A Micro-Benchmark Suite for Evaluating Hadoop MapReduce on High-Performance Networks

Dipti Shankar, **Xiaoyi Lu**, Md. Wasi-ur-Rahman, Nusrat Islam, and Dhabaleswar K. (DK) Panda

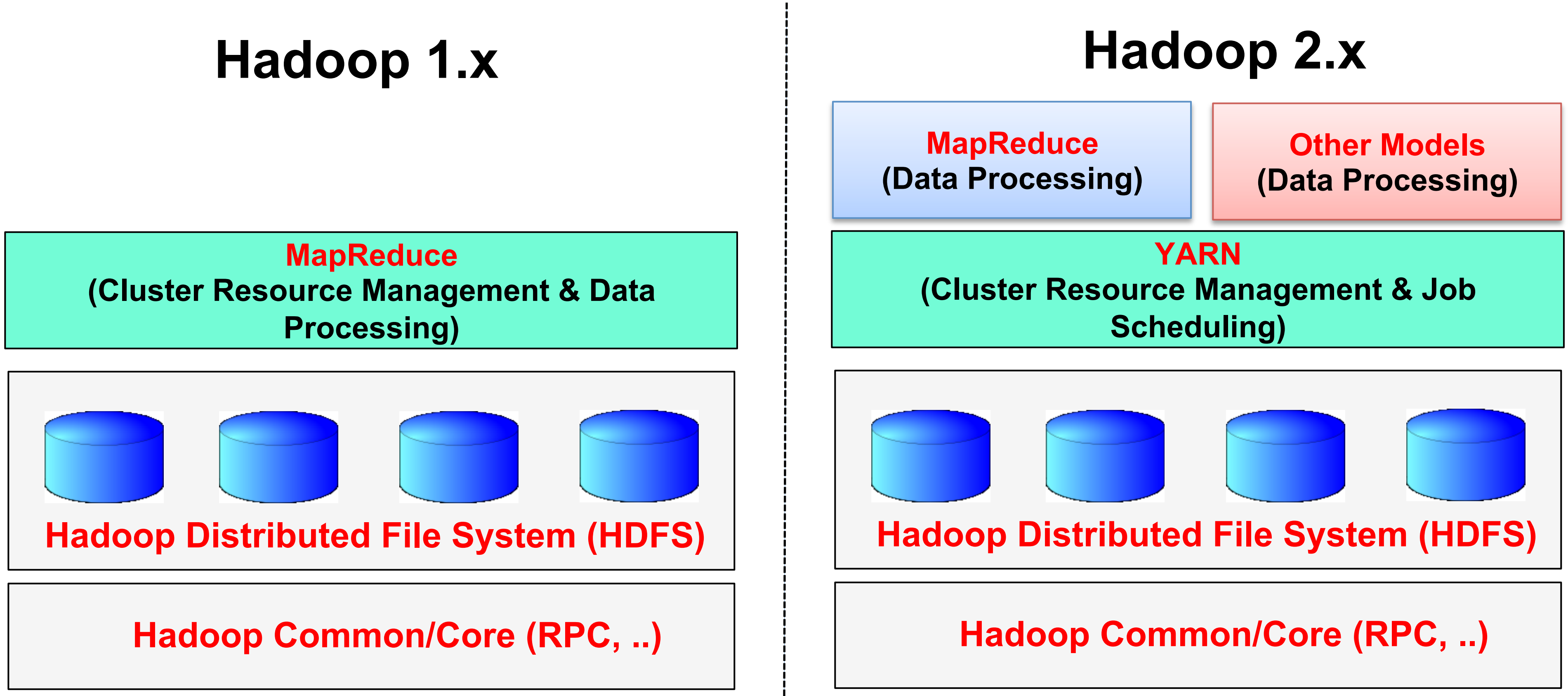
Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University, Columbus, OH, USA

Outline

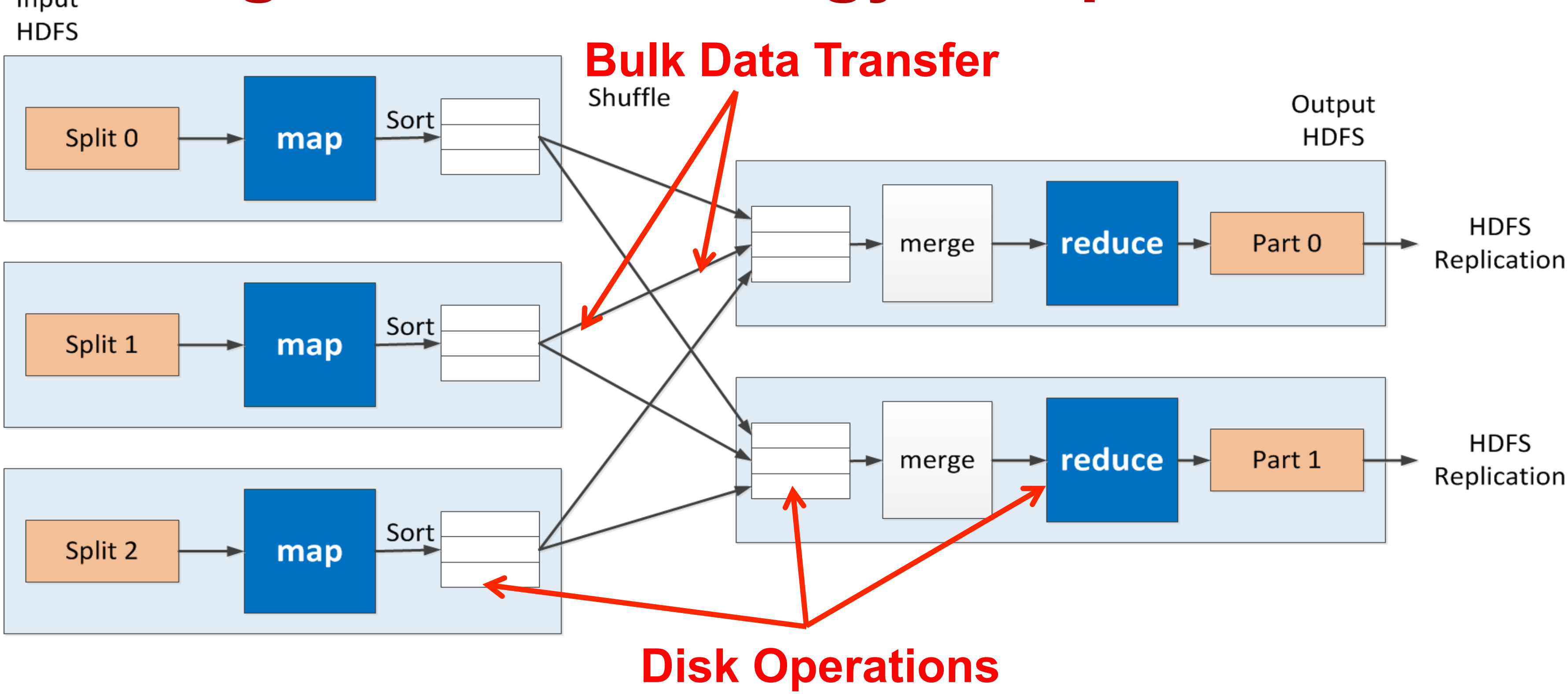
- Introduction & Motivation
- Design Considerations
- Micro-benchmark Suite
- Performance Evaluation
- Case Study with RDMA
- Conclusion & Future work

Big Data Technology - Hadoop

- Apache Hadoop is one of the most popular Big Data technology
  - Provides frameworks for large-scale, distributed data storage and processing
  - MapReduce, HDFS, YARN, RPC, etc.



