MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

High-Performance
Big Data

High-Performance
Deep Learning

MPI4Spark: A High-Performance Communication Framework for Spark using MPI

Presented at MUG 2022

Kinan Al-Attar and Aamir Shafi

Network-Based Computing Laboratory

Dept. of Computer Science and Engineering , The Ohio State University

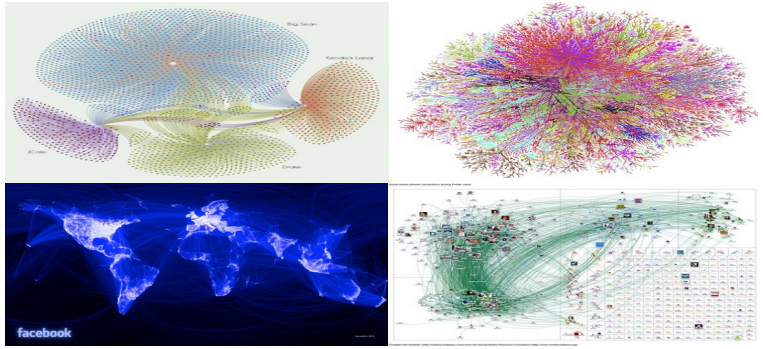
alattar.2@osu.edu, shafi.16@osu.edu

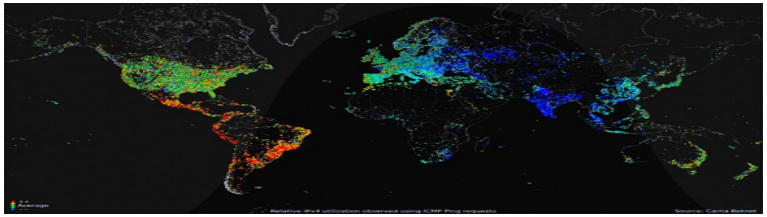
Presentation Outline

- **Background**
  - **Introduction to Spark and Netty**
- Motivation and Challenges
- MPI4Spark Design and Implementation
- MPI4Spark Performance Evaluation
- Conclusions and Future Work

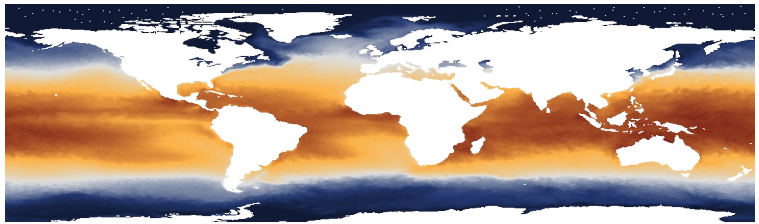
Introduction to Big Data Analytics and Trends

- 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
- Running High Performance Data Analysis (HPDA) workloads in the cloud is gaining popularity
  - According to the latest OpenStack survey, 27% of cloud deployments are running HPDA workloads
- Sometimes also called Data Science





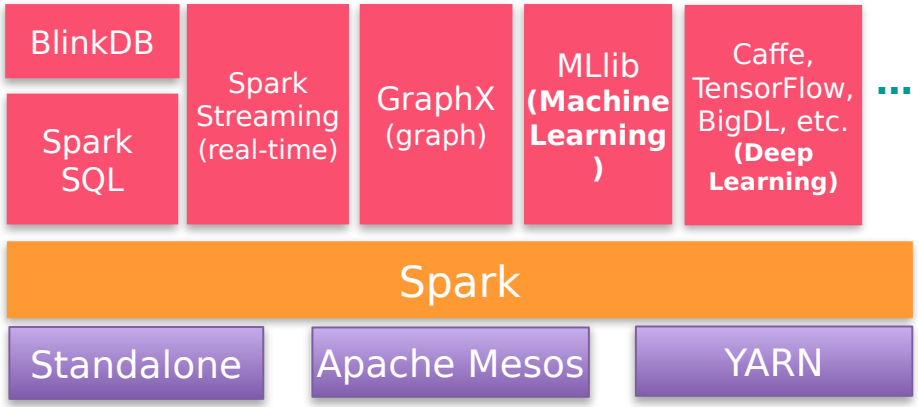
http://www.coolinfographics.com/blog/tag/data?currentPage=3



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

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

RDD Programming Model in Spark

- Key idea: *Resilient Distributed Datasets* (***RDDs***)
  - Immutable distributed collections of objects that can be cached in memory across cluster nodes
  - Created by transforming data in stable storage using data flow operators (map, filter, groupBy, …)
  - Manipulated through various parallel operators
  - Automatically rebuilt on failure
    - rebuilt if a partition is lost
- Interface
  - Clean language-integrated API in Scala (Python & Java)
  - Can be used *interactively* from Scala and PySpark console

RDD Operations

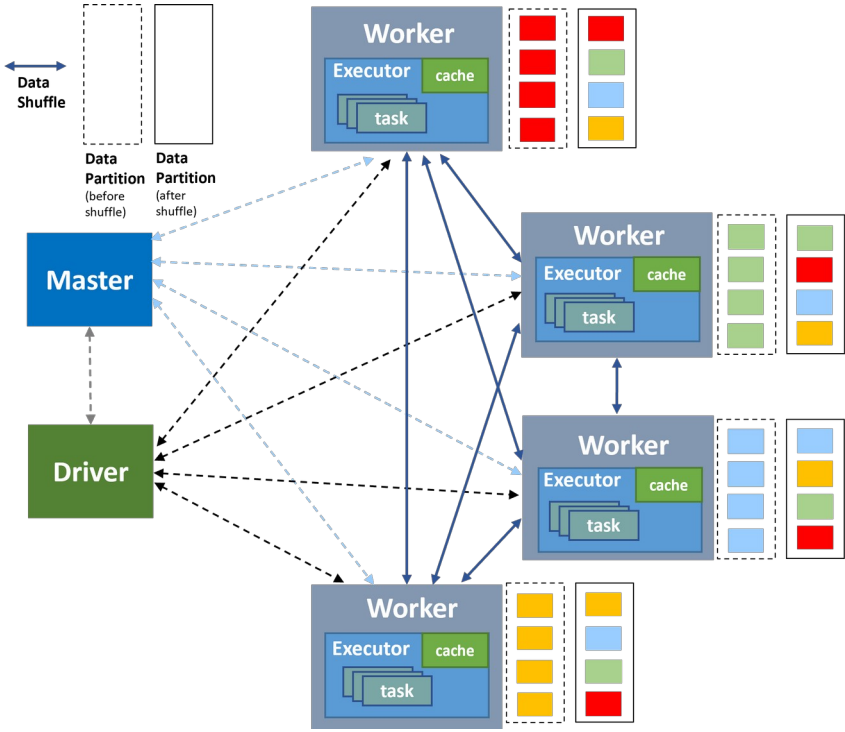
Transformations (define a new RDD)	Actions (return a result to driver)
map	reduce
filter	collect
sample	count
union	first
groupByKey	Take
reduceByKey	countByKey
sortByKey	saveAsTextFile
join	saveAsSequenceFile
…	…

