MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

HiBD
High-Performance
Big Data

HiDL
High-Performance
Deep Learning

High-Performance and Scalable Support for Big Data Stacks with MPI

Talk at the 2024 Annual MVAPICH User Group (MUG) Conference

by

Follow us on

https://x.com/mvapich

Aamir Shafi, Kinan Al Attar, Jinghan Yao

The Ohio State University

E-mail: shafi.16@osu.edu

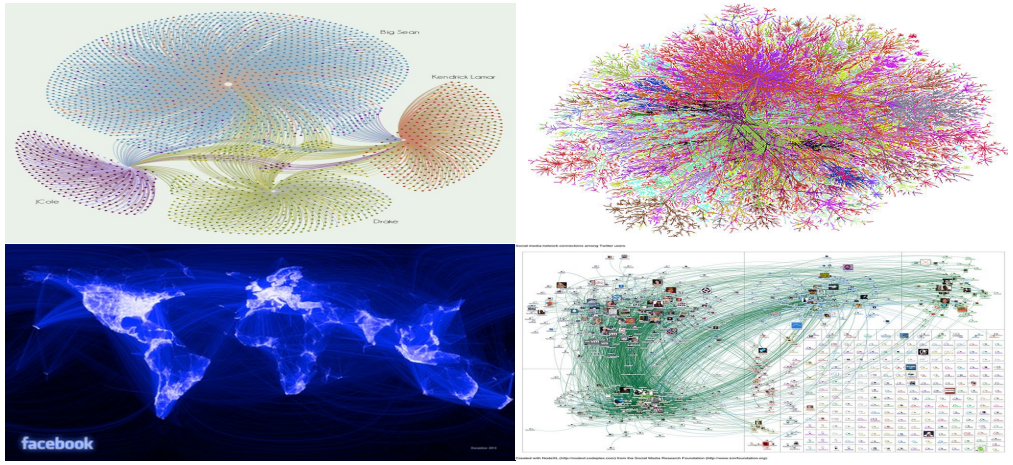
https://cse.osu.edu/people/shafi.16

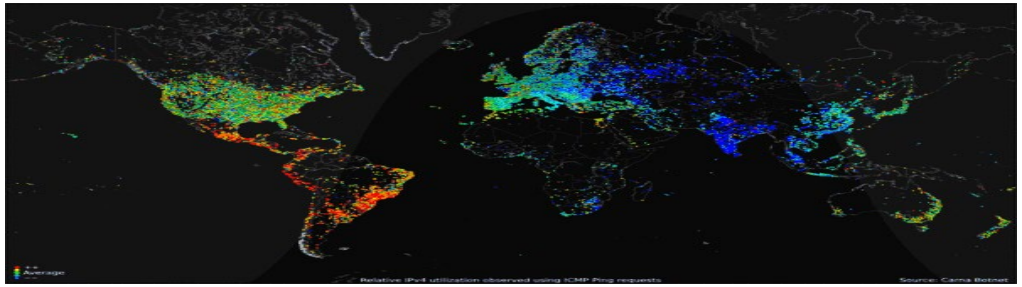
Presentation Outline

- **Introduction to Big Data Analytics**
- Overview, Design and Implementation
  - MPI4Spark
  - MPI4Dask
- Performance Evaluation
  - MPI4Spark
  - MPI4Dask
- Demo – Hands-on Exercises with MPI4Dask
- Related Publications and Summary

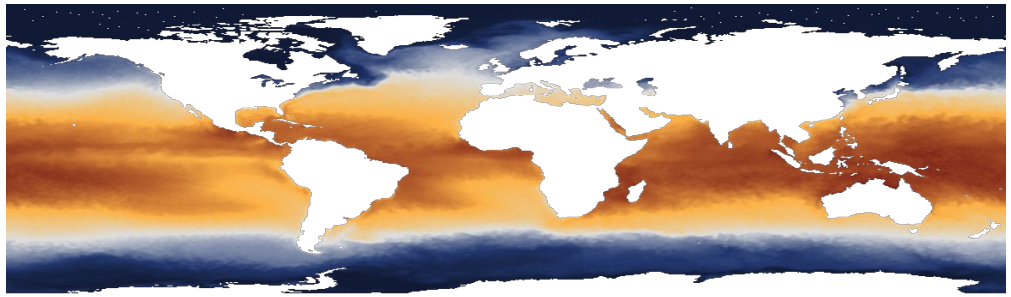
Introduction to Big Data Analytics

- Big Data has changed the way people understand and harness the power of data, both in the business and research domains
- Big Data has become one of the most important elements in business analytics
- Big Data and High Performance Computing (HPC) are converging to meet large scale data processing challenges
- Dask and Spark are two popular Big Data processing frameworks
- Sometimes also called Data Science



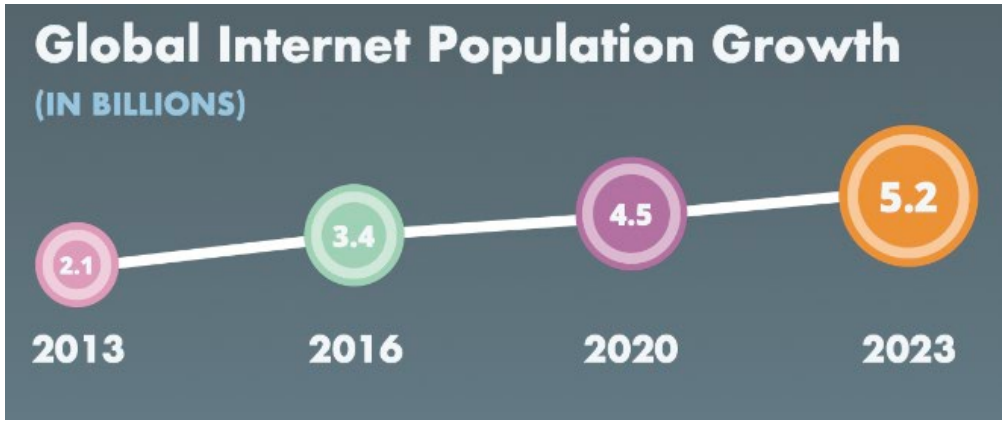


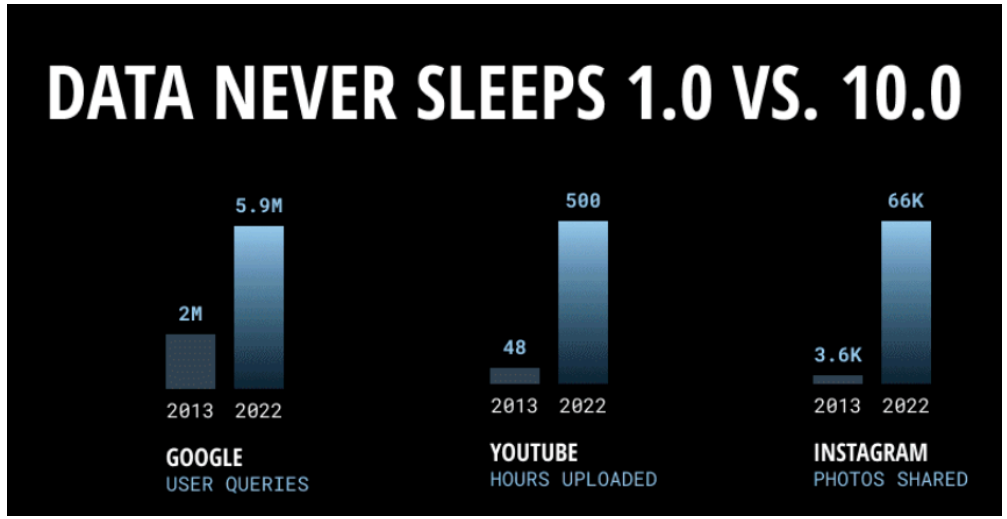
http://www.coolinfographics.com/blog/tag/data?currentPage=3



http://www.climatecentral.org/news/white-house-brings-together-big-data-and-climate-change-17194

Big Velocity – How Much Data Is Generated Every Minute on the Internet?





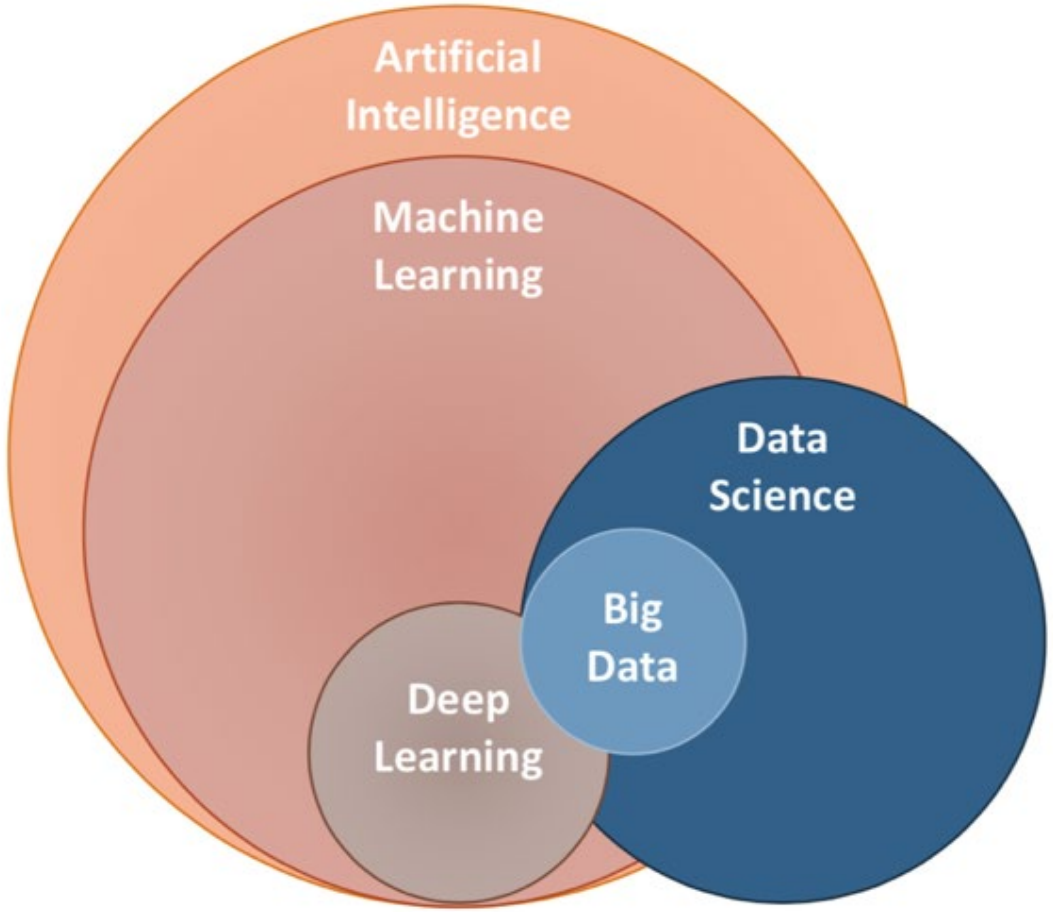
As of Nov 2023, the Internet reaches around 64.6% of the population and now represents

5.2 Billion People.

Courtesy: https://www.domo.com/blog/data-never-sleeps-11/

Intersection of Big Data and ML/DL

- Big Data, Machine and Deep learning are closely related and interconnected
- ML/DL workloads require collecting and processing of data
- HPC systems and distributed environments enable larger models and data to be trained
  - Growing quantities of training data requires Big Data solutions
- DL workloads pushing beyond traditional NLP and computer vision applications
  - Moving toward real-time analysis of streaming data