Big Data Technology - MapReduce



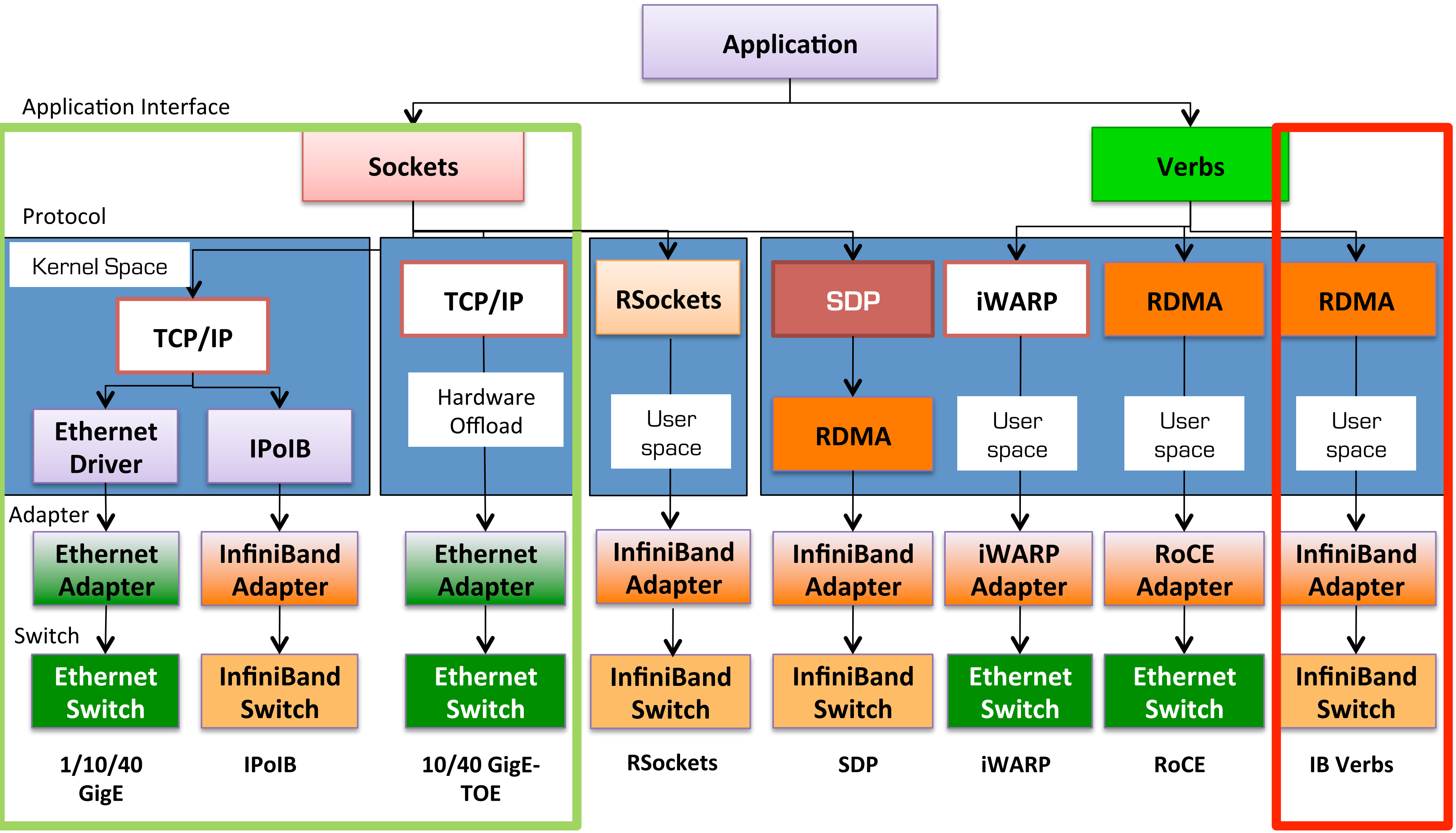
- Map and Reduce Tasks carry out the total job execution
  - Map tasks read from HDFS, operate on it, and write the intermediate data to local disk
  - Reduce tasks get these data by shuffle from TaskTrackers, operate on it and write to HDFS
- Scalable and communication intensive
  - Data shuffling

Factors Effecting Performance of Hadoop MapReduce

Performance of Hadoop MapReduce is influenced by many factors

- Network configuration of cluster
- Multi-core architecture
- Memory system
- Underlying storage system
  - Example: HDFS, Lustre etc.
- Controllable parameters in software
  - Number of Mappers and Reducers, Partitioning scheme used
- Many others …

Common Protocols using Open Fabrics



Can High-Performance Networks Benefit Big Data Processing?

- Previous studies: very good performance improvements for Hadoop (HDFS /MapReduce/RPC), Spark, HBase, Memcached over InfiniBand/RoCE
  - Hadoop Acceleration with RDMA
    - N. S. Islam, et.al., SOR-HDFS: A SEDA-based Approach to Maximize Overlapping in RDMA -Enhanced HDFS, HPDC'14
    - N. S. Islam, et.al., High Performance RDMA-Based Design of HDFS over InfiniBand, SC'12
    - M. W. Rahman, et.al. HOMR: A Hybrid Approach to Exploit Maximum Overlapping in MapReduce over High Performance Interconnects, ICS'14
    - M. W. Rahman, et.al., High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand, HPDIC'13
    - X. Lu, et. al., High-Performance Design of Hadoop RPC with RDMA over InfiniBand, ICPP'13
  - Spark Acceleration with RDMA
    - X. Lu, et. al., Accelerating Spark with RDMA for Big Data Processing: Early Experiences, HotI'14
  - HBase Acceleration with RDMA
    - J. Huang, et.al., High-Performance Design of HBase with RDMA over InfiniBand, IPDPS'12
  - Memcached Acceleration with RDMA
    - J. Jose, et. al., Scalable Memcached design for InfiniBand Clusters using Hybrid Transports, Cluster'11
    - J. Jose, et.al., Memcached Design on High Performance RDMA Capable Interconnects, ICPP'11

The High-Performance Big Data (HiBD) Project

- RDMA for Apache Hadoop 2.x (RDMA-Hadoop-2.x)
- RDMA for Apache Hadoop 1.x (RDMA-Hadoop)
- RDMA for Memcached (RDMA-Memcached)
- OSU HiBD-Benchmarks (OHB)
- **http://hibd.cse.ohio-state.edu**
- RDMA for Apache HBase and Spark

HiBD High-Performance Big Data

Network Based Computing Laboratory

THE OHIO STATE UNIVERSITY

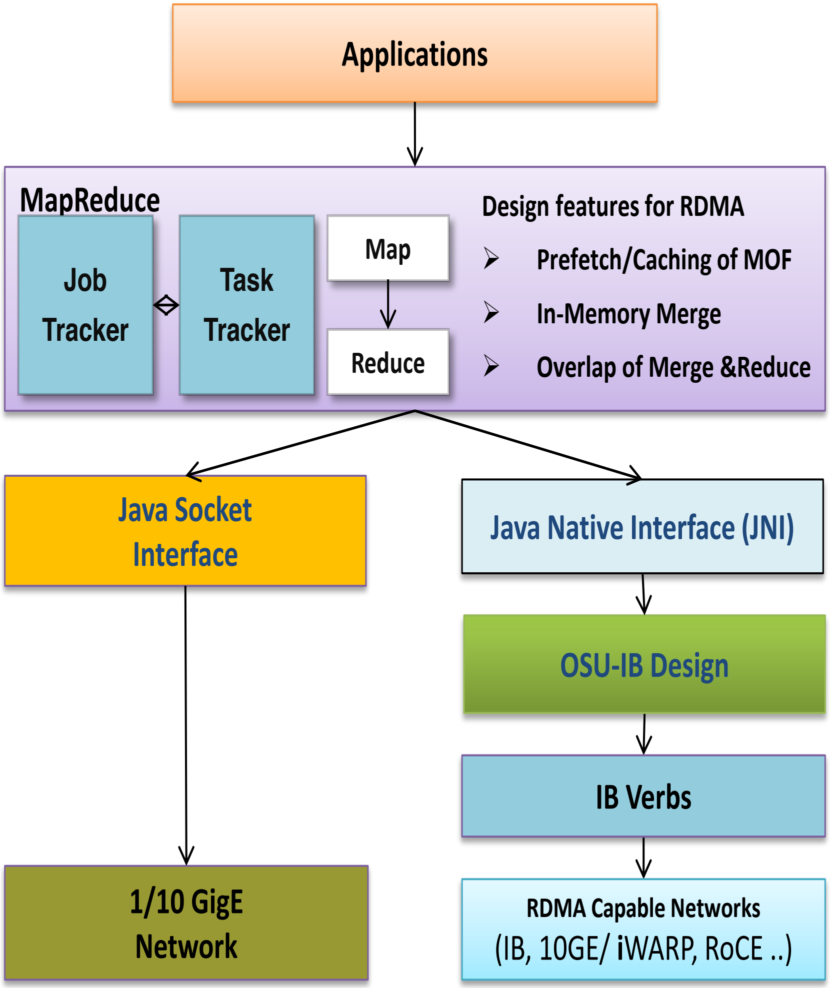
RDMA for Apache Hadoop 1.x/2.x Distributions