More Information:

- https://spark.apache.org/docs/latest/programming-guide.html#transformations
- https://spark.apache.org/docs/latest/programming-guide.html#actions

Apache Spark: Support for Parallel Execution

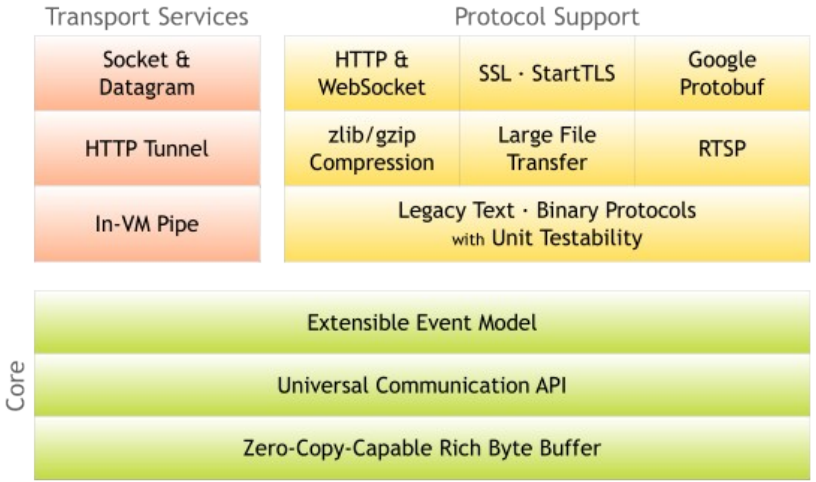
- Parallel and distributed computing:
  - Master, workers, and executors manage distributed execution of user applications
- Master communicates with other processes to allocate resources and launch executors on worker nodes
- The figure illustrates the communication patterns of the shuffle phase for a Spark cluster with four worker nodes
- The shuffle phase involves data shuffling across the network and is a performance bottleneck for Spark applications



http://spark.apache.org

Netty: Spark's Communication Backend

- Netty is an asynchronous event-driven network application framework
  - It uses Java New I/O (NIO) transport by default
- The NIO transport relies on a selector that utilizes the event notification API, to indicate which, among a set of non-blocking sockets, are ready for I/O
- Spark uses Netty to communicate RPC and shuffle messages
  - It does this through a set of message types that are divided into request and response message types



http://netty.io/

Presentation Outline

- Background
  - Introduction to Spark and Netty
- **Motivation and Challenges**
- MPI4Spark Design and Implementation
- MPI4Spark Performance Evaluation
- Conclusions and Future Work

Motivation

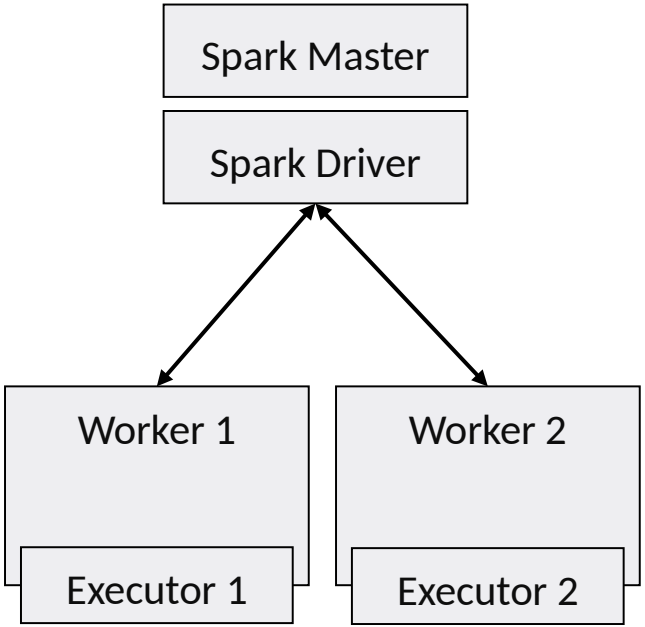
- The overall communication performance of the shuffle phase becomes a significant bottleneck in distributed execution of Big Data workloads as a result of relying on TCP/IP for communication
  - Spark framework fails to exploit high-performance and low latency interconnects provided by HPC systems
- The primary motivation for MPI4Spark is to utilize the communication functionality provided by production-quality MPI libraries in the Apache Spark framework without having to extend the high-level Spark API
- Existing approaches:
  - SparkUCX: a new ShuffleManager based on the UCX communication library
  - RDMA-Spark: a new BlockTransferService with existing ShuffleManagers

MPI4Dask vs. Prior Work

Features	MPI4Spark	RDMA-Spark	SparkUCX	Spark+MPI	Spark-MPI
Support for multiple interconnects	✓	✗	✓	✓	✓
Adheres to Spark API	✓	✓	✓	✗	✓
Studies with Existing Benchmark Suites	✓	✓	N/A	✓	N/A
Optimization Technique	MPI-Based Netty	RDMA-Based Block TransferService	UCX-Based Shuffle Manager	Offload to shared memory and use MPI	N/A