Big Data-ML/DL Venn Diagram

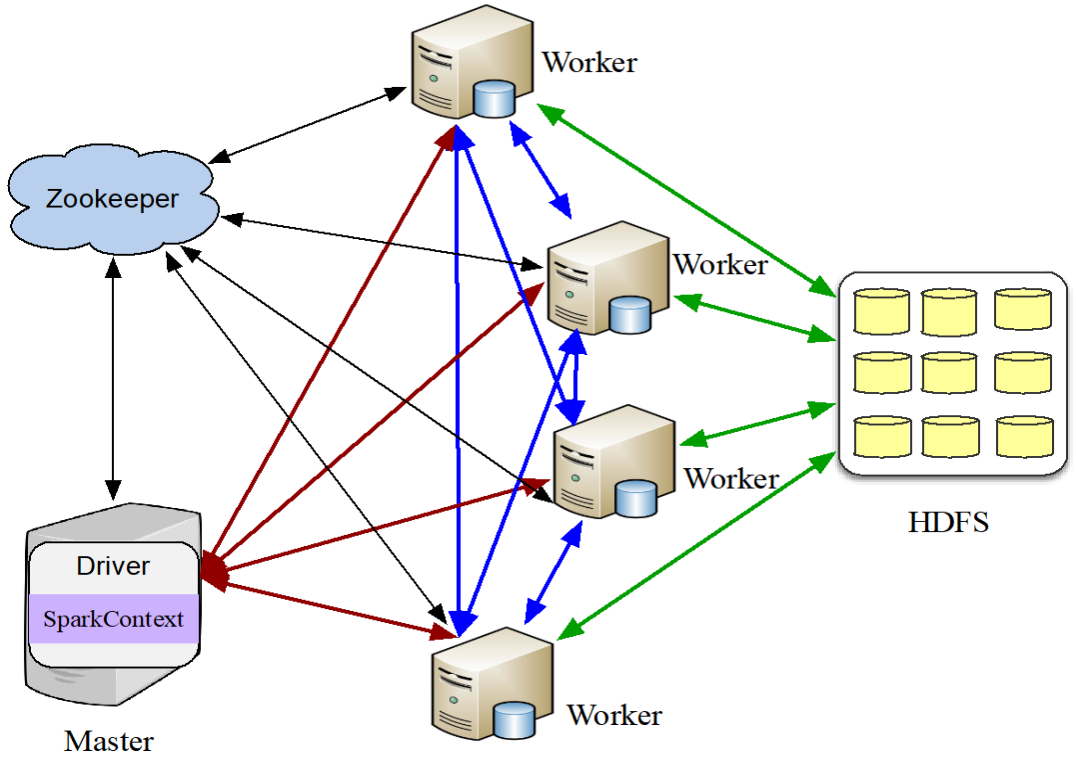
Courtesy: Thakur, N. (2023, February 25). *The differences between Data Science, Artificial Intelligence, Machine Learning, and Deep Learning*. Medium. Retrieved April 21, 2023, from https://ai.plainenglish.io/data-science-vs-artificial-intelligence-vs-machine-learning-vs-deep-learning-50d3718d51e5

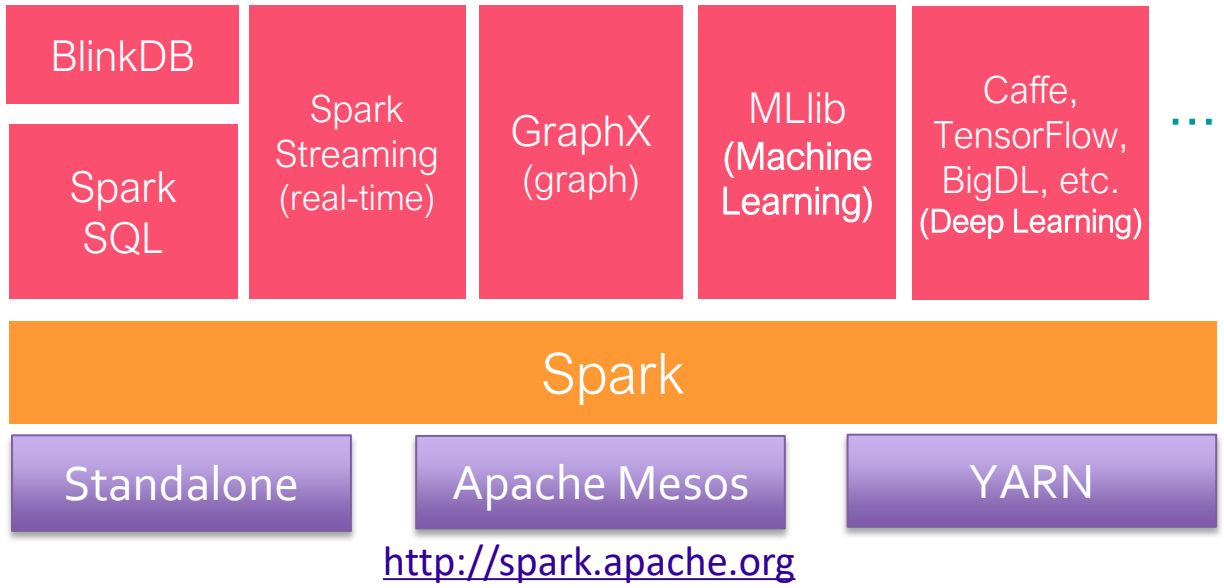
Presentation Outline

- Introduction to Big Data Analytics
- **Overview, Design and Implementation**
  - **MPI4Spark**
  - **MPI4Dask**
- Performance Evaluation
  - MPI4Spark
  - MPI4Dask
- Demo – Hands-on Exercises with MPI4Dask
- Related Publications and Summary

The Apache Spark Framework

- An in-memory data-processing framework
  - Iterative machine learning jobs
  - Interactive data analytics
  - Scala based Implementation
  - Standalone, YARN, Mesos
- A unified engine to support Batch, Streaming, SQL, Graph, ML/DL workloads
- Scalable and communication intensive
  - Wide dependencies between Resilient Distributed Datasets (RDDs)
  - MapReduce-like shuffle operations to repartition RDDs
  - Sockets based communication

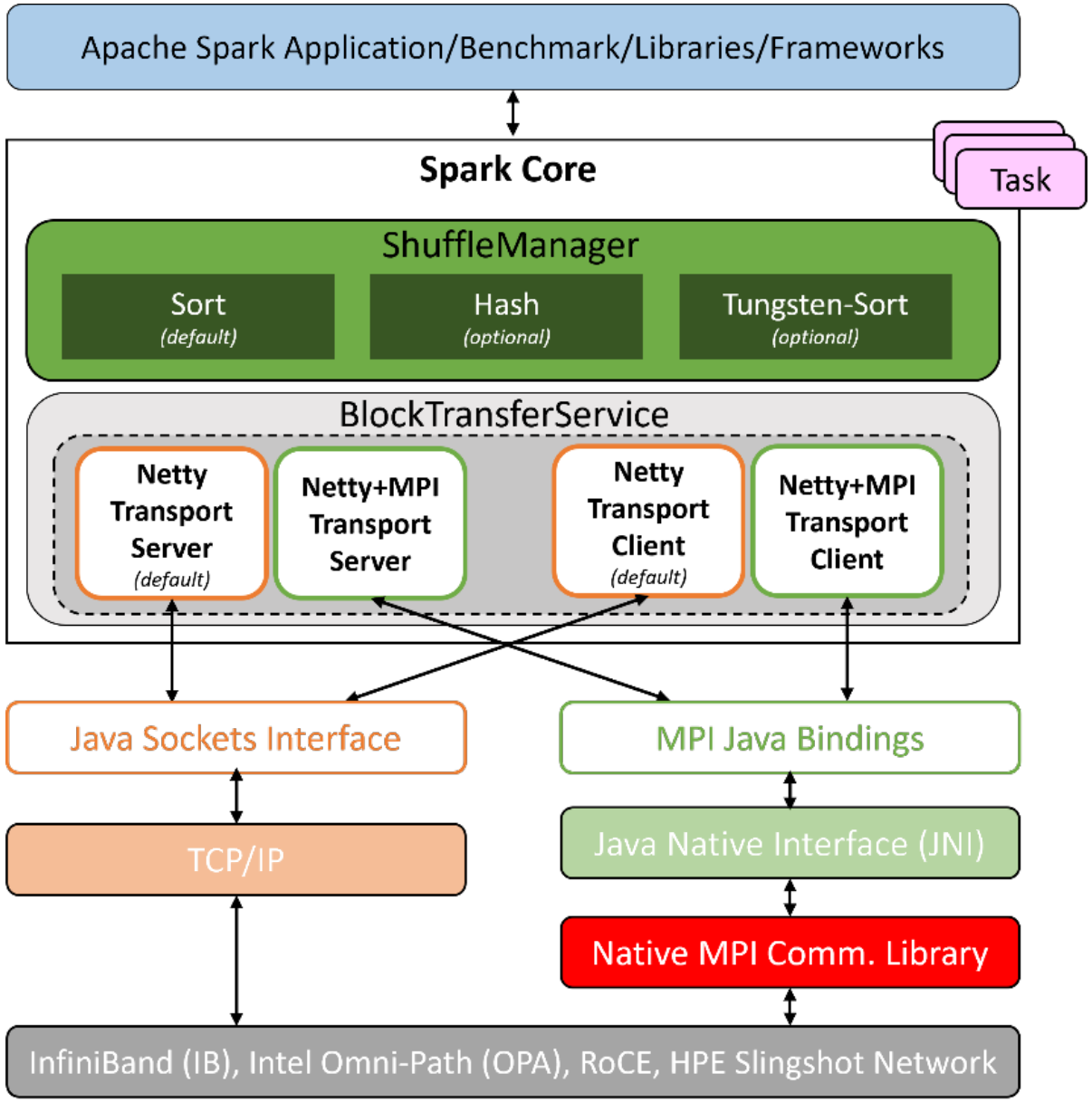




http://spark.apache.org

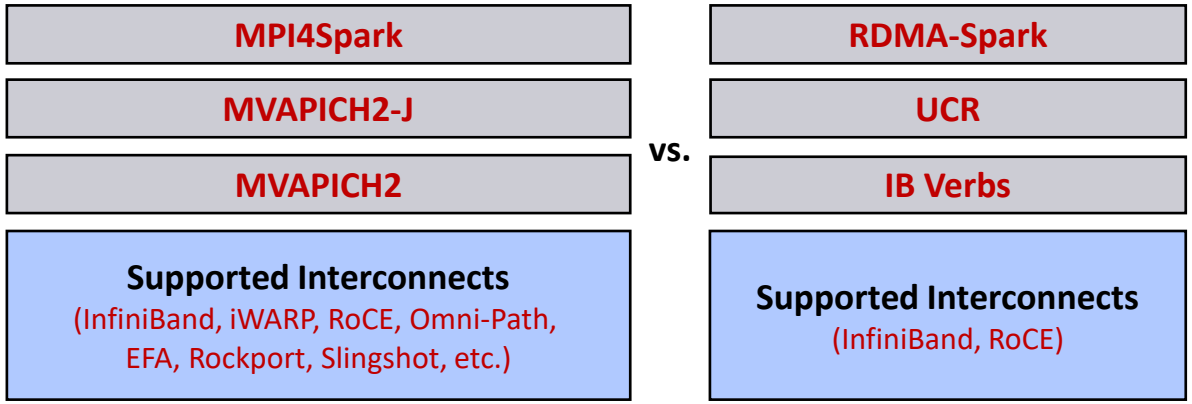
MPI4Spark: Using MVAPICH2 to Optimize Apache Spark

- The main motivation of this work is to utilize the communication functionality provided by MVAPICH2 in the Apache Spark framework
  - MPI4Spark relies on Java bindings of the MVAPICH2 library
- Spark's default Shuffle Manager relies on Netty for communication:
  - Netty is a Java New I/O (NIO) client/server framework for event-based networking applications



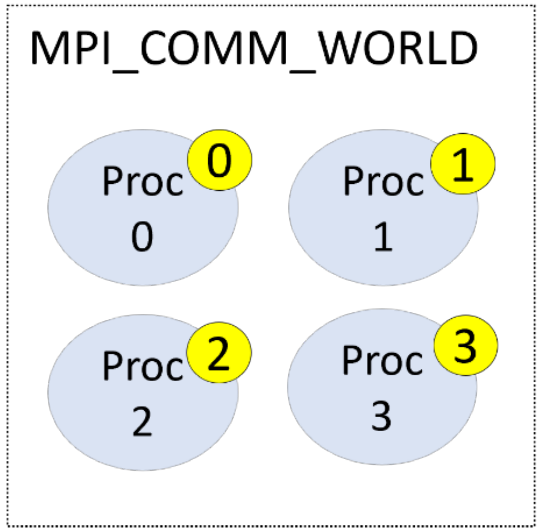
MPI4Spark Interconnect Support

- The current approach is different from its predecessor design, RDMA-Spark (http://hibd.cse.ohio-state.edu)
  - RDMA-Spark supports only InfiniBand and RoCE
  - Requires new designs for new interconnect
- MPI4Spark supports multiple interconnects/systems through a common MPI library
  - Such as InfiniBand (IB), Intel Omni-Path (OPA), HPE Slingshot, RoCE, and others
  - No need to re-design the stack for a new interconnect as long as the MPI library supports it