- High-Performance Design of Hadoop over RDMA-enabled Interconnects
  - High performance design with native InfiniBand and RoCE support at the verbs-level for HDFS, **MapReduce**, and RPC components
  - Easily configurable for native InfiniBand, RoCE and the traditional sockets-based support (Ethernet and InfiniBand with IPoIB)
- Current release: 0.9.9 (03/31/14)
  - Based on Apache Hadoop 1.2.1
  - Compliant with Apache Hadoop 1.2.1 APIs and applications
  - Tested with
    - Mellanox InfiniBand adapters (DDR, QDR and FDR)
    - RoCE support with Mellanox adapters
    - Various multi-core platforms, different file systems with disks and SSDs
- **RDMA for Apache Hadoop 2.x 0.9.1 is released in HiBD!**

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

Design Overview of MapReduce with RDMA

- Enables high performance RDMA communication, while supporting traditional socket interface
- JNI Layer bridges Java based MapReduce with communication library written in native code
- Design features
  - RDMA-based shuffle
  - Prefetching and caching map output
  - Efficient Shuffle Algorithms
  - In-memory merge
  - On-demand Shuffle Adjustment
  - Advanced overlapping
    - map, shuffle, and merge
    - shuffle, merge, and reduce
  - On-demand connection setup
  - InfiniBand/RoCE support



M. Wasi-ur-Rahman et al., High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand, HPDIC'13

M. Wasi-ur-Rahman et al., HOMR: A Hybrid Approach to Exploit Maximum Overlapping in MapReduce over High Performance Interconnects, ICS'14

Existing Benchmarks for Evaluating Hadoop MapReduce

Evaluation of performance Hadoop MapReduce job

Micro-benchmarks	Description
Sort	Sort random data, I/O bound
TeraSort	Sort in total order, Map Stage is CPU bound, Reduce Stage is I/O bound
Wordcount	CPU bound, Heavy use of partitioner
Other Benchmarks	**Description**
HiBench	Representative and comprehensive suite with both synthetic micro-benchmarks and real-world workloads
MRBS	Five benchmarks covering several application domains for evaluating the dependability of MapReduce systems
MRBench	Focuses on processing business oriented queries and concurrent data modifications
PUMA	Represents broad range of MapReduce applications with high/low computation and high/low shuffle volumes
SWIM	Real life MapReduce workloads from production systems, workload synthesis and replay tools for sampling traces

Existing Benchmarks for Evaluating Hadoop MapReduce (contd.)

- All benchmarks mentioned require the **involvement of HDFS** or some distributed file system
  - Interferes in the evaluation of the MapReduce's performance
  - Hard to benchmark and compare different shuffle and sort schemes
  - Benchmarks do not provision us to study different shuffle patterns and **impact of network protocols** on them
- Requirement: we need a way to evaluate MapReduce as an independent component !
  - To illustrate performance improvement and potential of new MapReduce designs
  - To help tune internal parameters specific to MapReduce and obtain optimal performance over high-performance networks

Can we benefit from a stand-alone MapReduce Micro-benchmark?

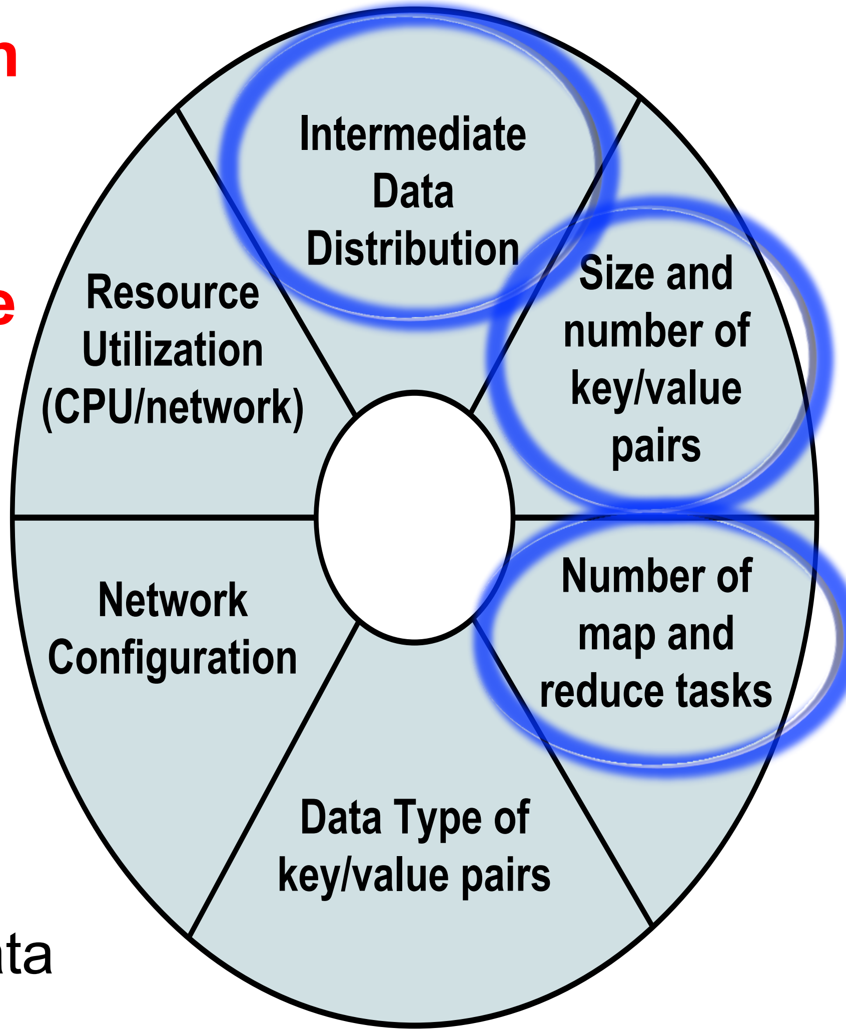
- Can we design a **simple micro-benchmark suite**
  - *That lets users and developers **evaluate Hadoop MapReduce in a stand-alone manner** over different networks or protocols?*
  - *Helps tune and optimize configurable parameters based on cluster and workload characteristics?*
  - *Helps us evaluate the performance of new or alternate MapReduce frameworks such as **our proposed MapReduce over RDMA** design?*
  - *That provisions studying the impact of different data distribution patterns, data types, etc., on the performance of the MapReduce job?*
- What will be the performance of stand-alone Hadoop MapReduce when evaluated on different high-performance networks?
- Basic motivation!