Challenges

- Launching Spark in an MPI environment
  - MPI follows Single Program Model Data (SPMD) model vs. manual launch (possibly aided by resource managers) of Spark processes
  - Spark dynamically launches processes (workers fork executors)
- Event-driven vs. Application-driven Communication Engines
  - Process naming (endpoints/channels/sockets vs. MPI ranks)



Presentation Outline

- Background
  - Introduction to Spark and Netty
- Motivation and Challenges
- **MPI4Spark Design and Implementation**
- MPI4Spark Performance Evaluation
- Conclusions and Future Work

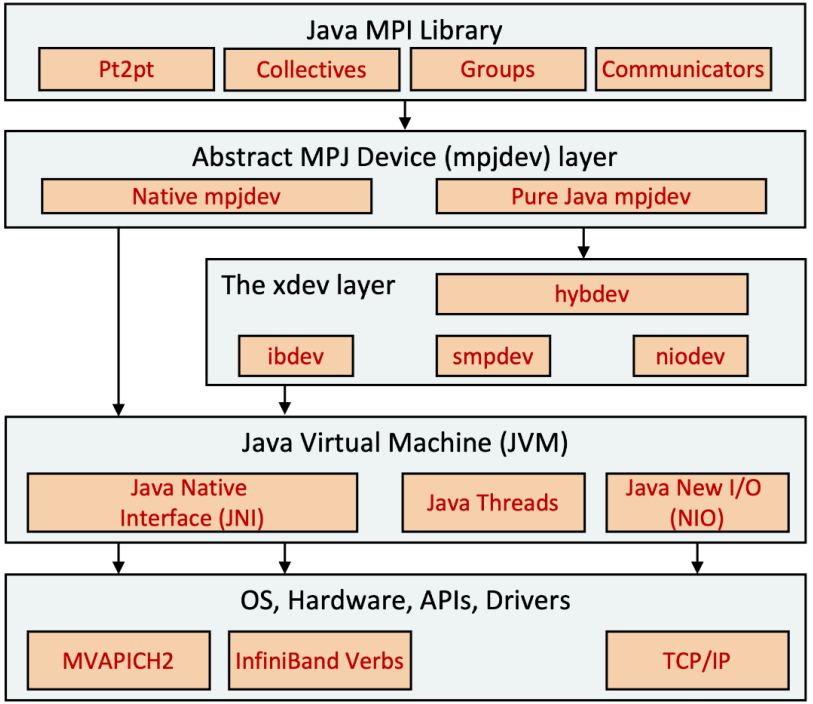
Overview of the MVAPICH2 Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), AWS EFA, Rockport Networks, and Slingshot10/11, Broadcom
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD)
- Started in 2001, first open-source version demonstrated at SC '02
- Supports the latest MPI-3.1 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA (since 2014) and AMD (since 2020) GPUs
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- Tools:
  - OSU MPI Micro-Benchmarks (OMB), since 2003
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- **Used by more than 3,275 organizations in 90 countries**
- **More than 1.61 Million downloads from the OSU site directly**
- Empowering many TOP500 clusters (June '22 ranking)
  - 6th, 10,649,600-core (Sunway TaihuLight) at NSC, China
  - 16th, 448, 448 cores (Frontera) at TACC
  - 30th, 288,288 cores (Lassen) at LLNL
  - 42nd, 570,020 cores (Nurion) in South Korea and many more
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 16th ranked TACC Frontera system
- **Empowering Top500 systems for more than 20 years**

MVAPICH2-J: Java Bindings to MVAPICH2

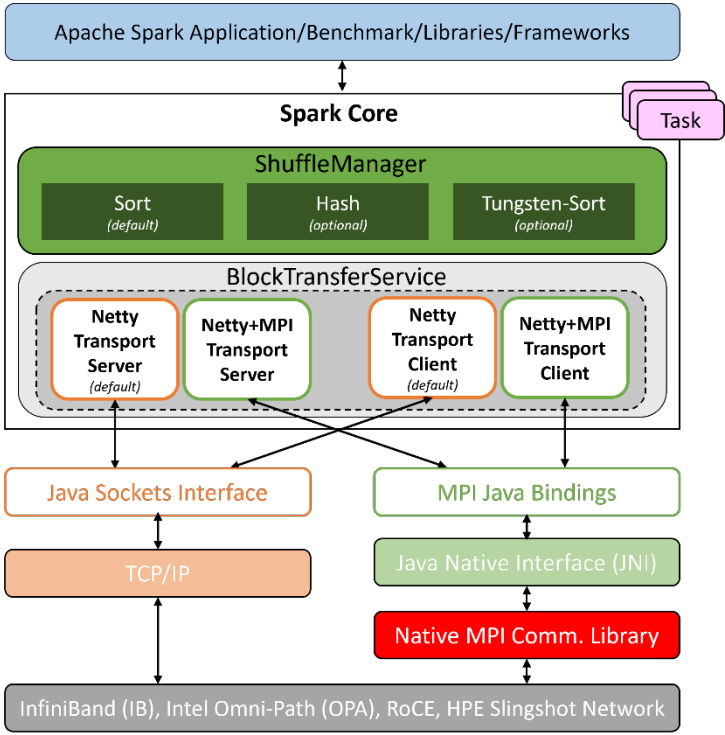
- We have recently added Java bindings to the MVAPICH2 library:
  - Allows writing HPC applications in the Java programming language
- The library currently implements a subset of the MPI API:
  - Our bindings follow the same API as Open MPI Java bindings
- MVAPICH2-J currently supports:
  - blocking/non-blocking point-to-point functions
  - blocking collective functions
  - blocking vectored collective functions
- 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/



