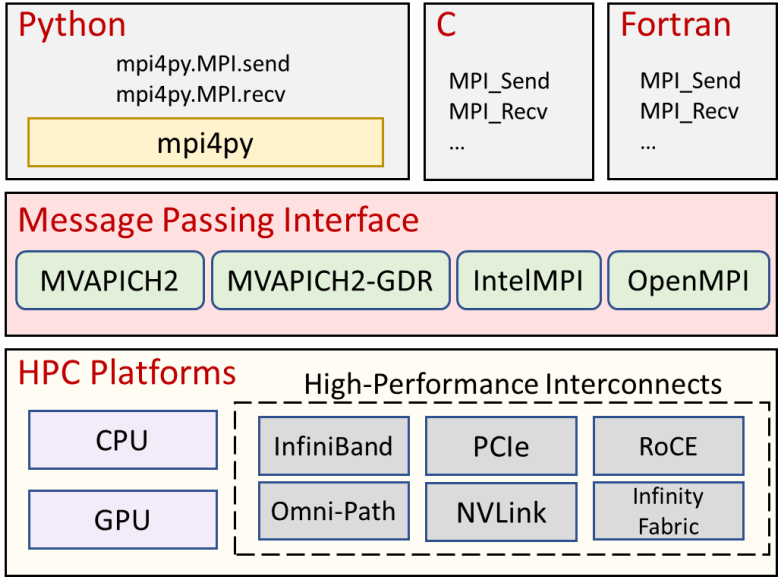


Image source: https://www.hardikp.com/2017/12/30/python-cpp/

MPI for Python

- Message Passing Interface (MPI) is considered the de-facto standard that defines communication operations for exchanging data in parallel computing environments.
- The MPI standard only provides bindings for the C and Fortran programming languages.
- To use MPI with higher-level programming languages such as Python, a communication wrapper library is needed to provide MPI-like bindings.
- mpi4py is a widely used package that provides a Python-based MPI interface which is built on top of an MPI library.



C and Fortran can directly call MPI operations whereas Python needs a wrapper to provide MPI-like bindings.

Package Comparison

- Python extensions to the open-source OMB suite aimed to evaluate communication performance of MPI-based parallel applications in Python.
- Performance characterization of MPI communication in Python on four HPC systems:
  - Point-to-point and collective communication operations using OMB as a baseline performance in C.
  - Evaluation on CPU and GPU devices for different buffers including Bytearrays, Numpy, CuPy, PyCUDA and Numba.
  - Pickle method evaluation for serializing communicated objects.
- Analysis of the overhead presented by mpi4py over native MPI libraries.

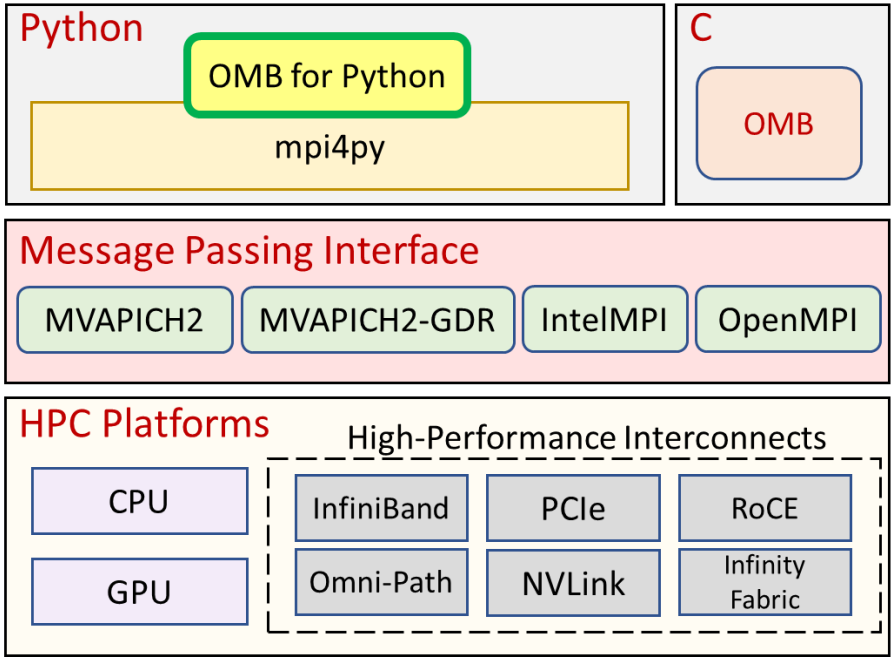
	Point-to-point	Blocking Collectives	Vector Variants	Support for Python	Bytearray Buffers	Numpy Buffers	CuPy Buffers	PyCUDA Buffers
OMB for Python	✓	✓	✓	✓	✓	✓	✓	✓
mpi4py Demo Codes [1]	✓	Partial	Partial	✓	X	✓	X	X
IMB [2]	✓	✓	✓	X	X	X	X	X
SMB [3]	✓	X	X	X	X	X	X	X