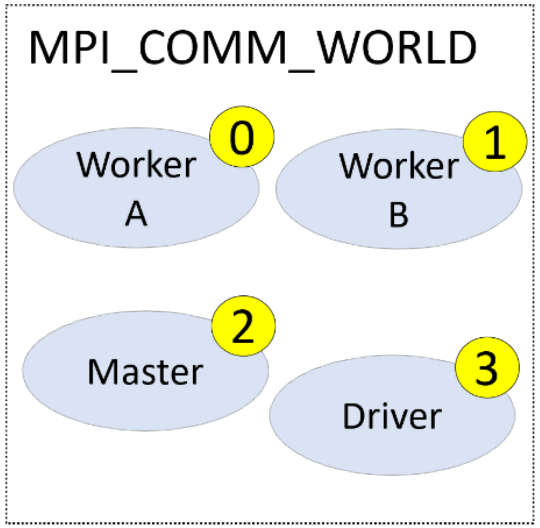


Launching Spark using MPI with Dynamic Process Management

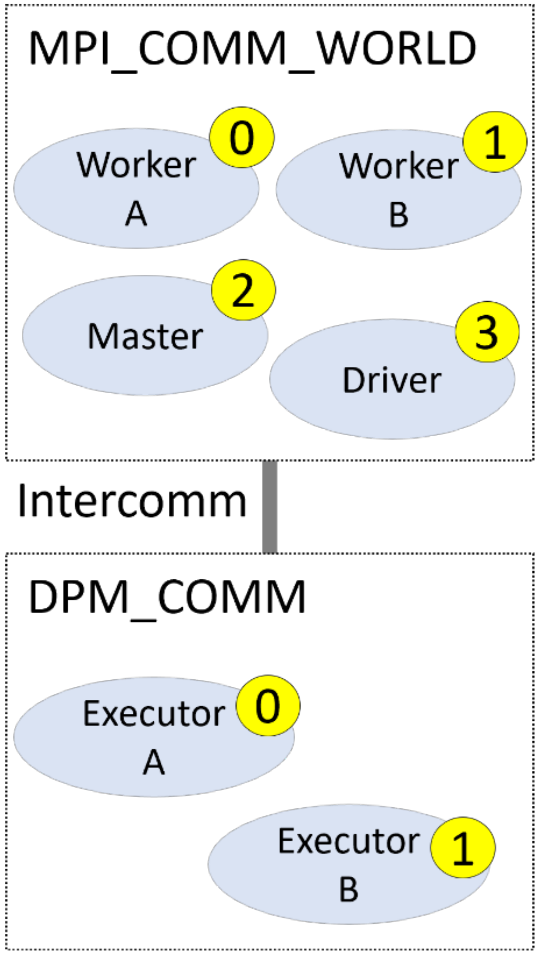
Step A: Launch 4 Wrapper Processes (for e.g. mpiexec –np 4 .. SparkMPI.java)

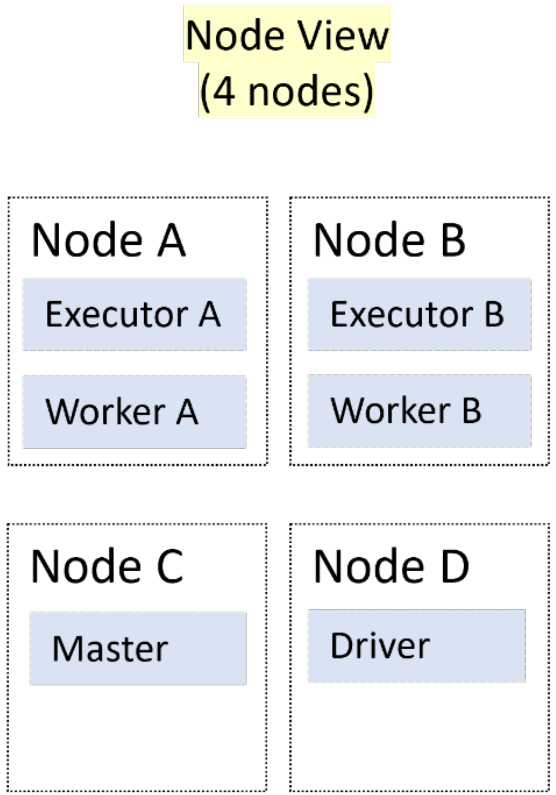


Step B: Each Wrapper Process Forks Spark Processes



Step C: Launch 2 Executor Processes MPI_Comm_spawn_multiple()





MPI4Spark Release (v0.3)

- MPI4Spark 0.3 release adds support for the YARN cluster manager:
  - Can be downloaded from http://hibd.cse.ohio-state.edu
- Features:
  - Based on Apache Spark 3.3.0
  - Support for YARN cluster manager
  - Compliant with user-level Apache Spark APIs and packages
  - High performance design that utilizes MPI-based communication
    - Utilizes MPI point-to-point operations
    - (NEW) Enhanced MPI Dynamic Process Management (DPM) logic for launching executor processes for the standalone cluster manager
    - (NEW) Relies on Multiple-Program-Multiple-Data (MPMD) launcher mode for the YARN and the Standalone cluster managers
    - (NEW) Supports MVAPICH versions 2.3.7 and 4.0
  - Built on top of the MVAPICH2-J Java bindings for MVAPICH2 family of MPI libraries
  - Tested with
    - (NEW) OSU HiBD-Benchmarks, GroupBy and SortBy
    - (NEW) Intel HiBench Suite, Micro Benchmarks, Machine Learning and Graph Workloads
    - Mellanox InfiniBand adapters (EDR and HDR 100G and 200G)
    - HPC systems with Intel OPA interconnects
    - Various multi-core platforms

Presentation Outline

- Introduction to Big Data Analytics
- **Overview, Design and Implementation**
  - **MPI4Spark**
  - **MPI4Dask**
- Performance Evaluation
  - MPI4Spark
  - MPI4Dask
- Demo – Hands-on Exercises with MPI4Dask
- Related Publications and Summary

Introduction to Dask

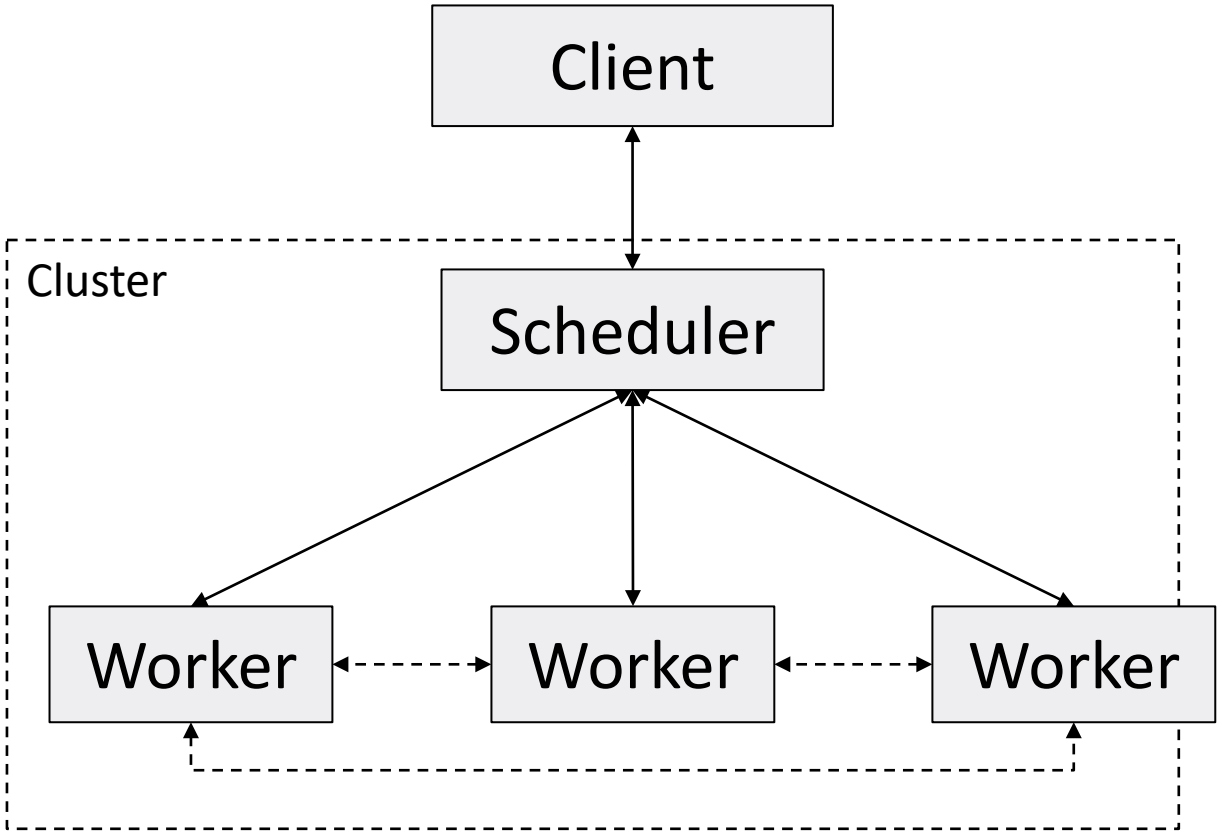
- Dask is a popular task-based distributed computing framework:
  - Scales Python applications from laptops to high-end systems
  - Builds a task-graph that is executed lazily on parallel hardware
  - Natively extends popular data processing libraries like numPy, Pandas
- Dask Distributed library supports parallel and distributed execution:
  - Built using the asyncio package that allows execution of asynchronous/non-blocking/concurrent operations called *coroutines:*
    - These are defined using async and invoked using await
  - Dask Distributed library originally had two communication backends:
    - TCP: Tornado-based
    - UCX: Built using a Cython wrapper called UCX-Py

MPI4Dask: MPI backend for Dask

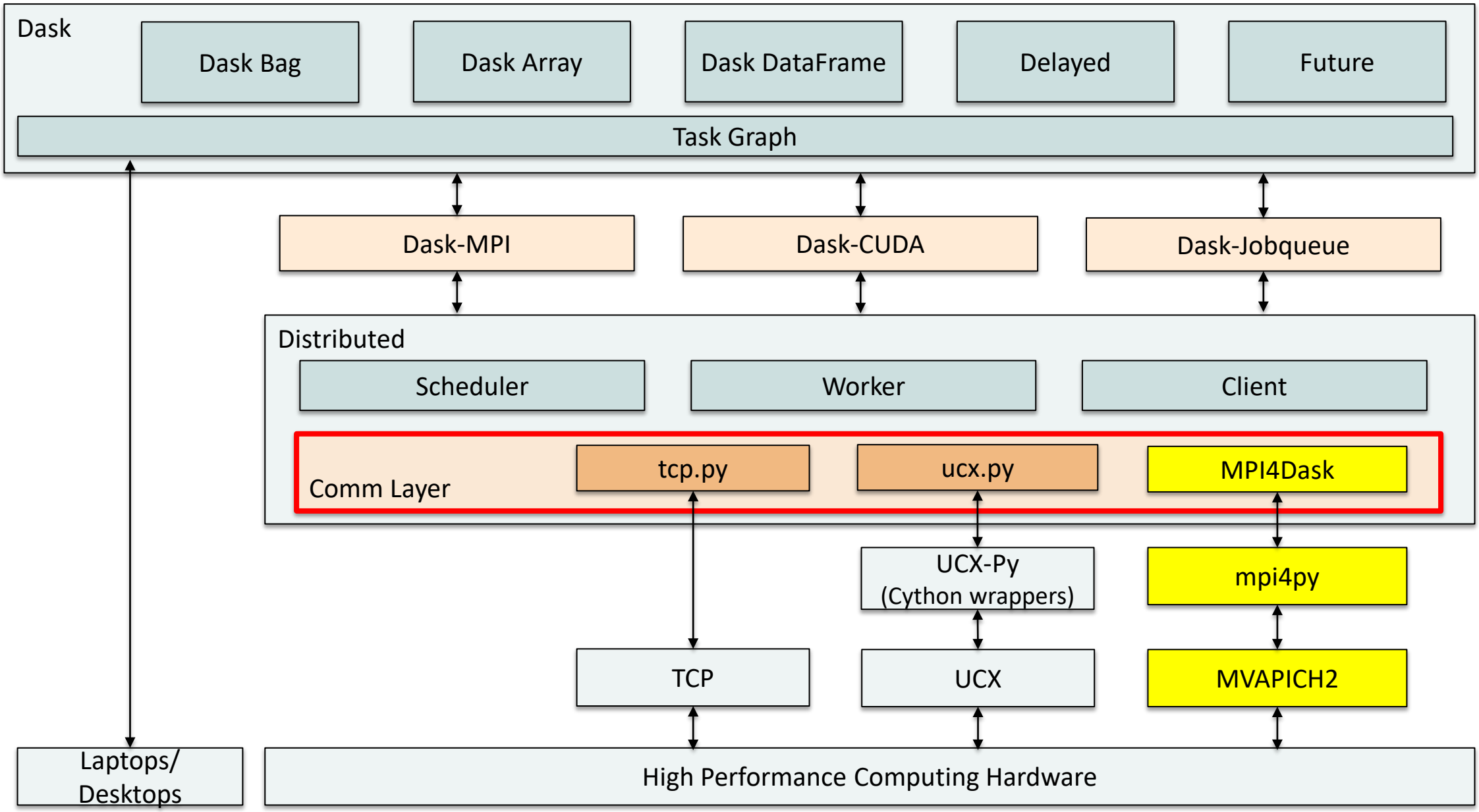
- Dask is a popular task-based distributed computing framework:
  - Scales Python applications from laptops to high-end systems
  - Builds a task-graph that is executed lazily on parallel hardware
- Dask Distributed library historically had two communication backends:
  - TCP: Tornado-based
  - UCX: Built using a GPU-aware Cython wrapper called UCX-Py
- Designed and implemented MPI4Dask communication device:
  - MPI-based backend for Dask
  - Implemented using mpi4py (Cython wrappers) and MVAPICH2
  - Uses Dask-MPI to bootstrap execution of Dask programs