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

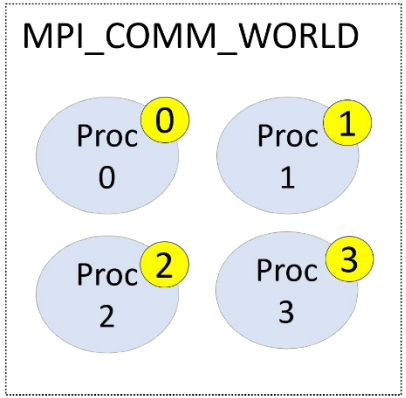
MPI4Spark: Using MVAPICH2 to Optimize Apache Spark

- The main goal 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 ShuffleManager relies on Netty for communication:
  - Netty is a Java New I/O (NIO) client/server framework for event-based networking applications
  - The key idea is to utilize MPI-based point-to-point communication inside Netty

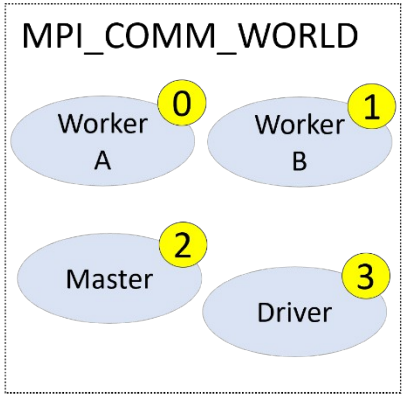


Launching Spark using MPI with Dynamic Process Management

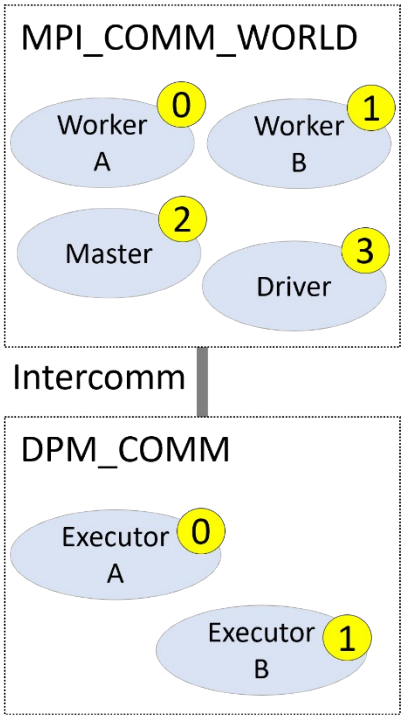
Step A: Launch 4 Wrapper Processes (for e.g. mpiexec –np 4 .. SparkMPI.java)

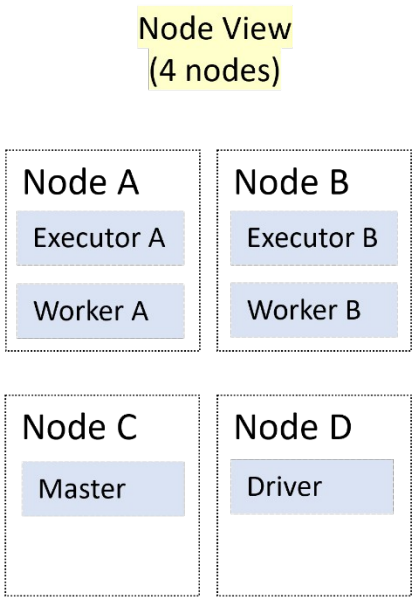


Step B: Each Wrapper Process Forks Spark Processes



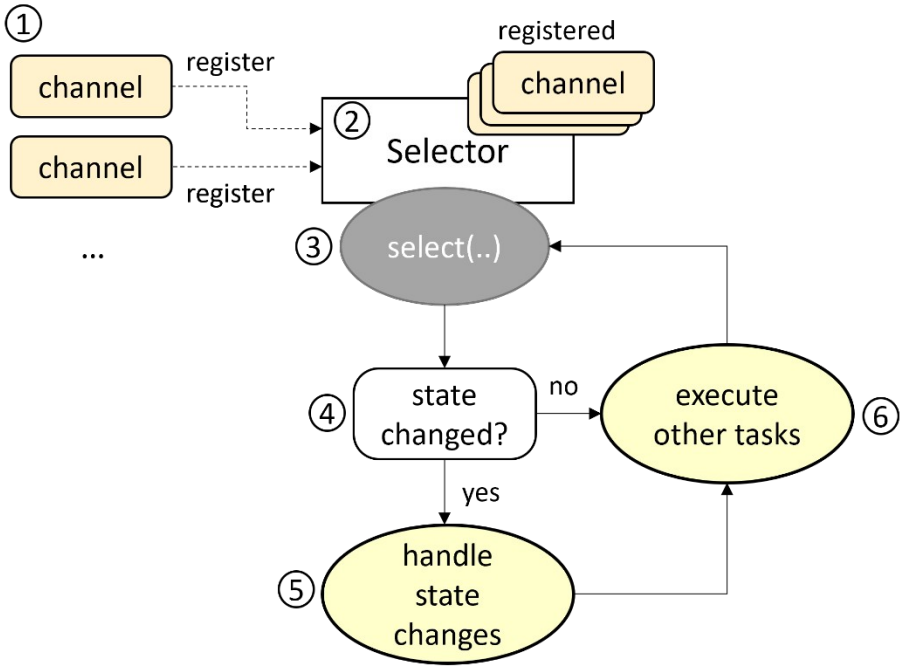
Step C: Launch 2 Executor Processes MPI_Comm_spawn_multiple()





MPI4Spark-Basic Design

- Modified the Netty NIO selector loop, which polls for channel state changes based on connection, read, or write events
- Inside of the selector loop checks were implemented with MPI non-blocking probing method (MPI_probe) for MPI_recv calls matching MPI_sends
- Netty Channels or simply Java sockets were still being used but only for connection establishment
- Too CPU-intensive, performed badly