Outline

- Introduction & Motivation
- Design Considerations
- Micro-benchmark Suite
- Performance Evaluation
- Case Study with RDMA
- Conclusion & Future work

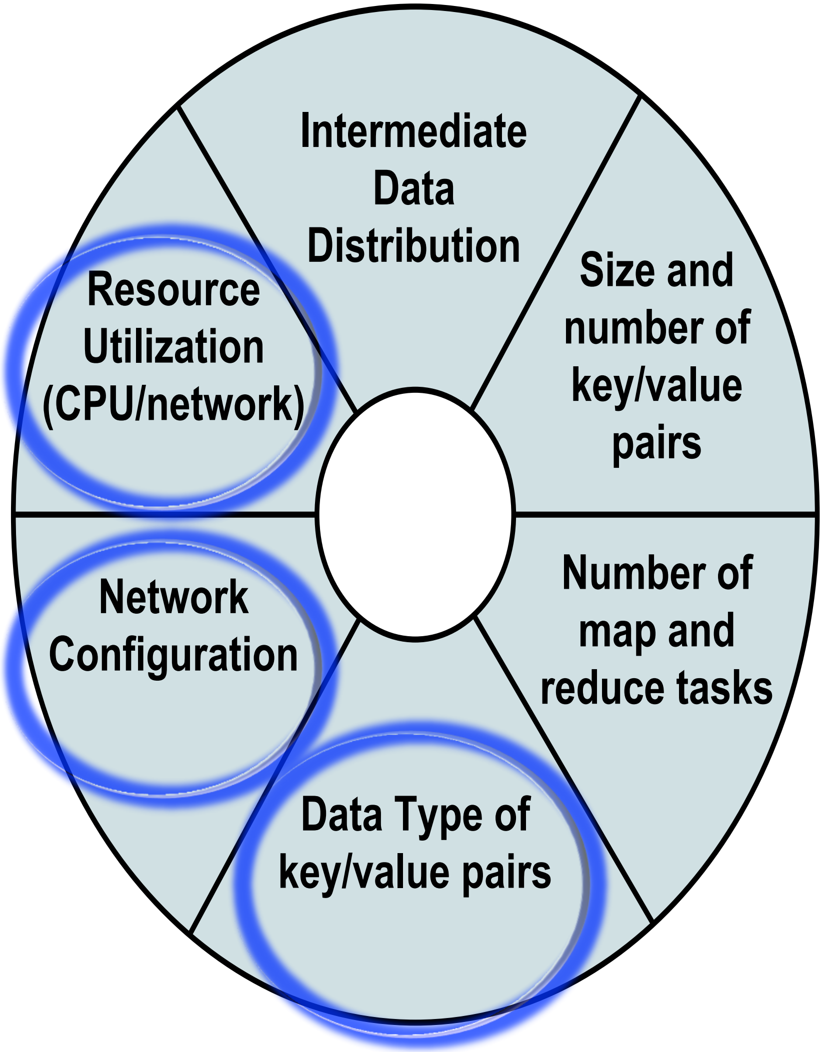
Design Considerations

- **Intermediate data distribution**
  - Is the shuffle data partitioned evenly amongst all reducers?
- **Size and number of key/value pairs**
  - Does the size of intermediate data matter?
  - Size Vs. Number of key/value pairs
- **Number of map and reduce tasks**
  - Number of tasks generating processing the intermediate data



Design Considerations (contd.)

- **Data type of key/value pairs**
  - E.g. Bytes Vs. Text
- **Network Configuration**
  - Help to evaluate performance of different networks for MapReduce workloads
  - E.g. 1GigE/10GigE/IB
- **Resource Utilization**
  - Correlation between workload characteristics and resource utilization in MapReduce



Outline

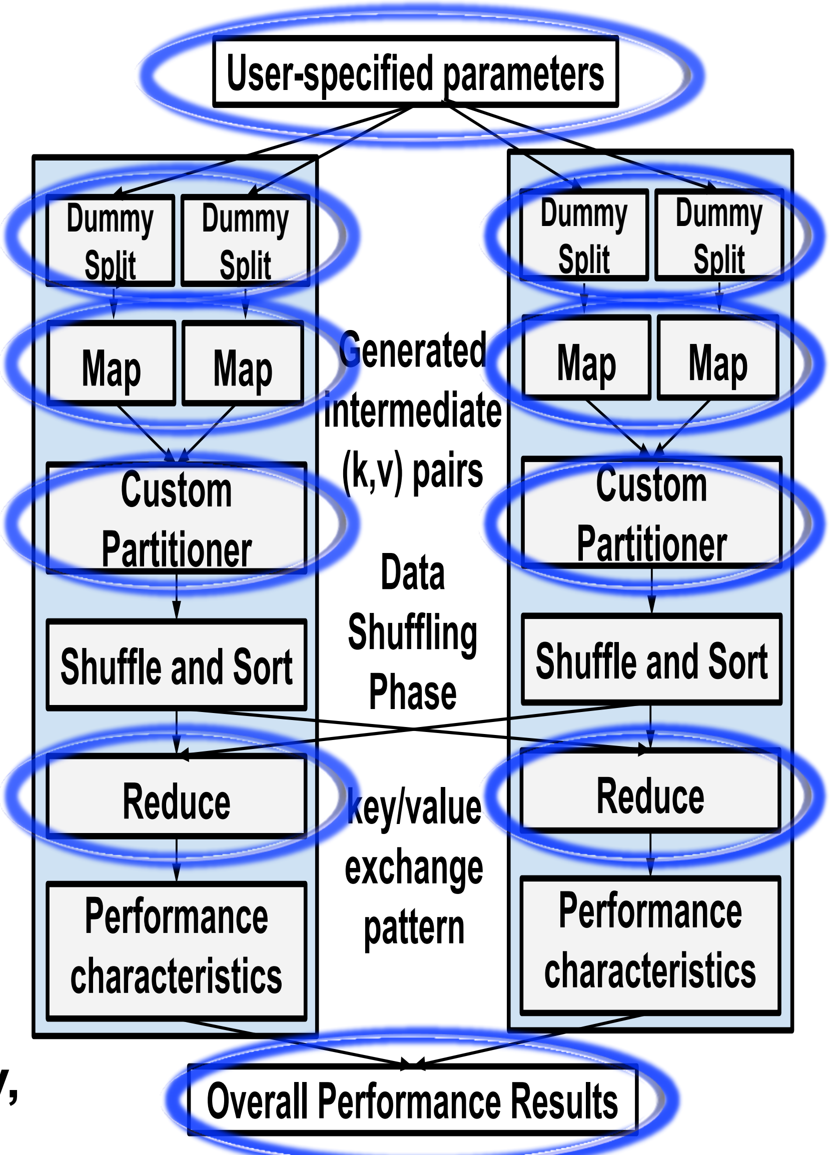
- Introduction & Motivation
- Design Considerations
- Micro-benchmark Suite
- Performance Evaluation
- Case Study with RDMA
- Conclusion & Future work

MapReduce Micro-Benchmark Suite

- Evaluate the performance of stand-alone MapReduce
  - Does not require or involve HDFS or any distributed file system
- Considers various factors that influence the data shuffling phase
  - Underlying network configuration, number of map and reduce tasks, intermediate shuffle data pattern, shuffle data size etc.
- Currently supports three different shuffle data distribution patterns
  - **Average data distribution:** intermediate data is evenly distributed among reduce tasks
  - **Random data distribution:** intermediate data is pseudo-randomly distributed among reduce tasks
  - **Skewed data distribution:** intermediate data is unevenly distributed among reduce tasks

Micro-Benchmark Suite Design

- **Stand-alone feature**
  - Simple No-HDFS MapReduce job
  - Map Phase
    - Custom Input Format i.e., ***NullInputFormat***
    - key/value pairs generated **in memory**
  - Reduce Phase uses Hadoop API's ***NullOutputFormat***
- **Custom Partitioners**
  - Simulate different intermediate data distribution scenarios described
- **Configurable Parameters**
  - Number of maps and reducers
  - Intermediate shuffle data size
    - Number of key/value pairs per map
    - Size of key/value pairs
  - Data type
- **Calculates statistics like job latency, CPU/Network utilization**



MapReduce Micro-Benchmarks

Three MapReduce Micro-benchmarks defined

1) MR-AVG micro-benchmark

- Intermediate data is evenly distributed among reduce tasks
- Custom partitioner distributes same number of intermediate key/value pairs amongst the reducers in a round-robin fashion
- Uniform distribution and fair comparison for all runs

2) MR-RAND micro-benchmark

- Intermediate data is pseudo-randomly distributed among reduce tasks
- Custom partitioner picks a reducer randomly and assigns the key/value pair to it
- Fair comparison for all runs on homogenous systems

MapReduce Micro-Benchmarks (contd.)