Dask Distributed Execution Model

- Key characteristics:
  1. Scalability
  2. Elasticity
  3. Support for coroutines
  4. Serialization/De-serialization to data to/from GPU memory

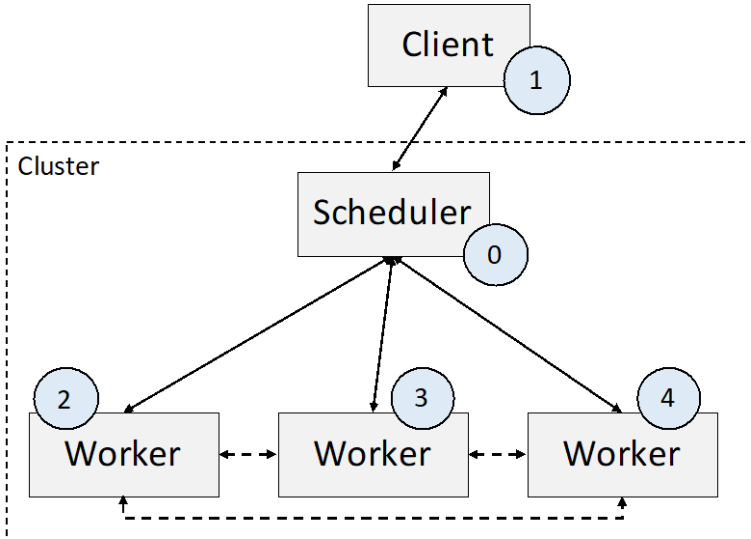


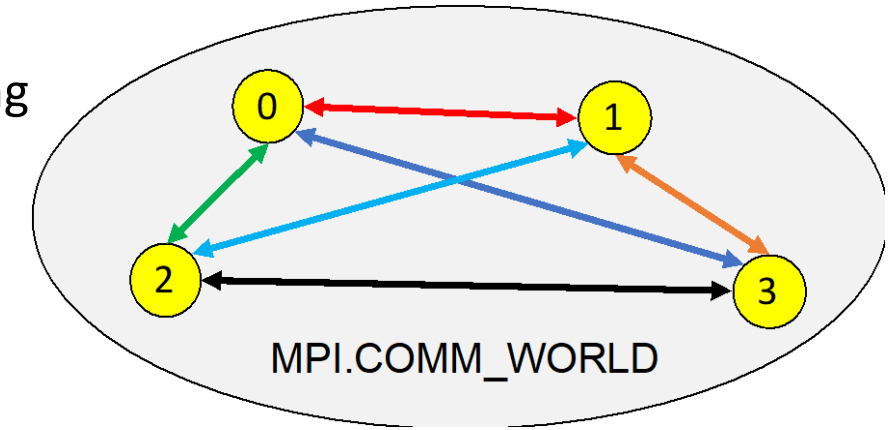
MPI4Dask in the Dask Architecture



MPI4Dask: Bootstrapping and Dynamic Connectivity

- Several ways to start Dask programs:
  - Manual
  - Utility classes:
    - LocalCUDACluster, SLURMCluster, SGECluster, PBCCluster, and others
- MPI4Dask uses the Dask-MPI to bootstrap execution of Dask programs
- Dynamic connectivity is established using the asyncio package in MPI4Dask:
  - Scheduler and workers listen for incoming connections by calling asyncio.start_server()
  - Workers and client connect using asyncio.open_connection()





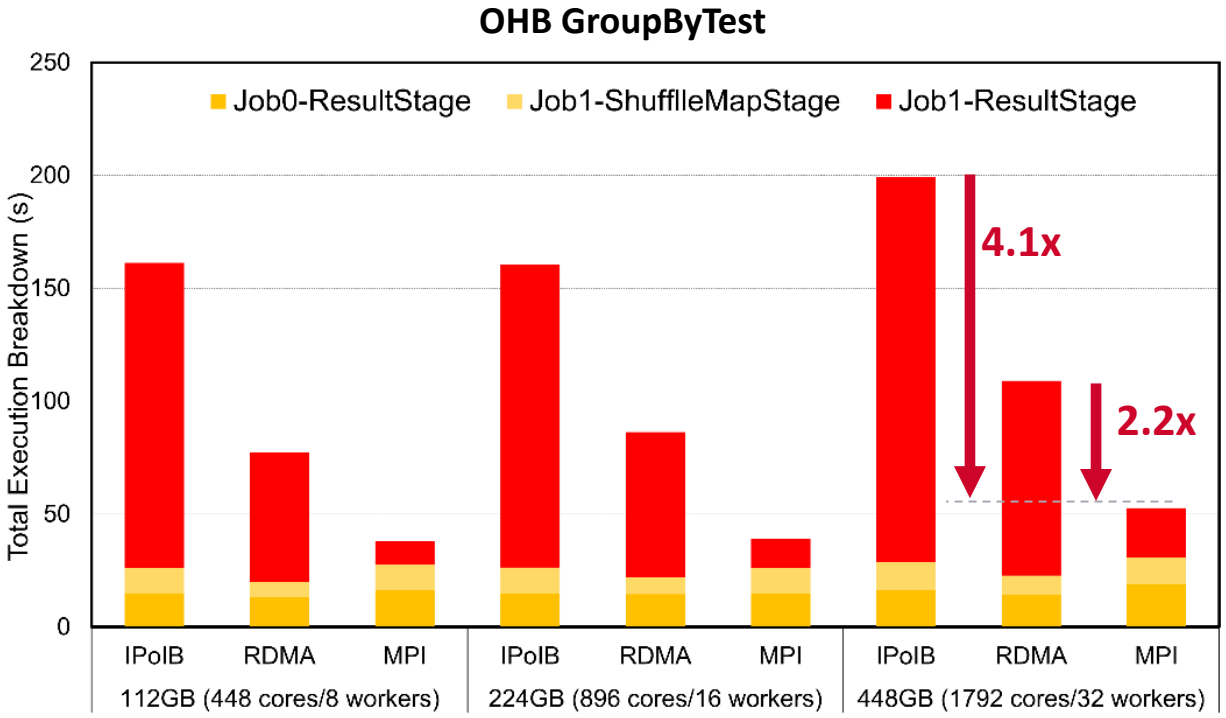
MPI4Dask Release

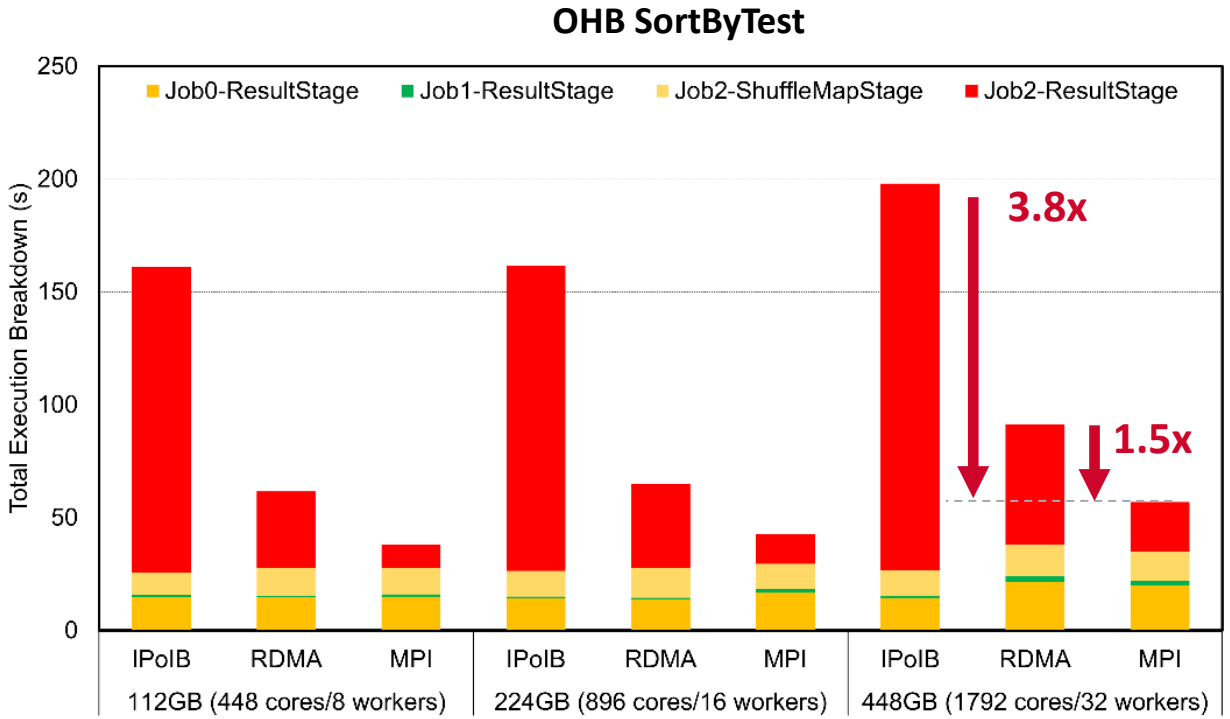
- MPI4Dask 0.3 was released in Feb ‘23 adding support for high-performance MPI communication to Dask:
  - Can be downloaded from: http://hibd.cse.ohio-state.edu
- Features:
  - (NEW) Based on Dask Distributed 2022.8.1
  - Compliant with user-level Dask APIs and packages
  - Support for MPI-based communication in Dask for cluster of GPUs
    - Implements point-to-point communication co-routines
    - Efficient chunking mechanism implemented for large messages
  - Built on top of mpi4py over the MVAPICH2-GDR library
  - Supports starting execution of Dask programs using Dask-MPI
  - Tested with
    - Mellanox InfiniBand adapters (FDR, EDR, and HDR)
    - (NEW) Various benchmarks used by the community (MatMul, Slicing, Sum Transpose, cuDF Merge, etc.)
    - (NEW) Various multi-core platforms
    - (NEW) NVIDIA V100 and A100 GPUs

Presentation Outline

- Introduction to Big Data Analytics
- Overview, Design and Implementation
  - MPI4Spark
  - MPI4Dask
- **Performance Evaluation**
  - **MPI4Spark**
  - **MPI4Dask**
- Demo – Hands-on Exercises with MPI4Dask
- Related Publications and Summary

Weak Scaling Evaluation with OSU HiBD Benchmarks (OHB)