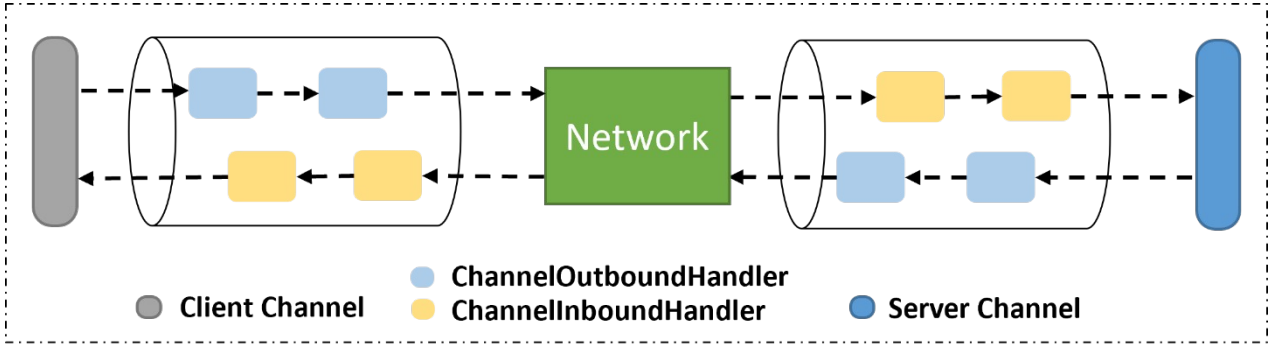


Types of Messages Communicated by Spark

Message Type	Function
StreamRequest	A request to stream data from the remote end
StreamResponse*	A response to a StreamRequest when the stream has been successfully opened
RpcRequest	A request to perform a generic Remote Procedure Call (RPC)
RpcResponse	A response to a RpcRequest for a successful RPC
ChunkFetchRequest	A request to fetch a sequence of a single chunk of a stream
ChunkFetchSuccess*	A response to ChunkFetchRequest when a chunk exists and has been successfully fetched
OneWayMessage	A RPC that does not expect a reply

MPI4Spark-Optimized Design

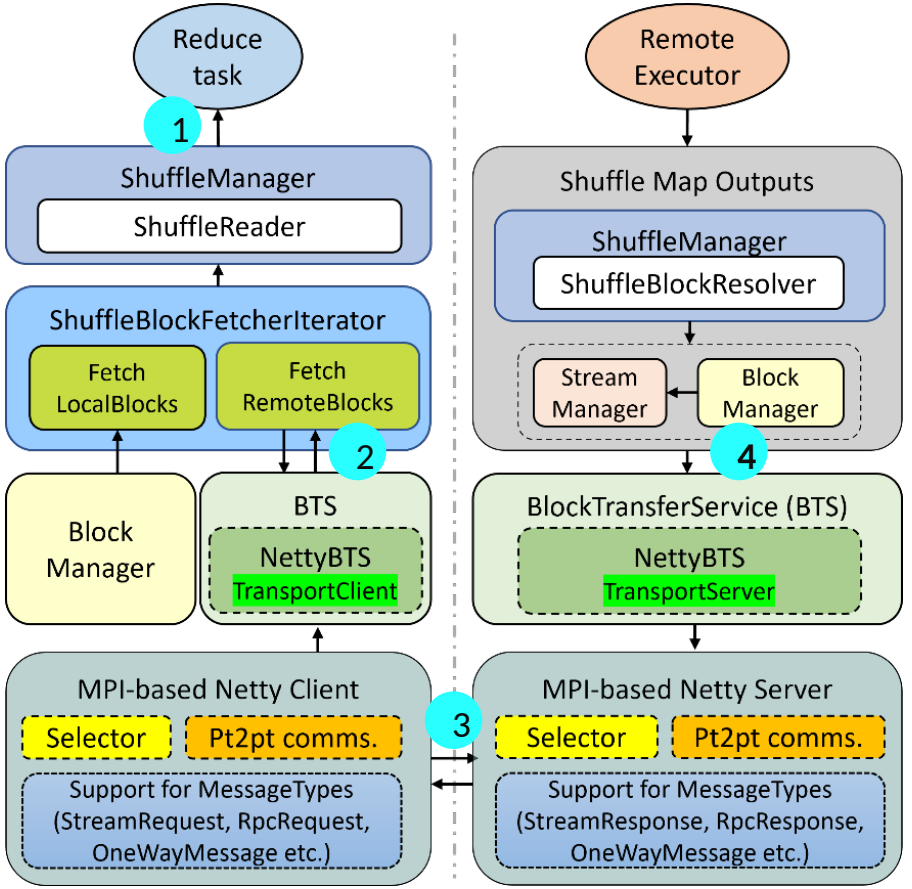
- The MPI4Spark-Optimized design avoids the pitfalls of the MPI4Spark-Basic design and is a lot simpler
- In this design, we only target shuffle messages, Knowing that the shuffle phase was a performance bottleneck and can account for 80% of total execution time
  - non-blocking MPI probes are avoided
  - the idea was now to trigger MPI_recv calls by parsing the headers of shuffle messages inside of ChannelHandlers that reside in ChannelPipelines in Netty



MPI4Spark: Optimizing the Communication (Shuffle) Phase

- Dataflow for two executors
  - One of the executors performs a reduce task that requires fetching of remote blocks

1. The reduce task starts with reading records inside of ShuffleReader
2. ShuffleBlockFetcherIterator is used to fetch data blocks locally or remotely
3. When remote fetches take place, the ShuffleBlockFetcherIterator will send requests to the underlying NettyBlockTransferService
4. MPI-based Netty will then be used to communicate the remote data block using the ShuffleBlockResolver



Presentation Outline

- Background
  - Introduction to Spark and Netty
- Motivation and Challenges
- MPI4Spark Design and Implementation
- **MPI4Spark Performance Evaluation**
- Conclusions and Future Work