3) MR-SKEW micro-benchmark

- Intermediate data is unevenly distributed among reduce tasks
- Custom partitioner uses fixed skewed distribution pattern
  - 50% to reducer 0
  - 25% to reducer 1
  - 12.5% to reducer 2
  - …
- Skewed distribution pattern is fixed for all runs
- Fair comparison for all runs on homogenous systems

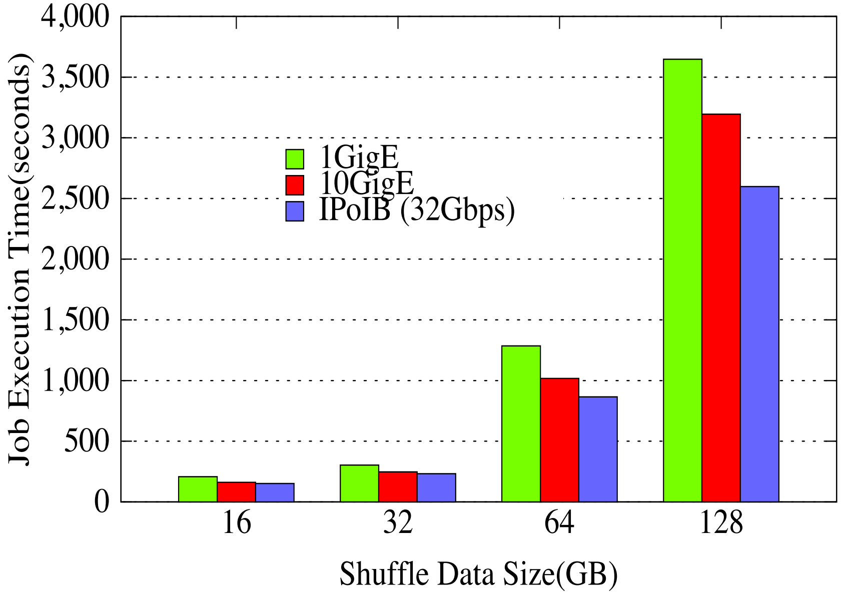
Outline

- Introduction & Motivation
- Design Considerations
- Micro-benchmark Suite
- Performance Evaluation
- Case Study with RDMA
- Conclusion & Future work

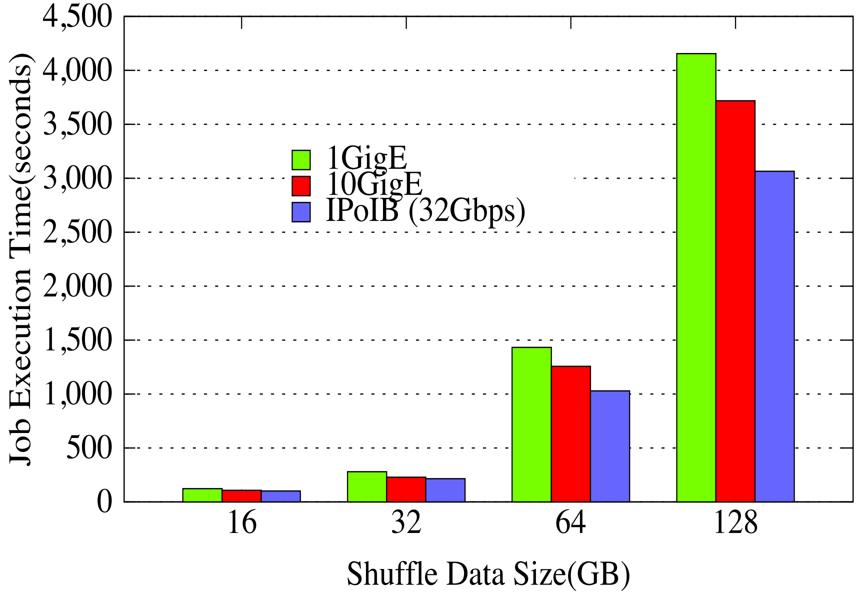
Experimental Setup

- **Hardware**
  - **Intel Westmere Cluster (A)** (Up to 9 nodes)
    - Each node has 8 processor cores on 2 Intel Xeon 2.67 GHz quad-core CPUs, 24 GB main memory
    - Mellanox QDR HCAs (32 Gbps) + 10GigE
  - **TACC Stampede Cluster (B)** (Up to 17 nodes)
    - Intel Sandy Bridge (E5-2680) dual octa-core processors, running at 2.70GHz, 32 GB main memory
    - Mellanox FDR HCAs (56 Gbps)
  - Performance comparisons over 1 GigE, 10 GigE, IPoIB (32 Gbps) and, IPoIB (56 Gbps)
- **Software**
  - JDK 1.7.0
  - Apache Hadoop 1.2.1
  - Apache Hadoop NextGen MapReduce (YARN) 2.4.1

Performance Evaluation with Different Data Distribution Patterns



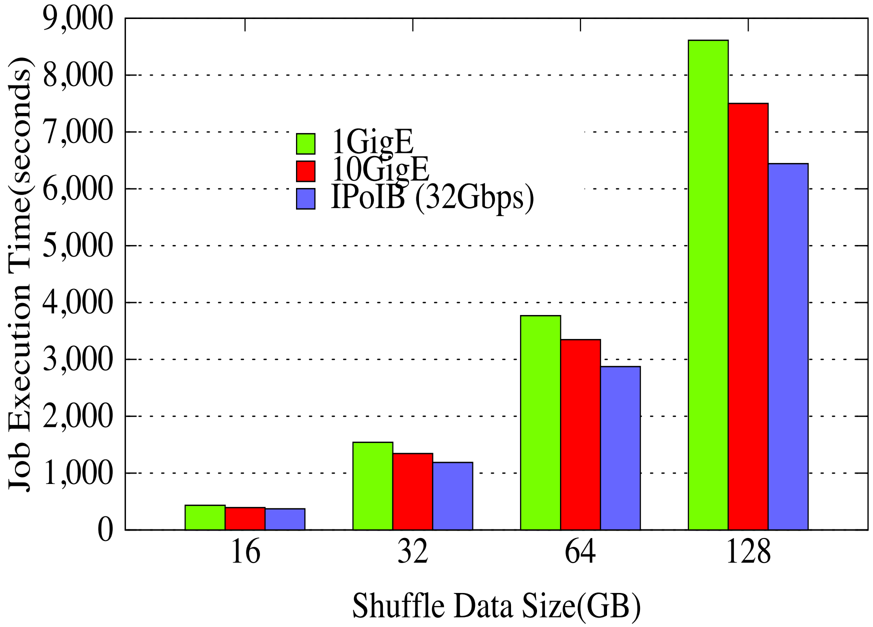
MR-AVG micro-benchmark



MR-RAND micro-benchmark

- Comparing different shuffle data distribution patterns on Cluster A, key/value pair size of 1 KB on 4 node cluster,16 maps and 8 reduces
- For MR-AVG, IPoIB (32Gbps) gives a **24%** improvement over 1 GigE and up to **17%** over 10 GigE
- For MR-RAND, IPoIB (32Gbps) gives a **22%** improvement over 1 GigE and up to **15%** over 10 GigE