Feature Comparison Between Benchmark Packages

[1] L. Dalcin, R. Paz, and M. Storti, MPI for Python, Journal of Parallel and Distributed Computing, 65(9):1108-1115, 2005. https://doi.org/10.1016/j.jpdc.2005.03.010
[2] "Sandia MPI Micro-Benchmark Suite (SMB)." http://www.cs.sandia.gov/smb/index.html
[3] "Intel MPI Benchmarks (IMB)." https://software.intel.com/en-us/articles/intel-mpi-benchmarks

N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

Hierarchy and Supported Benchmarks



Architectural hierarchy of OMB for Python with mpi4py, MPI, and HPC platforms

Point-to-Point	Bi-directional bandwidth, Bandwidth, Latency, Multi latency
Blocking Collectives	Allgather, Allreduce, Alltoall, Barrier, Bcast, Gather, Reduce_scatter, scatter
Vector Variant	Allgatherv, Alltoallv, Gatherv, Scatterv

Point-to-Point, blocking collectives, and vector variant benchmarks supported by OMB for Python

N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

Experimental Setup

	Frontera	Stampede2	RI2	Bridges-2
CPU	Two Intel Xeon Platinum 8280 (Cascade Lake). 28 cores per socket (56 per node) @2.70GHz	Two Intel(R) Xeon(R) Platinum 8160 (Skylake). 24 cores per socket (48 per noe) @2.70GHz	Two Intel(R) Xeon(R) Gold 6132 with 14 cores (28 cores per node) @2.40GHz.	Two Intel Xeon Gold 6248 (Cascade Lake). 20 cores per socket (40 cores per node) @ 2.50GHz
RAM	192GB of RAM per node.	192GB of RAM per node.	128GB of RAM per node	512GB of RAM per node.
Interconnect	Mellanox InfiniBand HDR	Intel Omni-Path	Mellanox InfiniBand	Mellanox InfiniBand HDR
GPU	N/A	N/A	N/A	Eight NVIDIA Tesla V100-32GB SXM2 per node

Configuration of nodes used on each of the experimental systems

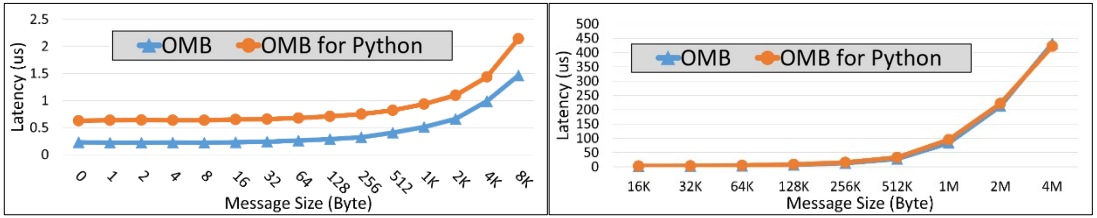
Software packages:

- For CPU experiments: MVAPICH2 2.3.6, OMB v5.8, mpi4py 3.1.1
- For GPU experiments: MVPICH2-GDR 2.3.6, CUDA 11.2, OMB v5.8, mpi4py 3.1.1

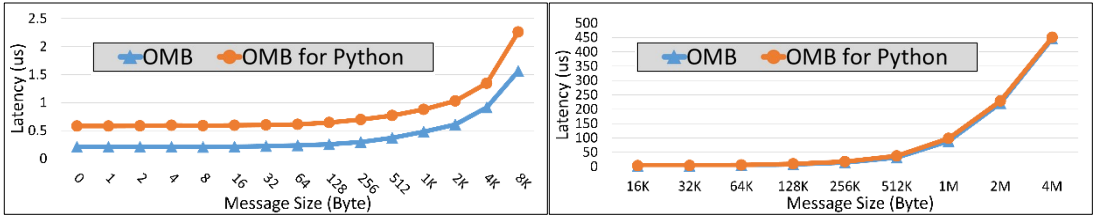
N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

Point-to-Point Evaluation on CPU

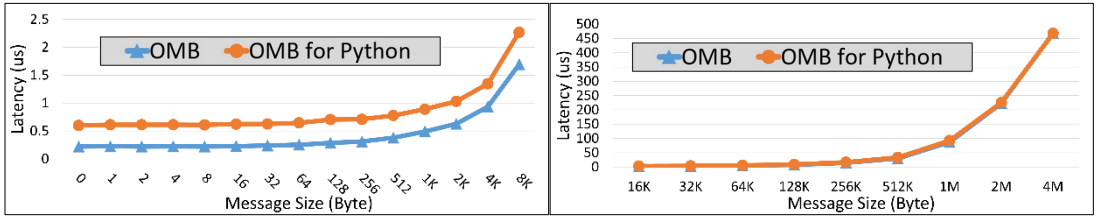
CPU communication latency for small and large message sizes comparing OMB and OMB for Python benchmarks