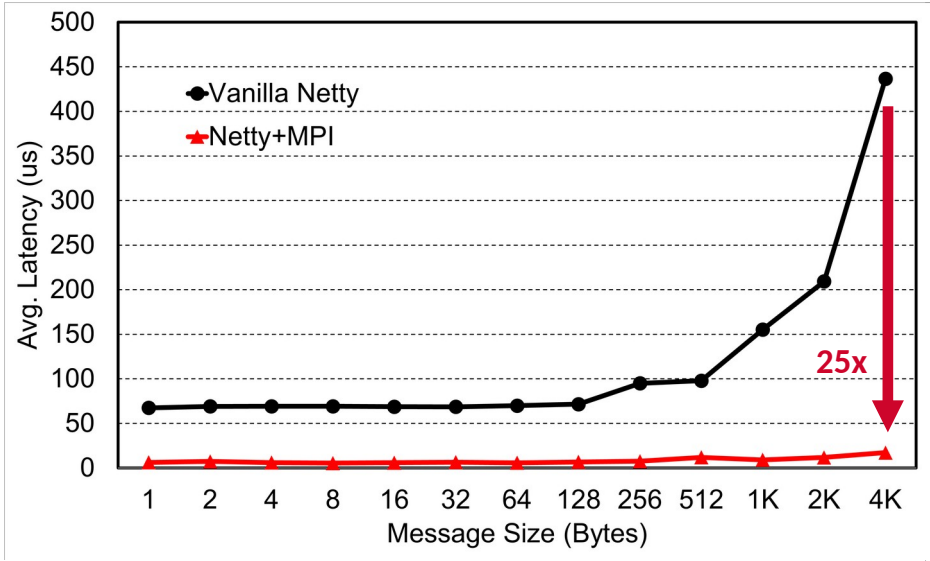
Performance Evaluation

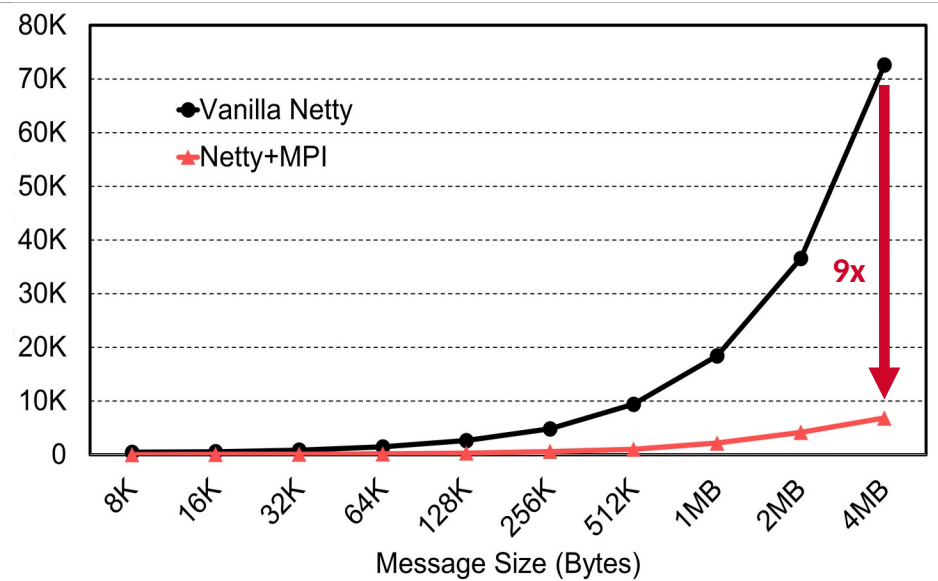
- Communication performance of MPI enhancements at the Netty layer
- MPI4Spark vs. Vanilla Spark and RDMA-Spark using two benchmark suites

Specification	Frontera	Stampede2	Internal Cluster
Number of Nodes	18	10	2
Processor Family	Xeon Platinum	Xeon Platinum	Xeon Broadwell
Clock Speed	2.7 GHz	2.1GHz	2.1GHz
Sockets	2	2	2
Cores Per socket	28	28	14
RAM	192 GB	192 GB	128 GB
Hyper-threading	✗	2 threads/core	✗
Interconnect	IB-HDR (100G)	OPA (100G)	IB-EDR (100G)

MPI4Spark: Performance of MPI-based Netty

- These figures represent the latency numbers for small and large message sizes
- The performance was analyzed using a ping pong Netty benchmark
- For small messages, we see a speed-up of **25x** at 4K
- For large messages, we see a speed-up of **9x** at 4MB