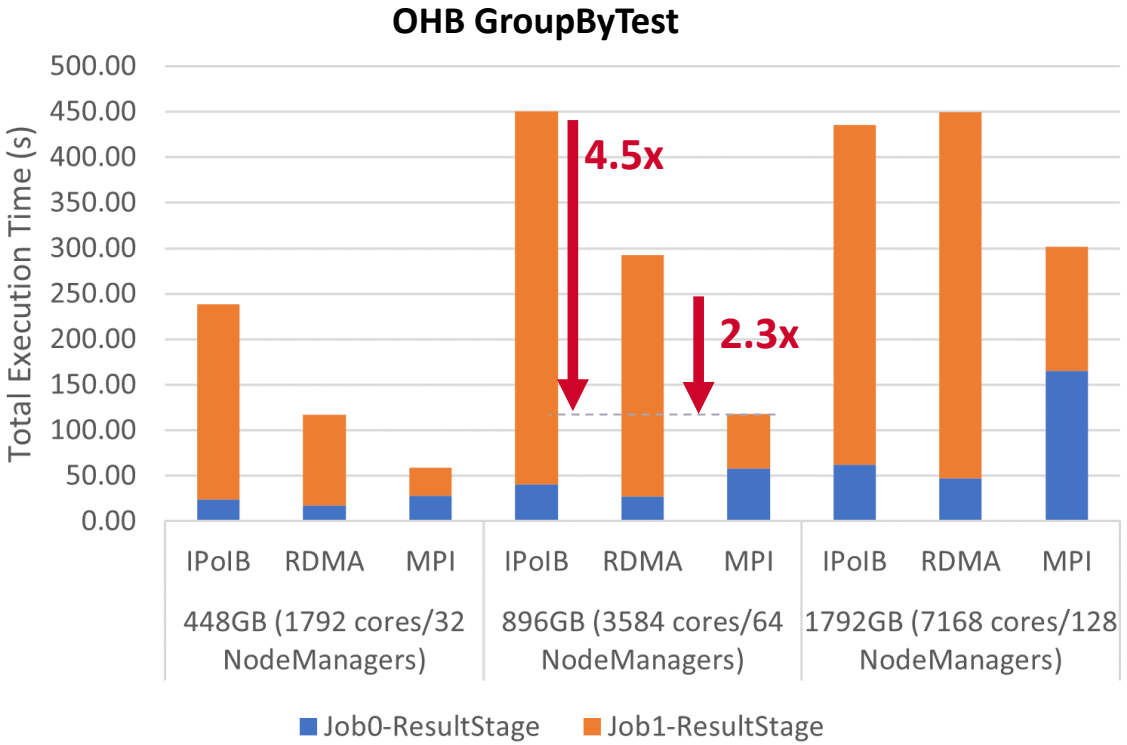


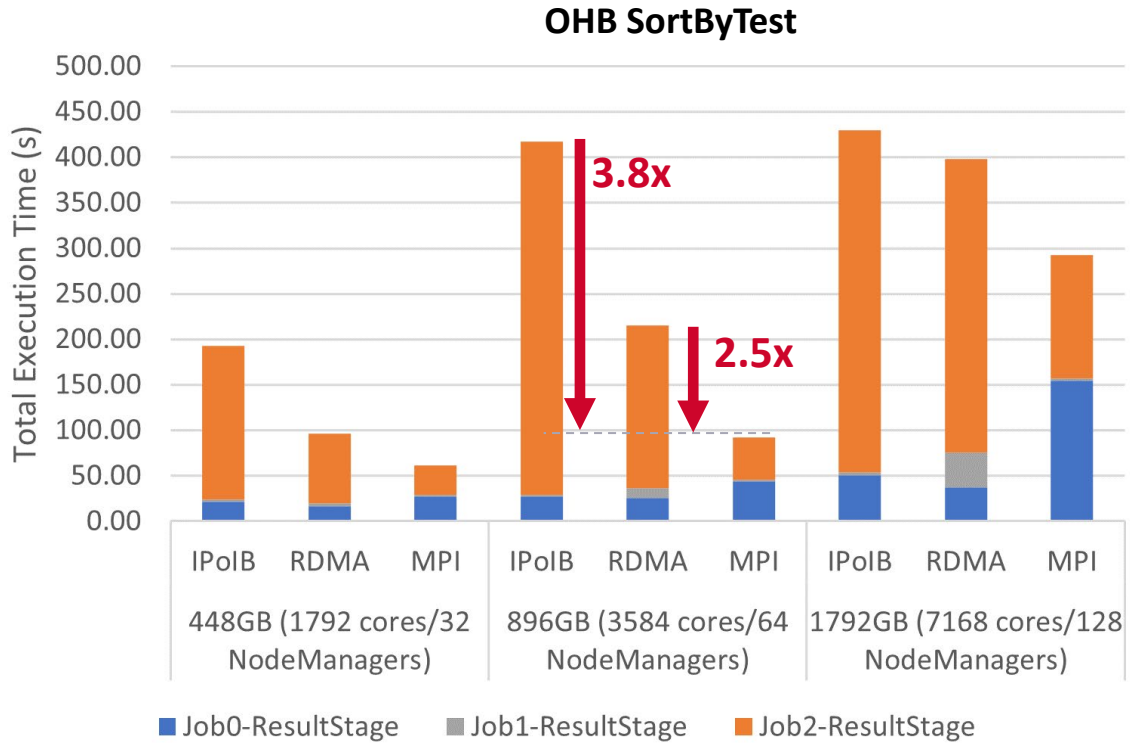


- The above are **weak-scaling** performance numbers of OHB benchmarks (GroupByTest and SortByTest) executed on the TACC Frontera system using the **Standalone cluster manager** in Spark
- Speed-ups for the overall total execution time for 448GB with GroupByTest is **4.1x** and **2.2x** compared to IPoIB and RDMA, and for SortByTest the speed-ups are **3.8x** and **1.5x**, respectively
- Speed-ups for the shuffle read stage for 112GB with GroupByTest are **13x** compared with IPoIB and **5.6x** compared to RDMA, while for SortByTest the speed-ups are **12.8x** and **3.2x**, respectively

K. Al Attar, A. Shafi, M. Abduljabbar, H. Subramoni, D. Panda, Spark Meets MPI: Towards High-Performance Communication Framework for Spark using MPI, IEEE Cluster '22, Sep 2022.

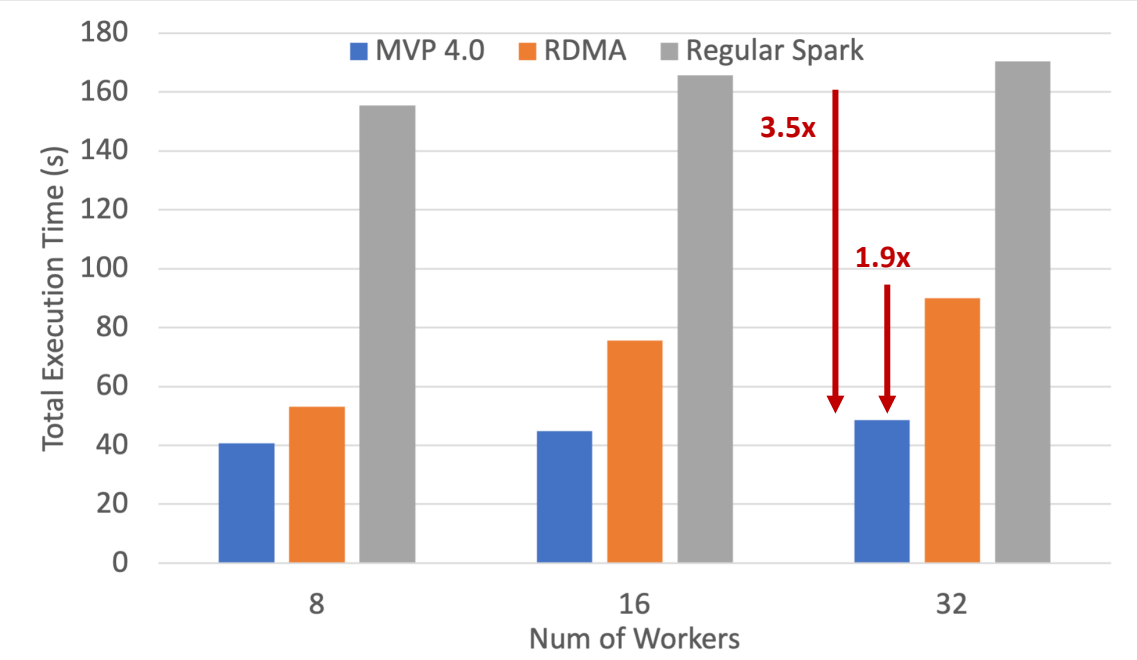
Weak Scaling Evaluation with OHB (YARN)



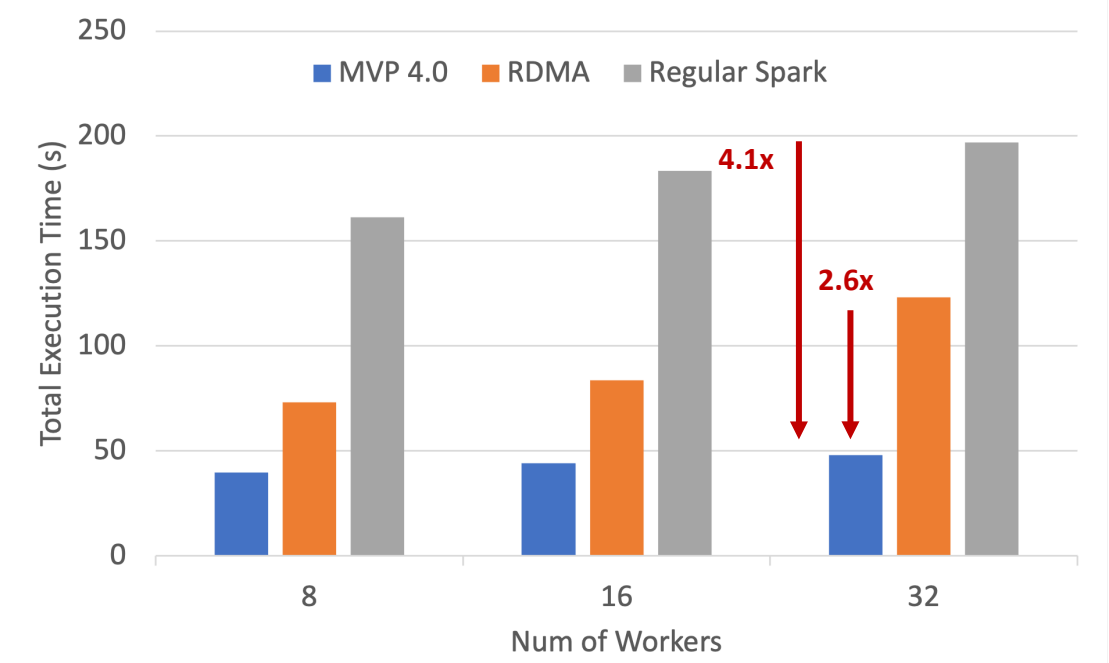


- The above are **weak-scaling** performance numbers of OHB benchmarks (GroupByTest and SortByTest) executed on the TACC Frontera system using the **YARN cluster manager** in Spark
- Speed-ups for the overall total execution time for SortByTest, 64 NodeManagers, are **4.5x** and **2.3x** compared to IPoIB and RDMA, and for GroupByTest, also 64 NodeManagers, the speed-ups are **3.8x** and **2.5x**, respectively
- Speed-ups for the shuffle read stage for 896GB with GroupByTest are **6.8x** compared with IPoIB and **4.4x** compared to RDMA, while for SortByTest the speed-ups are **8.4x** and **3.9x**, respectively

Performance Evaluation with MPI4Spark + MVP 4.0



OHB-Sortby

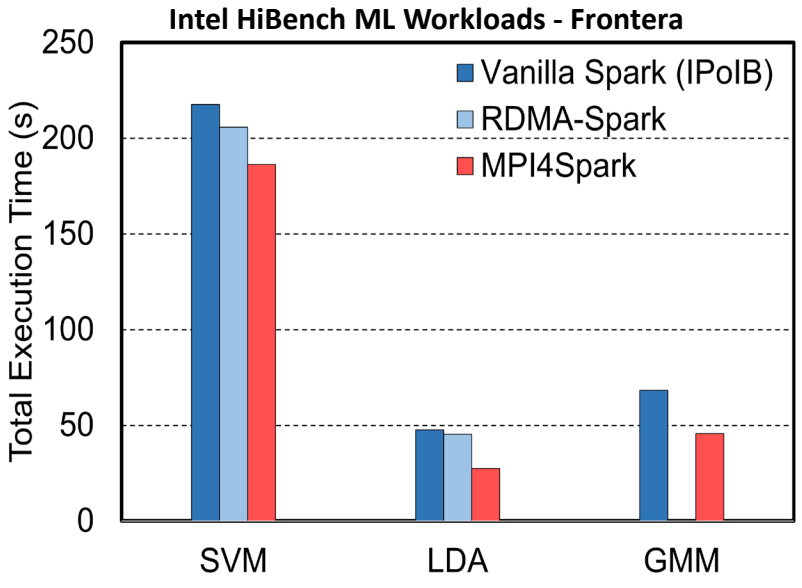


OHB-Groupby

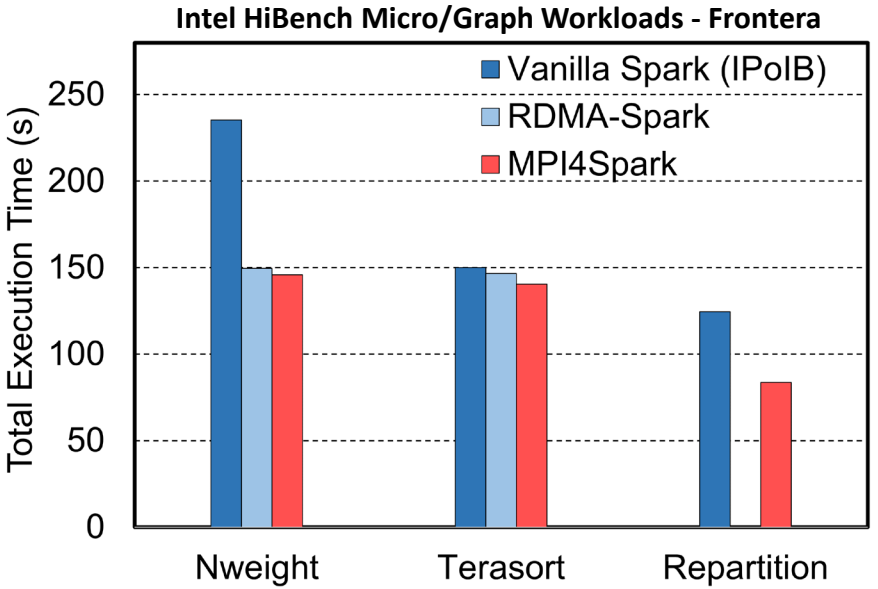
- The following are weak-scaling performance numbers of OHB benchmarks (GroupByTest and SortByTest) executed on the TACC Frontera system using MVAPICH version 4.0
- Speed-ups for the overall total execution time for 32 workers with GroupByTest is 4.1x and 2.6x compared to (regular) Spark and RDMA Spark, and for SortByTest the speed-ups are 3.5 and 1.9x, respectively.