Performance Evaluation with Different Data Distribution Patterns

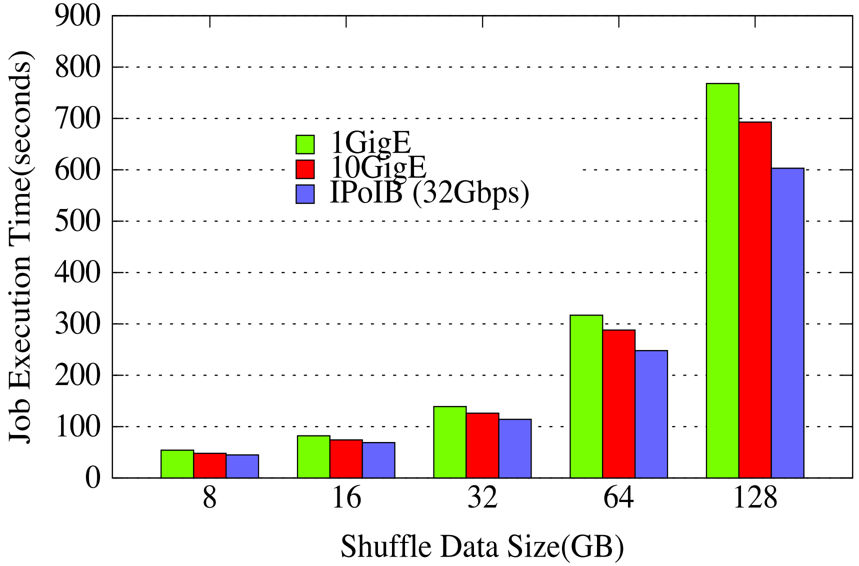


MR-SKEW micro-benchmark

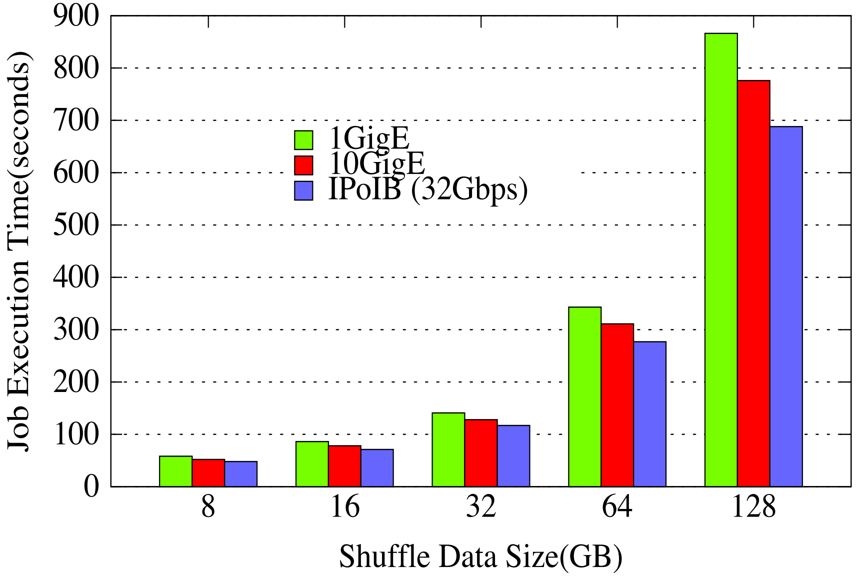
- For MR-SKEW, 10 GigE gives up to **13%** improvement over 1 GigE
- For MR-SKEW, IPoIB (32Gbps) gives up to **14%** improvement over 10 GigE

- IPoIB (32Gbps) provides better improvement with increased shuffle data sizes and more skewed workloads
- Skewed data distribution causes 2x increase in job execution time for a given data size, irrespective of the underlying network interconnect

Performance Evaluation with Apache Hadoop NextGen MapReduce (YARN)



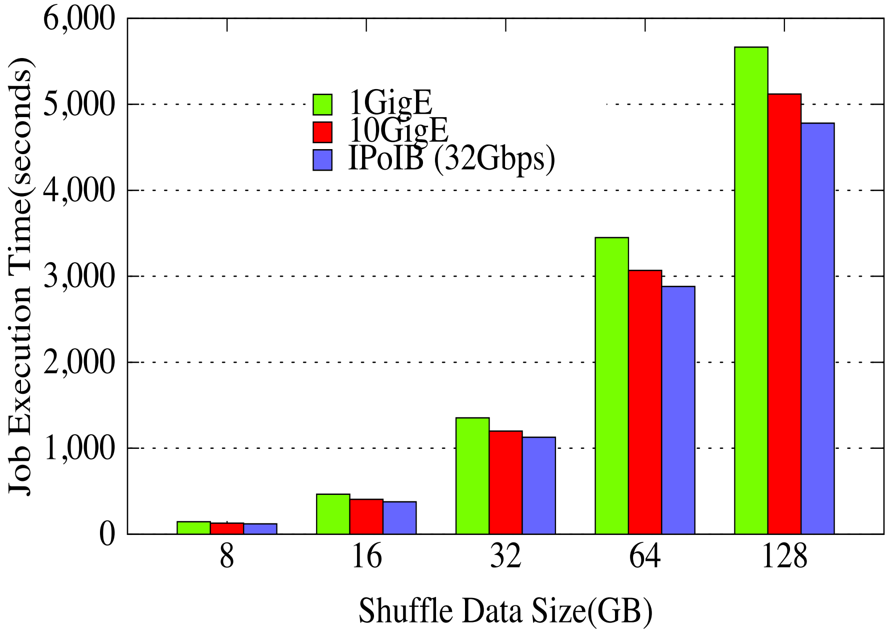
MR-AVG micro-benchmark



MR-RAND micro-benchmark

- Running Hadoop 2.4.1 with different data distributions on Cluster A, key/value pair size of 1 KB on 8 node cluster, 32 maps and 16 reducers
- For MR-AVG, IPoIB (32Gbps) gives a **21%** improvement over 1 GigE and up to **12%** over 10 GigE
- For MR-RAND, IPoIB (32Gbps) gives a **19%** improvement over 1 GigE and up to **11%** over 10 GigE

Performance Evaluation with Apache Hadoop NextGen MapReduce (YARN)