Frontera



Stampede2

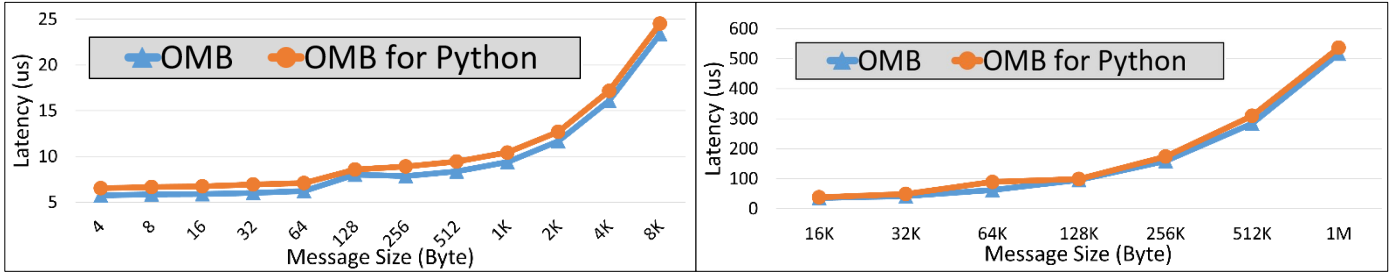


RI2

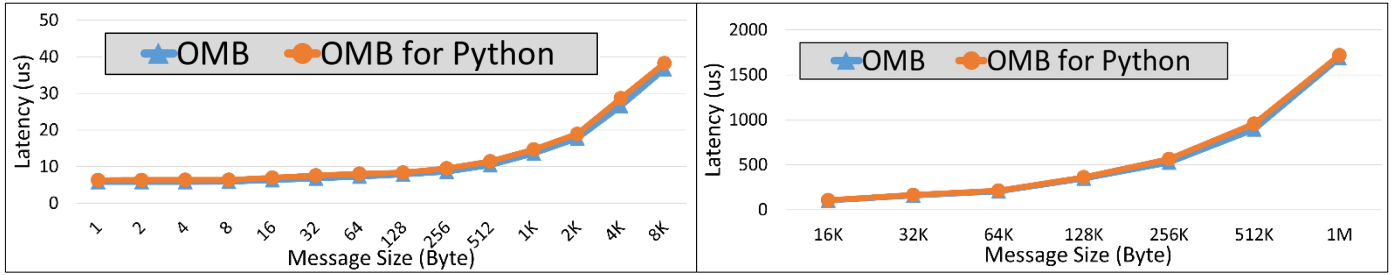
- Consistent trends across three clusters.
- Average overhead of 0.44, 0.41, and 0.41 on Frontera, Stampede2, and RI2 respectively for small message sizes and 2.31, 4.13, 1.76 microseconds for large message sizes.

N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

Collectives Evaluation on CPU



Allreduce CPU communication latency for small and large message sizes on 16 nodes on the Frontera Cluster

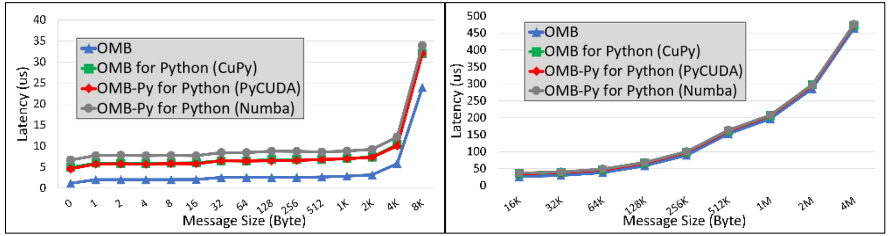


Allgather CPU communication latency for small and large message sizes on 16 nodes on the Frontera Cluster

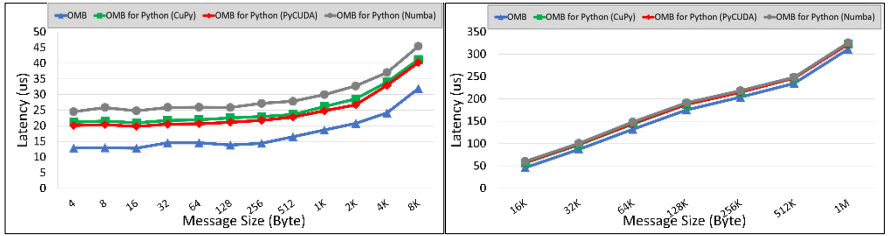
Average overheads of 0.93 and 0.92 microseconds for Allreduce and Allgather respectively for small message sizes. 14.13 and 23.4 for large message sizes.