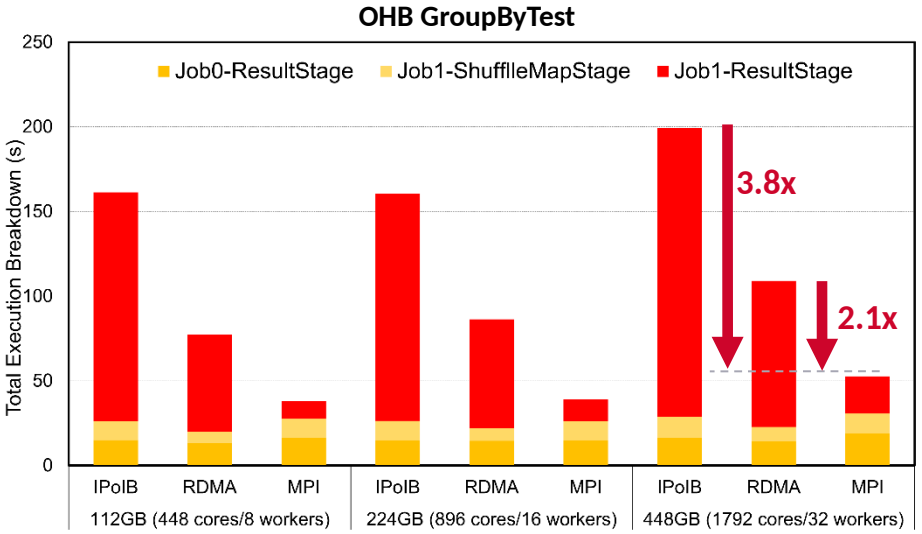


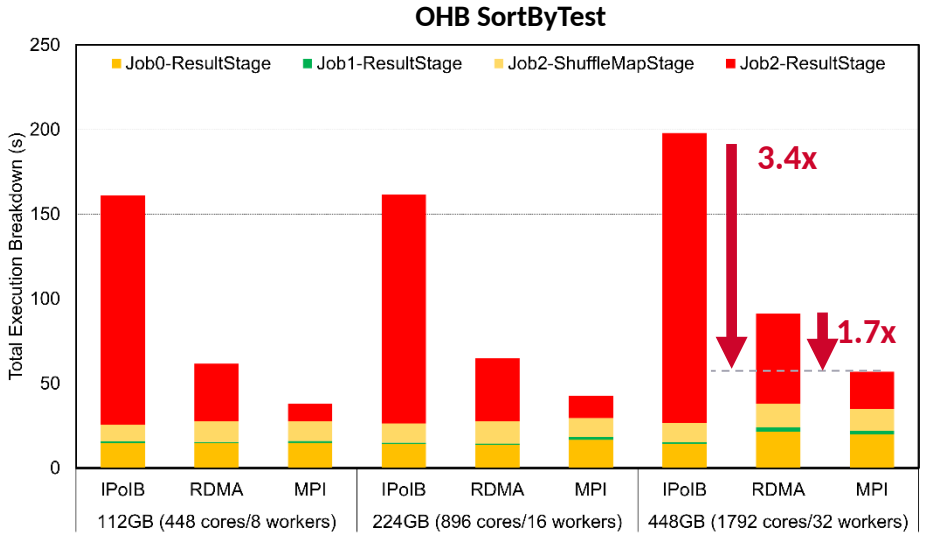
K. Al Attar, A. Shafi, M. Abduljabbar, H. Subramoni, D. Panda, Spark Meets MPI: Towards High-Performance Communication Framework for Spark using MPI, IEEE Cluster '22, Sep 2022.

Performance Evaluation with Intel HiBench and OHB

Benchmark Suite	Workload	Description	Category
Intel HiBench	Support Vector Machine	standard method for large-scale classification tasks	ML
	Latent Dirichlet allocation	a topic model which infers topics from a collection of text documents	
	Gaussian Mixture Model	represents a composite distribution whereby points are drawn from one of k Gaussian sub-distributions	
	Logistic Regression	a popular method to predict a categorical response	
	Repartition	This workload benchmarks shuffle performance	Micro Benchmarks
	TeraSort	A standard benchmark to sort input data	
	Nweight	Computes associations between two vertices that are n-hop away	Graph
OSU HiBD Benchmarks (OHB)	GroupBy	RDD-level benchmark to group the values for each key in the RDD into a single sequence	RDD Benchmarks
	SortBy	RDD-level benchmark to sort the the RDD by key	

Weak Scaling Evaluation with OSU HiBD Benchmarks (OHB)

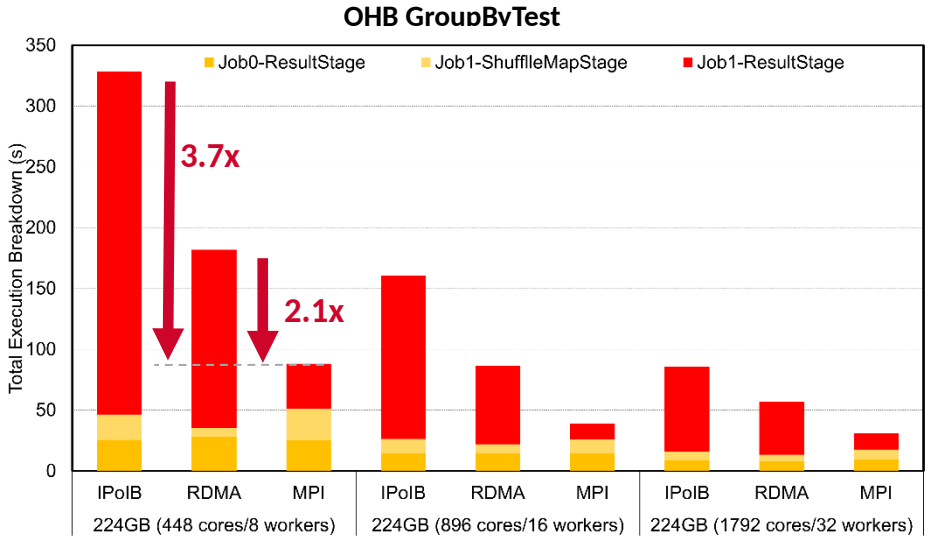


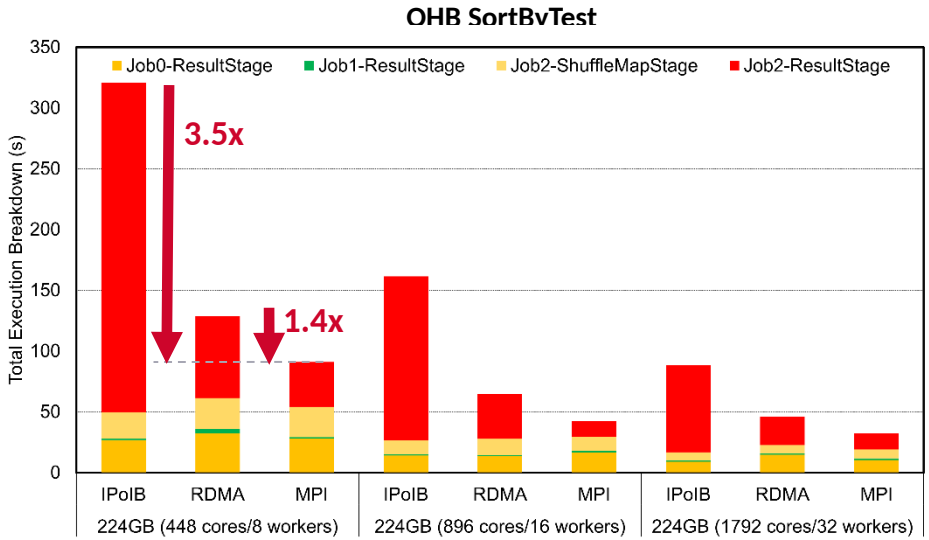


- The above are **weak-scaling** performance numbers of OHB benchmarks (GroupByTest and SortByTest) executed on the TACC Frontera system
- Speed-ups for the overall total execution time for 448GB with GroupByTest is **3.8x** and **2.1x** compared to IPoIB and RDMA, and for SortByTest the speed-ups are **3.4x** and **1.7x**, 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.