Performance Evaluation with Intel HiBench Workloads

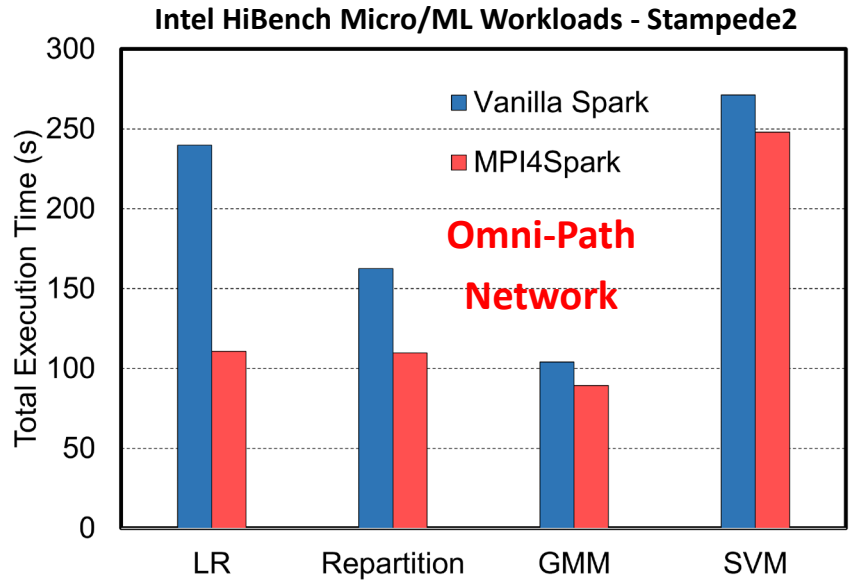
1.4x on average than RDMA-Spark



1.4x on average than Vanilla Spark



1.5x on average than Vanilla Spark



- This evaluation was done on the TACC Frontera (IB) and the TACC Stampede2 (OPA) Systems
- This illustrates the portability of MPI4Spark on different interconnects
- We see a speed-up for the LR machine learning workload on Stampede2 of about **2.2x**
- Speed-ups for the LDA machine learning workload on Frontera are **1.7x** for both IPoIB and RDMA

K. Al Attar, A. Shafi, M. Abduljabbar, H. Subramoni, D. Panda, Spark Meets MPI: Towards High-Performance Communication Framework for Spark using MPI, IEEE Cluster '22, Sep 2022.

Presentation Outline

- Introduction to Big Data Analytics
- Overview, Design and Implementation
  - MPI4Spark
  - MPI4Dask
- **Performance Evaluation**
  - **MPI4Spark**
  - **MPI4Dask**
- Demo – Hands-on Exercises with MPI4Dask
- Related Publications and Summary

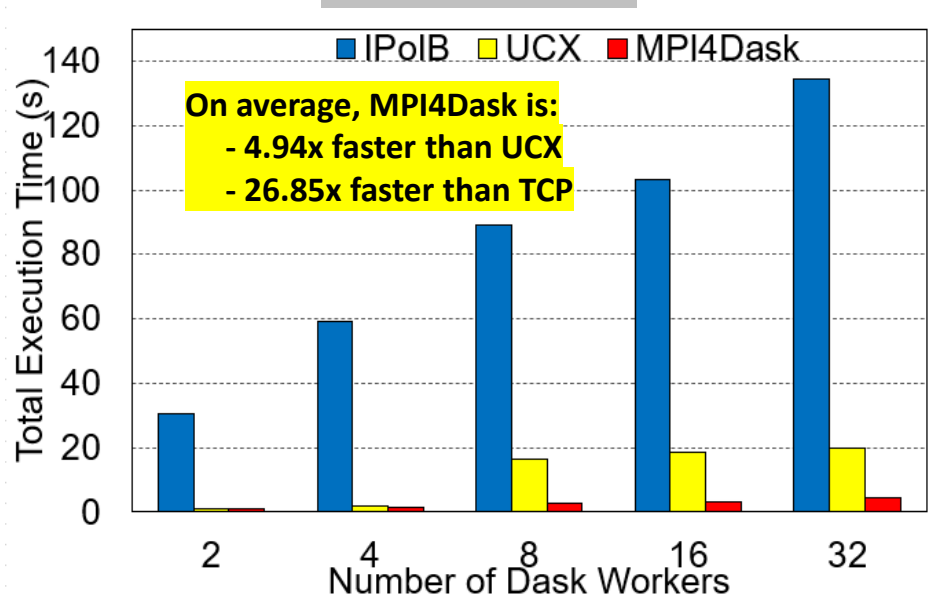
cuDF Merge Benchmark on the Cambridge Wilkes-3 System

- GPU-based Operation: $ddf1.merge(ddf2)$, using persist
  - Merge two GPU data frames, each with length of 32*1e8
  - Compute() will gather the data from all worker nodes to the client node, and make a copy on the host memory.
  - Persist() will leave the data on its current nodes without any gathering

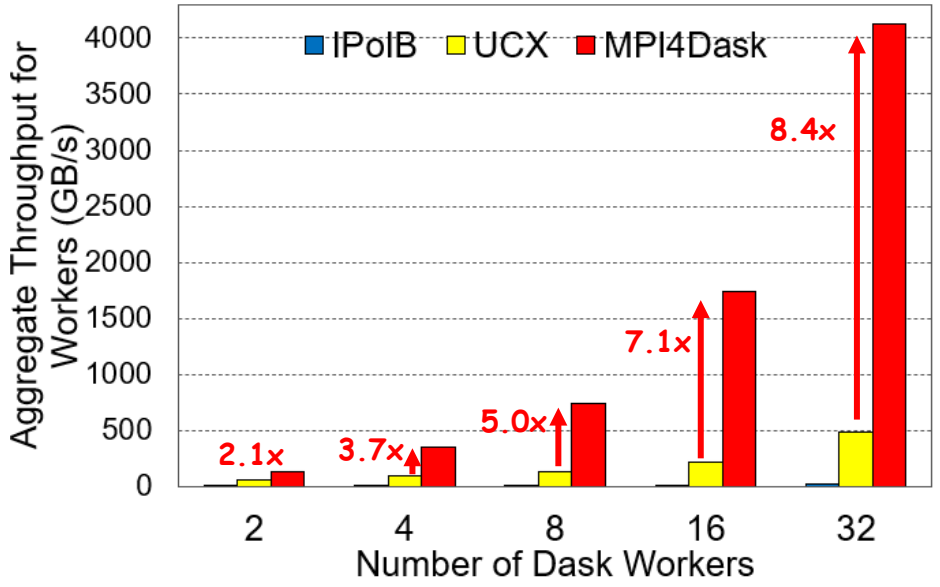
Wilke3 GPU System:

- 80 nodes
- 2x AMD EPYC 7763 64-core Processors
- 1000 GiB RAM
- Dual-rail Mellanox HDR200 IB
- 4x NVIDIA A100 SXM4 80 GB

Execution Time



Aggregated Throughput



MPI4Dask 0.3, Dask 2022.8.1, Distributed, 2022.8.1, MVAPICH2-3.0, UCX v1.13.1, UCX-py 0.27.00

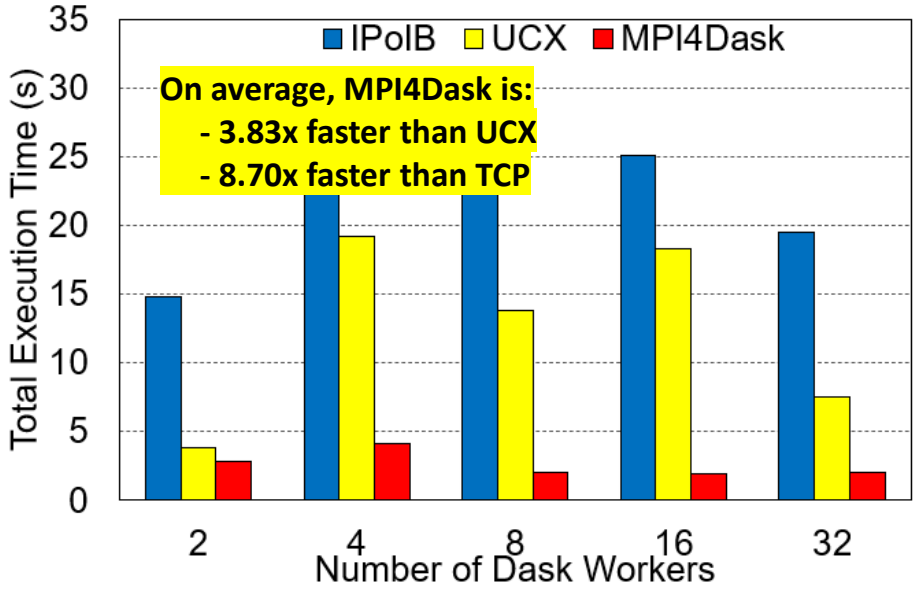
cupy GEMM Benchmark on the Cambridge Wilkes-3 System

- GPU-based Operation: $x.dot(y)$, using persist
  - Arrays are distributed on multiple GPUs
  - Compute() will gather the data from all worker nodes to the client node, and make a copy on the host memory.
  - Persist() will leave the data on its current nodes without any gathering

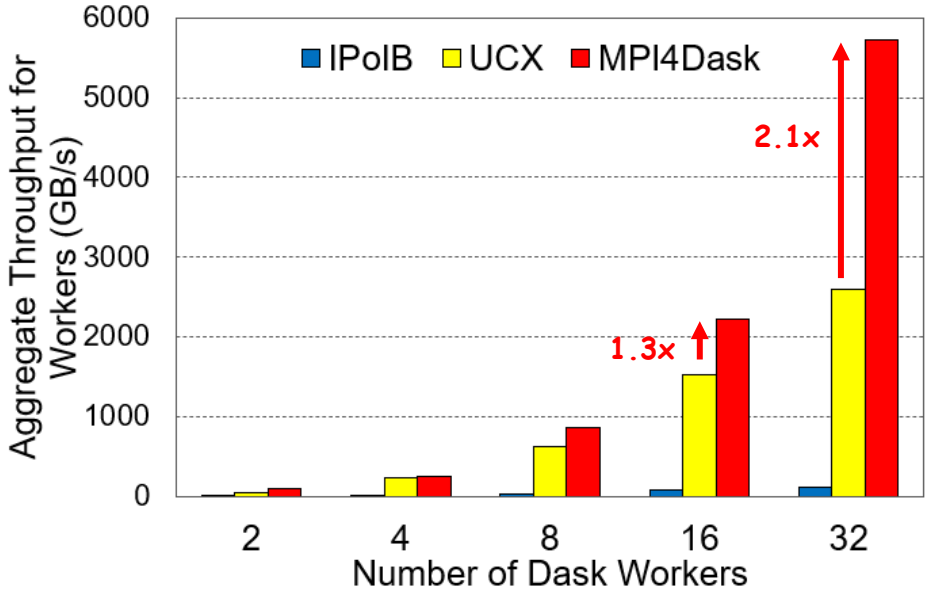
Wilke3 GPU System:

- 80 nodes
- 2x AMD EPYC 7763 64-core Processors
- 1000 GiB RAM
- Dual-rail Mellanox HDR200 IB
- 4x NVIDIA A100 SXM4 80 GB