N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

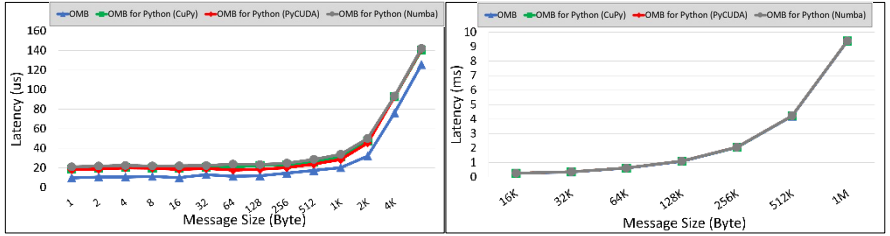
Evaluation on GPU



Point-to-Point GPU communication latency for small and large message sizes on Bridges-2



Allreduce GPU communication latency for small and large message sizes on Bridges-2 (2 nodes – 8 GPUs)



Allgather GPU communication latency for small and large message sizes on Bridges-2 (2 nodes – 8 GPUs)

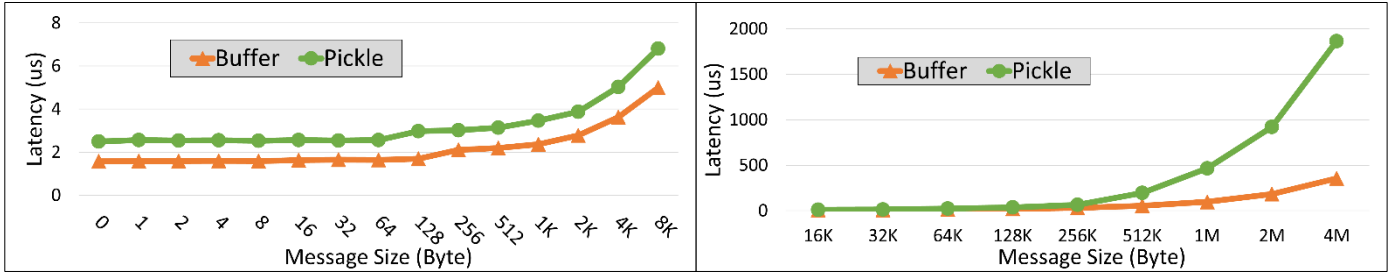
CuPy, PyCUDA, and Numba libraries allow initializing different types of data buffers directly on the GPU to carry out complex matrix operations. In these benchmarks, communication happens directly from/to the GPU by utilizing these GPU buffers.

Across all benchmarks, CuPy and PyCUDA show better MPI communication performance on the GPU compared to Numba.

N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

Pickle Method Evaluation

- mpi4py offers a built-in feature for serialization of the communicated Python objects.
- This is mainly referred to as “pickling” when an object is converted into a byte stream and “unpickling” when it is converted back to its original format.
- In mpi4py, the MPI methods that use the pickle method are defined with a lower case first letter such as send(), recv(), reduce(), allgather, etc. The direct buffer methods (no serialization) are defined with upper case first letter such as Send(), Recv(), Reduce(), Allgather(), etc.



CPU latency for small and large message sizes using OMB for Python to compare the pickle and direct buffer methods on Frontera

N. Alnaasan , A. Jain , A. Shafi , H. Subramoni , and DK Panda, OMB-Py: Python Micro-Benchmarks for Evaluating Performance of MPI Libraries on HPC Systems

Agenda

- OSU MPI Micro-Benchmarks (OMB) Suite
- Java Extensions to OMB
  - MVAPICH2-J: Java Bindings to MVAPICH2
  - Performance Evaluation using Java OMB benchmarks
- Python Extensions to OMB
  - Benchmarks Implementation
  - Performance Characterization using Python OMB benchmark
- **Summary**

Summary

- This talk presented the following extensions to OMB:
  - Java benchmarks
  - Python benchmarks
- The Java and Python benchmarks provide a variety point-to-point and collectives benchmarks:
  - Presented performance evaluation using MVAPICH2-J, Open MPI Java Bindings, and mpi4py
- Python extensions have support for communicating data to/from both CPU and GPU buffers
- Java and Python extensions have been released as part of the OMB 6.0 release:
  - https://mvapich.cse.ohio-state.edu/benchmarks/

Thank You!

shafi.16@osu.edu, alnaasan.1@osu.edu

Network Based Computing Laboratory

Follow us on

https://twitter.com/mvapich

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

HiBD
High-Performance Big Data

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

HiDL
High-Performance Deep Learning

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