Strong Scaling Evaluation with OSU HiBD Benchmarks (OHB)



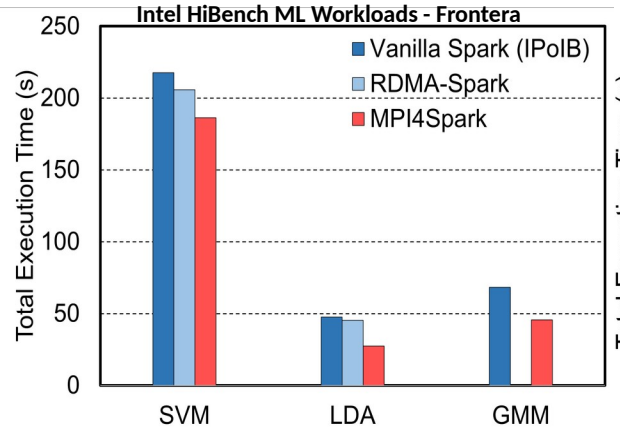


- The above are **strong-scaling** performance numbers of OHB benchmarks (GroupByTest and SortByTest) also executed on the TACC Frontera System
- Speed-ups for the overall total execution time for 8 workers with GroupByTest is **3.7x** and **2.1x** compared to IPoIB and RDMA, and for SortByTest the speed-ups are **3.5x** and **1.4x**, respectively
- Speed-ups for the shuffle read stage for 8 workers GroupByTest between MPI4Spark and IPoIB is **7.6x** and between MPI4Spark and RDMA is **4x**, while for SortByTest the speed-ups are **7.3x** and **1.8x**, respectively

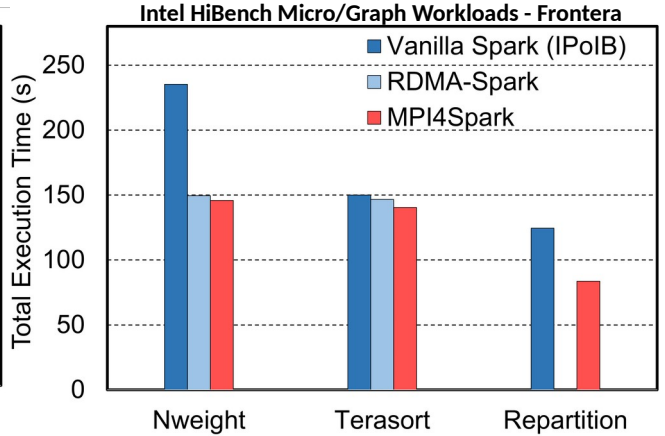
K. Al Attar, A. Shafi, M. Abduljabbar, H. Subramoni, D. Panda, Spark Meets MPI: Towards High-Performance Communication Framework for Spark using MPI, IEEE Cluster '22, Sep 2022.

Performance Evaluation with Intel HiBench Workloads

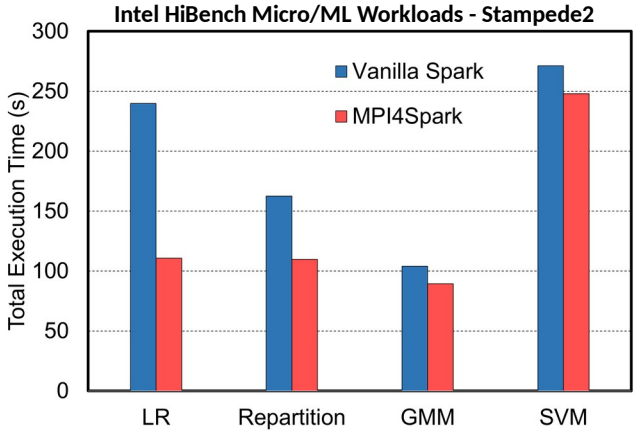
1.4x on average than RDMA-Spark



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

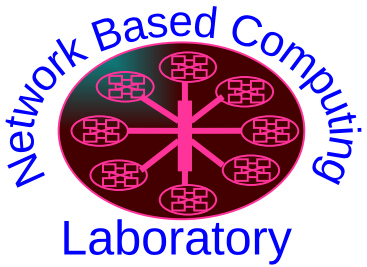
- Background
  - Introduction to Spark and Netty
- Motivation and Challenges
- MPI4Spark Design and Implementation
- MPI4Spark Performance Evaluation
- **Conclusions and Future Work**

Conclusions and Future Work

- Introduced a new design (MPI4Spark) that utilizes MPI communication in the Spark framework at the Netty level for HPC
- MPI4Spark outperformed both Vanilla and RDMA Spark by up to **3.8x** using OHB Benchmarks
- We saw speed-ups of **1.4x** and **1.5x** on average compared to Vanilla and RDMA Spark on Stampede 2 and Frontera using Intel HiBench benchmarks
- The performance evaluation, on TACC's Frontera and Stampede2, showcased the portability of our design on both InfiniBand and Intel Omni-Path interconnects and the performance benefits gained through MPI4Spark
- We plan to release MPI4Spark in the near future
- We also plan to incorporate fault-tolerance along with support for GPU communication in the future

Thank You!

shafi.16@osu.edu



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/