Execution Time



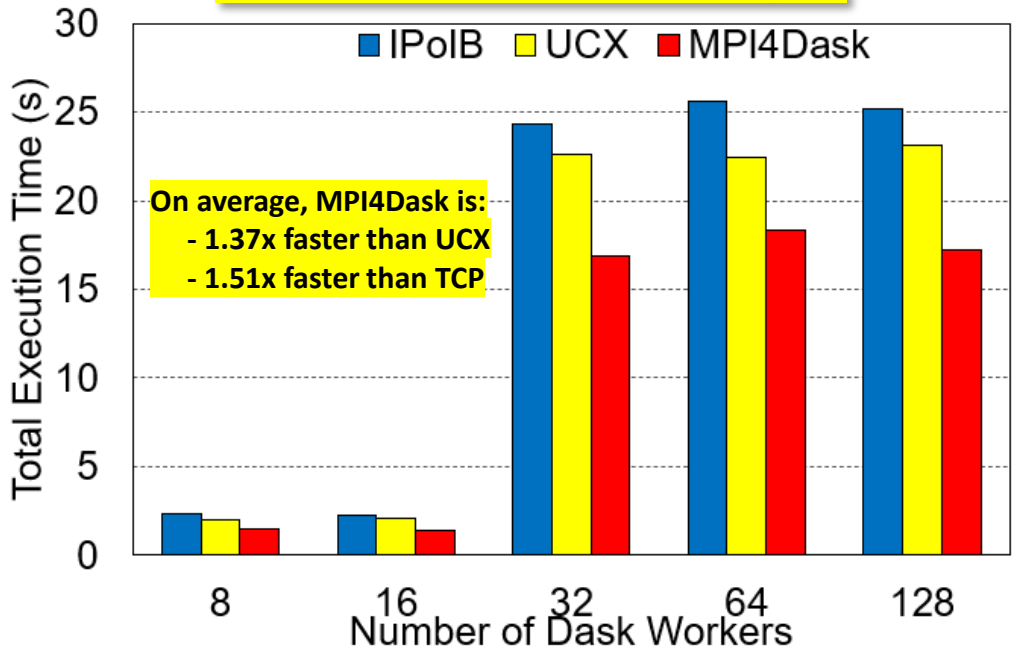
Aggregated Throughput



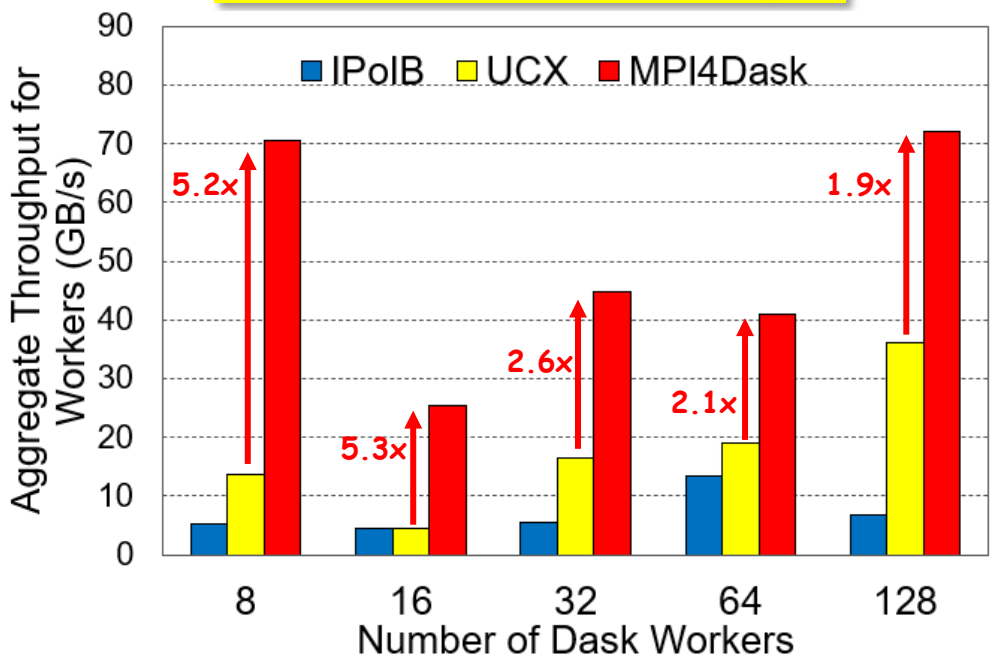
MPI4Dask 0.3, Dask 2022.8.1, Distributed, 2022.8.1, MVAPICH2-3.0, UCX v1.13.1, UCX-py 0.27.00

NumPy Array Slicing Benchmark on TACC Frontera CPU System

1.26x better on average



3.17x better on average



From 32 workers, we increase array size by 16 times

A. Shafi , J. Hashmi , H. Subramoni , and D. K. Panda, Efficient MPI-based Communication for GPU-Accelerated Dask Applications, CCGrid '21
https://arxiv.org/abs/2101.08878

MPI4Dask 0.3 release

(http://hibd.cse.ohio-state.edu)

Presentation Outline

- Introduction to Big Data Analytics
- Overview, Design and Implementation
  - MPI4Spark
  - MPI4Dask
- Performance Evaluation
  - MPI4Spark
  - MPI4Dask
- **Demo – Hands-on Exercises with MPI4Dask**
- Related Publications and Summary

Lab 2 – Hands-on Lab with MPI4Dask

- Objectives
  - How to run parallel and distributed data science applications using Dask on HPC systems
  - How to use multi-node GPUs for Dask-based applications

- Tasks
  - Task 1: Sum of CuPy Array and its Transpose
  - Task 2: Cupy Matrix Multiplication
  - Task 3: Cupy Array Clicing

Task 1a: Sum of CuPy Array and its Transpose (GPU-based)

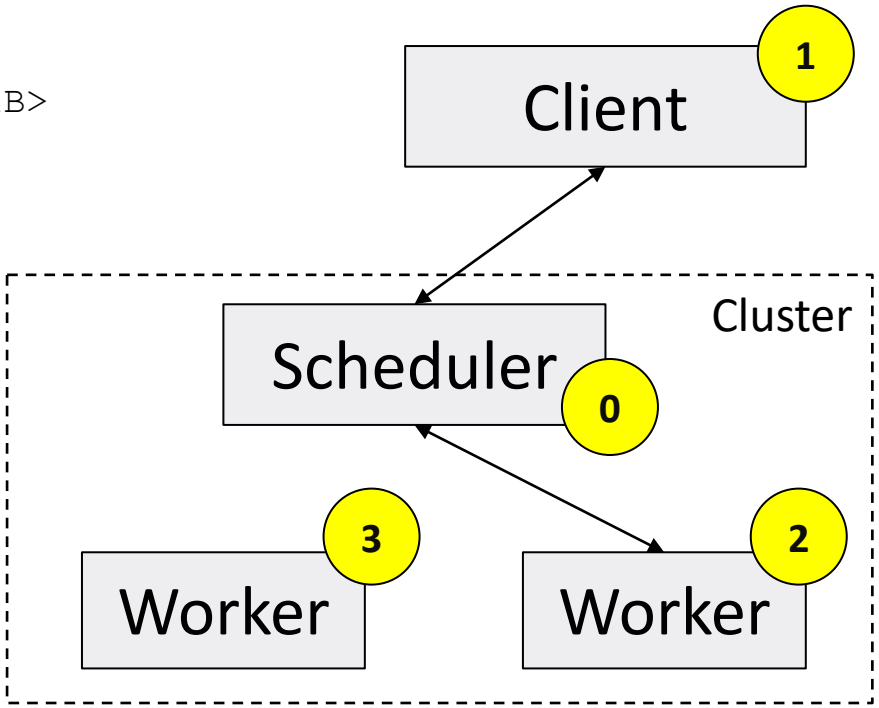
- Run the benchmark with a TCP communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task1
$ sh run_task1.sh tcp
```

- Expected output:

```
<Client: 'tcp://10.3.1.2:44230' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 3.9933362007141113
Time for iteration 1 : 1.7020411491394043
Time for iteration 2 : 1.6842925548553467
Time for iteration 3 : 1.6863949298858643
Time for iteration 4 : 1.577439546585083
Time for iteration 5 : 1.6131360530853271

Median Time: 1.68s
```



Task 1b: Sum of CuPy Array and its Transpose (GPU-based)

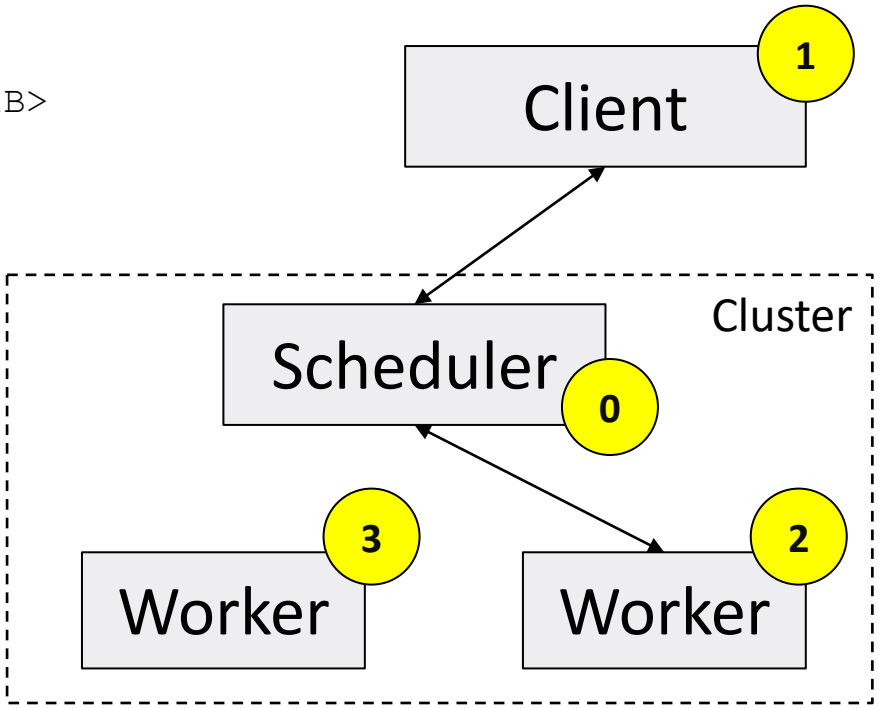
- Run the benchmark with a UCX communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task1
$ sh run_task1.sh ucx
```

- Expected output:

```
<Client: 'ucx://10.3.1.2:37564' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 2.47611141204834
Time for iteration 1 : 1.1098558902740479
Time for iteration 2 : 1.288067102432251
Time for iteration 3 : 1.0797405242919922
Time for iteration 4 : 1.0817945003509521
Time for iteration 5 : 1.069718360900879

Median Time: 1.09s
```



Task 1c: Sum of CuPy Array and its Transpose (GPU-based)

- Run the benchmark again with the new MPI communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task1
$ sh run_task1.sh mpi
```

- Expected output:

```
<Client: 'mpi://10.3.1.2:36198' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 3.2865982055664062
Time for iteration 1 : 0.3287394046783447
Time for iteration 2 : 0.36843132972717285
Time for iteration 3 : 0.3794410228729248
Time for iteration 4 : 0.3915135860443115
Time for iteration 5 : 0.3674592971801758
```

Median Time: 0.38s

MVAPICH2: 4.4x faster than TCP
2.8x faster than UCX

Task 2a: Cupy Matrix Multiplication (GPU-based)

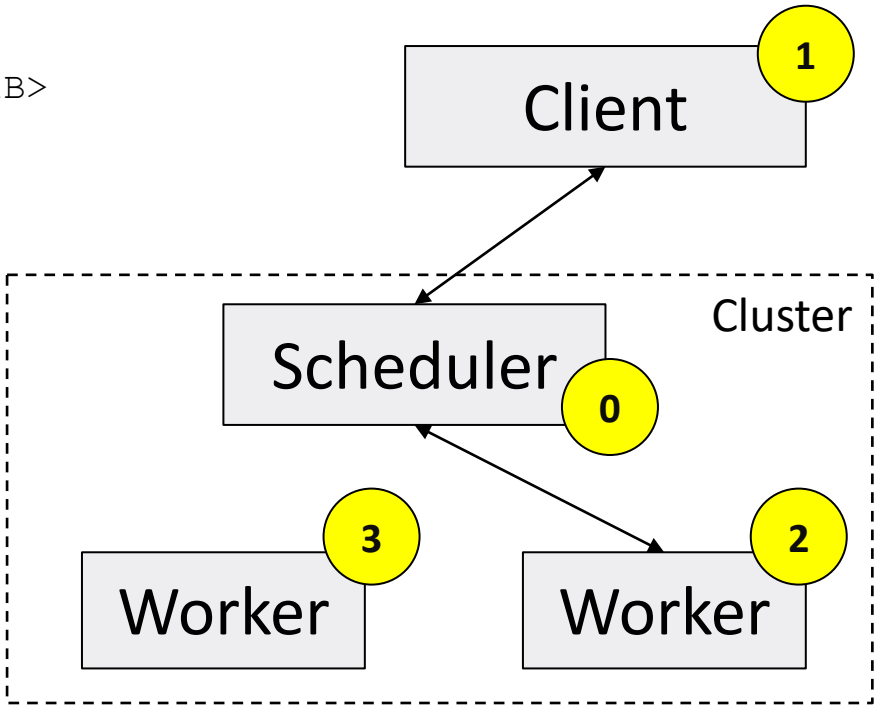
- Run the benchmark with a TCP communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task2
$ sh run_task2.sh tcp
```