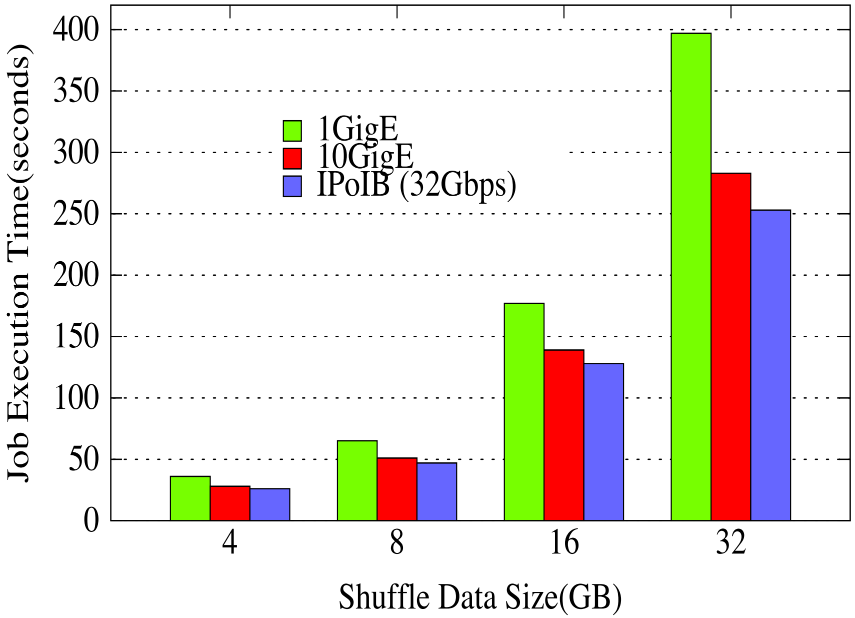


MR-SKEW micro-benchmark

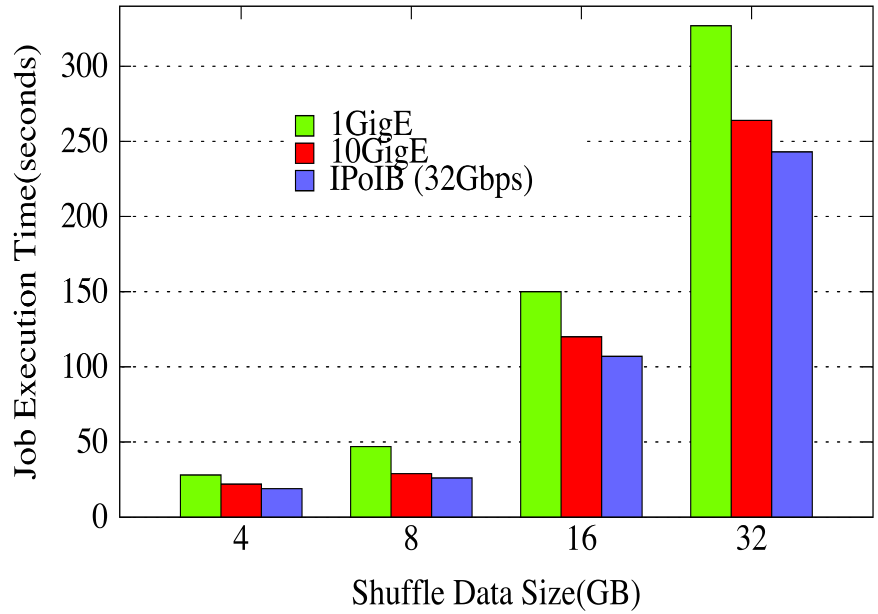
- For MR-SKEW, 10 GigE gives up to **12%** improvement over 1 GigE
- For MR-SKEW, IPoIB (32Gbps) gives up to **15%** over 10 GigE

- Skewed data distribution
  - causes around 3x increase in job execution time for a given data size
  - Increased concurrency does not show significant improvement as Reduce phase still depends on the slowest reduce task

Performance Evaluation with Varying Key/Value Pair Sizes



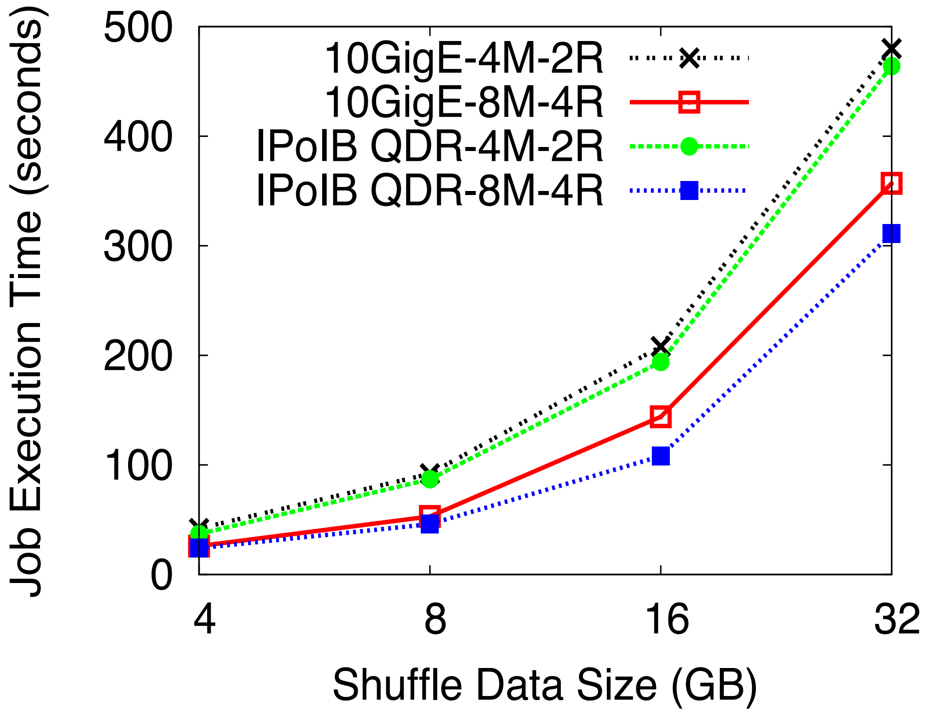
MR-AVG micro-benchmark with 100 B key/value size



MR-AVG micro-benchmark with 10 KB key/value size

- MR-AVG on Cluster A, on 4 node cluster, 16 maps and 8 reduces
- For both key/value pair sizes, IPoIB (32Gbps) gives up to
  - **22-24%** improvement over 1 GigE and
  - **10-13%** improvement over 10 GigE
- Increasing key/value pair sizes lowers job execution time for a given shuffle data size

Performance Evaluation with Varying Number of Map and Reduce Tasks



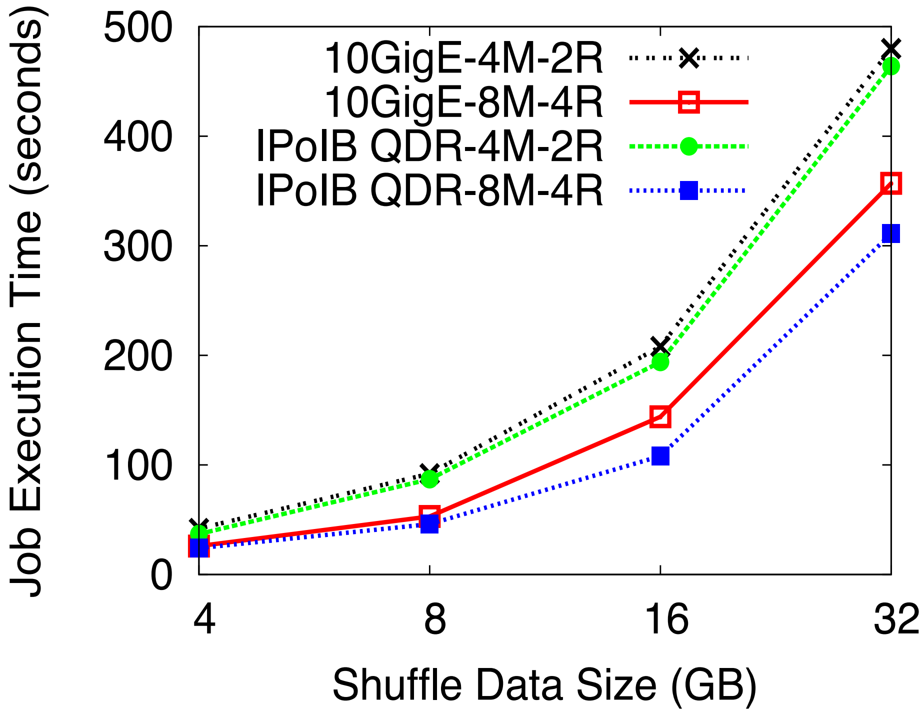
MR-AVG with different number of maps and reduces

- MR-AVG on Cluster A, on 4 node cluster, 16 maps and 8 reduces, 1 KB key/value pair size
- Performance evaluations with
  - 8 map and 4 reduce tasks (8M-4R)
  - 4 map and 2 reduce tasks (4M-2R)
- IPoIB (32Gbps) outperforms 10 GigE by about **13% on an average** (over different data sizes)

- IPoIB (32Gbps) gives better improvement with increasing concurrency

  For instance
  - up to 32% for IPoIB (32Gbps) for 32 GB shuffle data size
  - up to 24% for 10GigE for 32 GB shuffle data size