- Expected output:

```
<Client: 'tcp://10.3.1.6:33132' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 5.673777103424072
Time for iteration 1 : 3.202324867248535
Time for iteration 2 : 3.323018789291382
Time for iteration 3 : 3.1695098876953125
Time for iteration 4 : 3.1934258937835693
Time for iteration 5 : 3.257124423980713

Median Time: 3.22s
```



Task 2b: Cupy Matrix Multiplication (GPU-based)

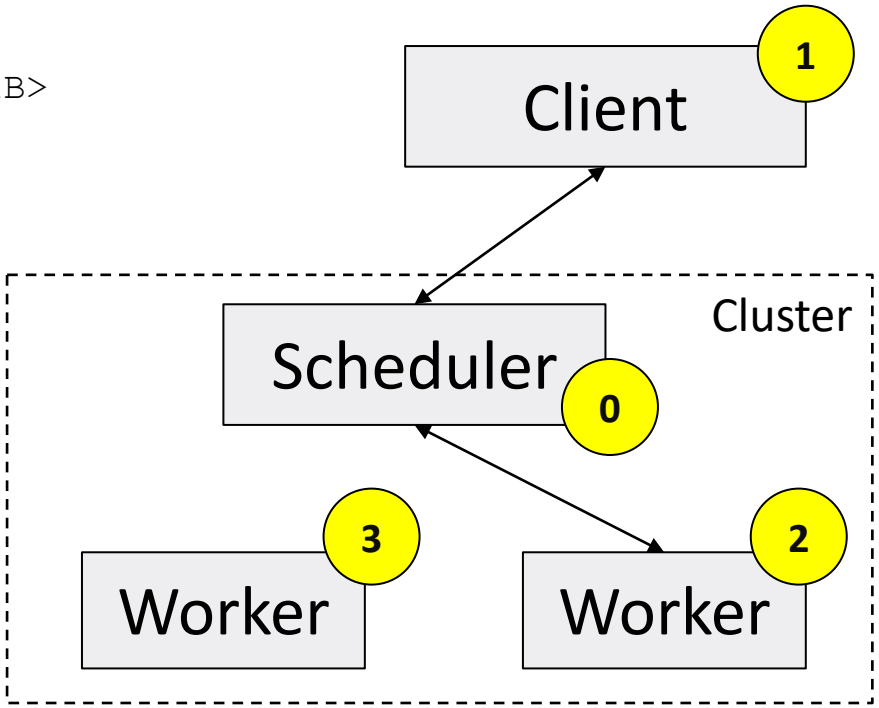
- Run the benchmark with a UCX communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task2
$ sh run_task2.sh ucx
```

- Expected output:

```
<Client: 'ucx://10.3.1.2:55172' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 2.83543062210083
Time for iteration 1 : 2.19091534614563
Time for iteration 2 : 2.189948558807373
Time for iteration 3 : 2.125943660736084
Time for iteration 4 : 2.200505495071411
Time for iteration 5 : 2.326890230178833

Median Time: 2.19s
```



Task 2c: Cupy Matrix Multiplication (GPU-based)

- Run the benchmark again with the new MPI communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task2
$ sh run_task2.sh mpi
```

- Expected output:

```
<Client: 'mpi://10.3.1.6:34910' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 2.369664192199707
Time for iteration 1 : 1.2211949825286865
Time for iteration 2 : 1.2420144081115723
Time for iteration 3 : 1.2281405925750732
Time for iteration 4 : 1.2588093280792236
Time for iteration 5 : 1.2160212993621826
```

Median Time: 1.25s

MVAPICH2: 2.5x faster than TCP
1.7x faster than UCX

Task 3a: Cupy Array Slicing (GPU-based)

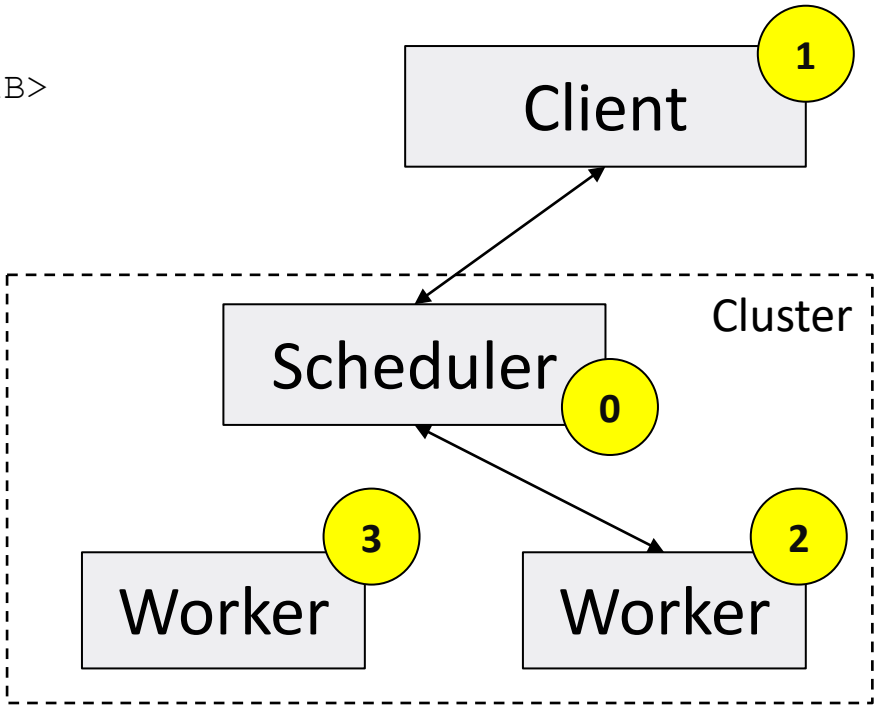
- Run the benchmark with a TCP communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task3
$ sh run_task3.sh tcp
```

- Expected output:

```
<Client: 'tcp://10.3.1.6:40202' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 3.7195968627929688
Time for iteration 1 : 1.3150527477264404
Time for iteration 2 : 1.2060997486114502
Time for iteration 3 : 1.2438180446624756
Time for iteration 4 : 1.2373754978179932
Time for iteration 5 : 1.164992332458496

Median Time: 1.24s
```



Task 3b: Cupy Array Slicing (GPU-based)

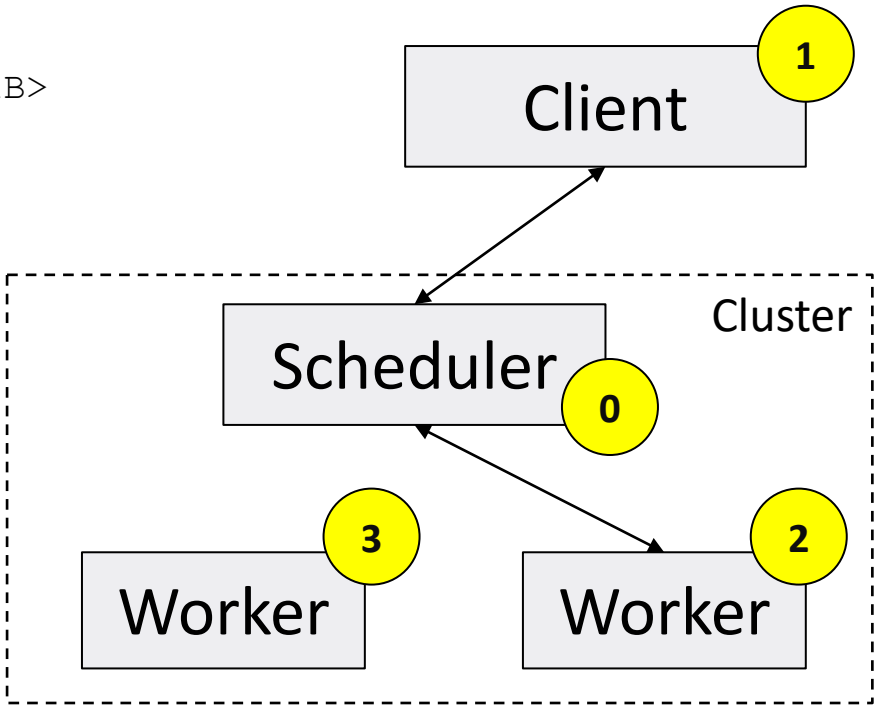
- Run the benchmark with a UCX communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task3
$ sh run_task3.sh ucx
```

- Expected output:

```
<Client: 'ucx://10.3.1.2:56148' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 2.9743316173553467
Time for iteration 1 : 0.9042410850524902
Time for iteration 2 : 0.8928432464599609
Time for iteration 3 : 0.8946189880371094
Time for iteration 4 : 0.8854148387908936
Time for iteration 5 : 0.8948419094085693

Median Time: 0.89s
```



Task 3c: Cupy Array Slicing (GPU-based)

- Run the benchmark again with the new MPI communicator

```
$ salloc --nodes=4 --time=3:00 --reservation=hibd-tutorial
$ cd /opt/tutorials/hibd/mpi4dask-usrs/$USER/labs/task3
$ sh run_task3.sh mpi
```

- Expected output:

```
<Client: 'mpi://10.3.1.6:30125' processes=2 threads=16, memory=143.58 GiB>
Time for iteration 0 : 3.952059268951416
Time for iteration 1 : 0.39922380447387695
Time for iteration 2 : 1.061549425125122
Time for iteration 3 : 0.3944559097290039
Time for iteration 4 : 0.3925657272338867
Time for iteration 5 : 0.41716957092285156
```

Median Time: 0.40s

MVAPICH2: 3.1x faster than TCP
2.2x faster than UCX

Presentation Outline

- Introduction to Big Data Analytics
- Overview, Design and Implementation
  - MPI4Spark
  - MPI4Dask
- Performance Evaluation
  - MPI4Spark
  - MPI4Dask
- Demo – Hands-on Exercises with MPI4Dask
- **Related Publications and Summary**

Related Publications

- Spark Meets MPI: Towards High-Performance Communication Framework for Spark using MPI K. Al Attar, A. Shafi, M. Abduljabbar, H. Subramoni, D. Panda IEEE Cluster '22, Sep 2022.

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

- Efficient MPI-based Communication for GPU-Accelerated Dask Applications A. Shafi, J. Hashmi, H. Subramoni, D. Panda, The 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, May 2021. https://arxiv.org/abs/2101.08878

- Blink: Towards Efficient RDMA-based Communication Coroutines for Parallel Python Applications A. Shafi, J. Hashmi, H. Subramoni, D. Panda, 27th IEEE International Conference on High Performance Computing, Data, and Analytics, Dec 2020.

Summary

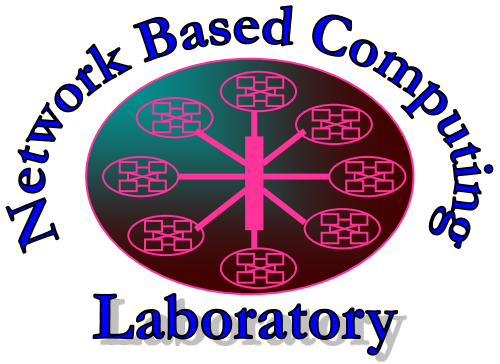
- Apache Spark and Dask are two popular Big Data processing frameworks
- There is existing support for parallel and distributed on HPC systems:
  - One bottleneck is the lack of support for low-latency and high-bandwidth interconnects
- This talk presented latest developments in the MPI4Dask (MPI-based Dask ecosystem) and MPI4Spark (MPI-based Spark ecosystem)
- Provided an overview of issues, challenges, and opportunities for designing efficient communication runtimes
  - Efficient, scalable, and hierarchical designs are crucial for Big Data/Data Science frameworks
  - Co-design of communication runtimes and BigData/Data Science frameworks will be essential

Thank You!

{shafi.16}@osu.edu

Follow us on

https://x.com/mvapich



Network-Based Computing Laboratory

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

The MVAPICH Project

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

The High-Performance Deep Learning Project

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