Performance Evaluation with Varying Number of Map and Reduce Tasks



MR-AVG with different number of maps and reduces

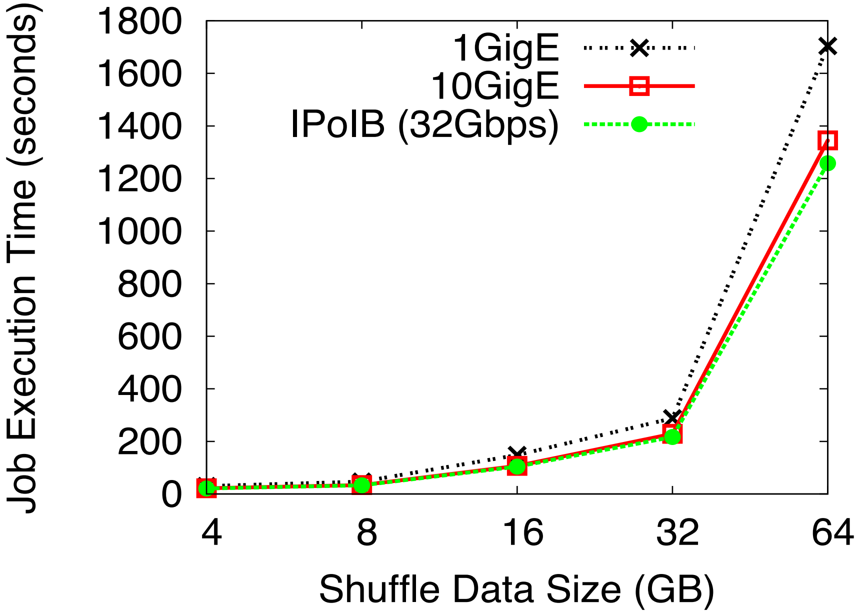
- MR-AVG on Cluster A, on 4 node cluster, 16 maps and 8 reduces, 1 KB key/value pair size
- Performance evaluations with
  - 8 map and 4 reduce tasks (8M-4R)
  - 4 map and 2 reduce tasks (4M-2R)
- IPoIB (32Gbps) outperforms 10 GigE by about **13% on an average** (over different data sizes)

- IPoIB (32Gbps) gives better improvement with increasing concurrency

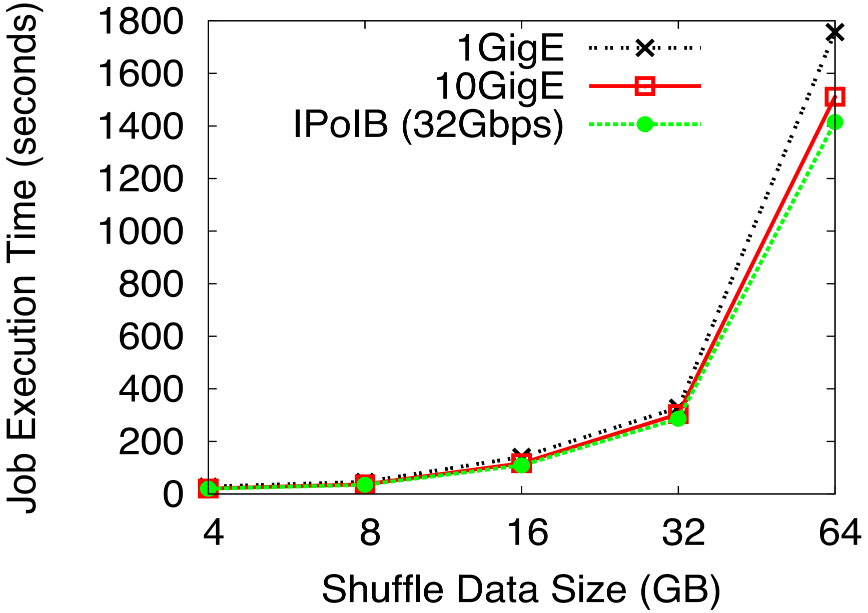
  For instance

  - up to 32% for IPoIB (32Gbps) for 32 GB shuffle data size
  - up to 24% for 10GigE for 32 GB shuffle data size

Performance Evaluation with Different Data Types



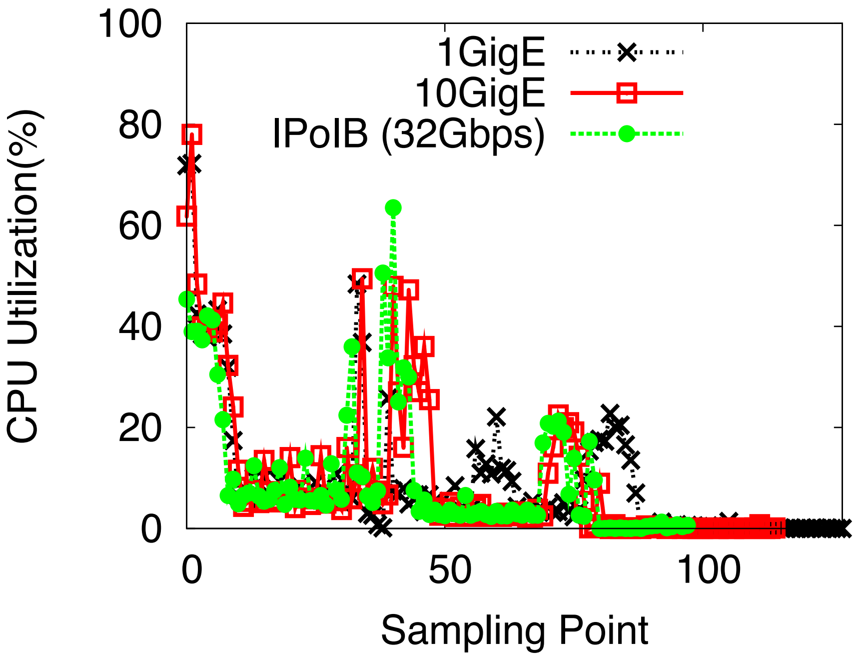
MR-RAND micro-benchmark with BytesWritable



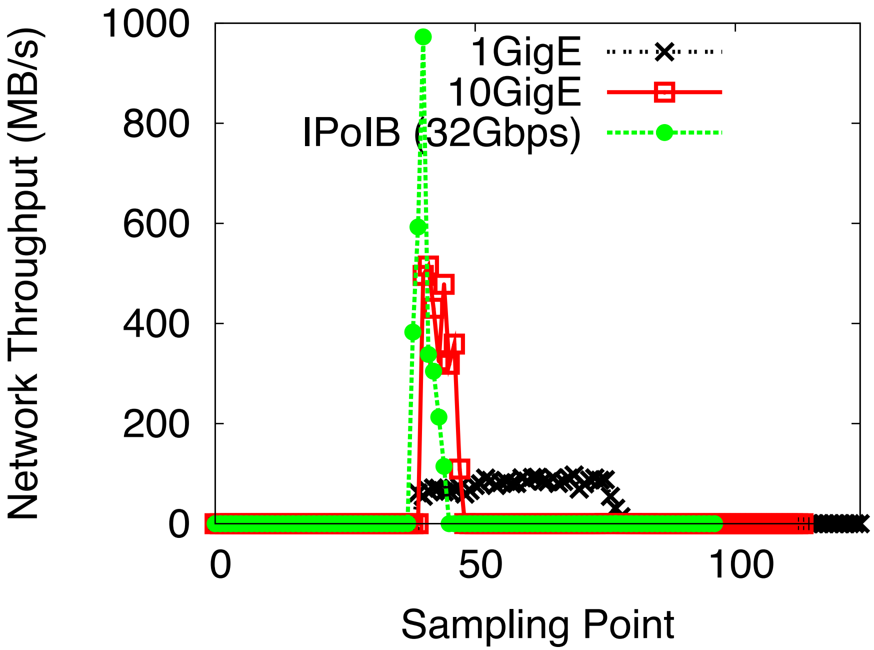
MR-RAND micro-benchmark with Text

- MR-RAND on Cluster A, key/value pair size of 1 KB on 4 node cluster, 16 maps and 8 reducers
- For both BytesWritable and Text Data Type, IPoIB (32Gbps) shows,
  - significant improvement potential for larger data sizes over 10 GigE on an average
  - similar improvement potential to both types

Resource Utilization Analysis



CPU utilization one one slave node



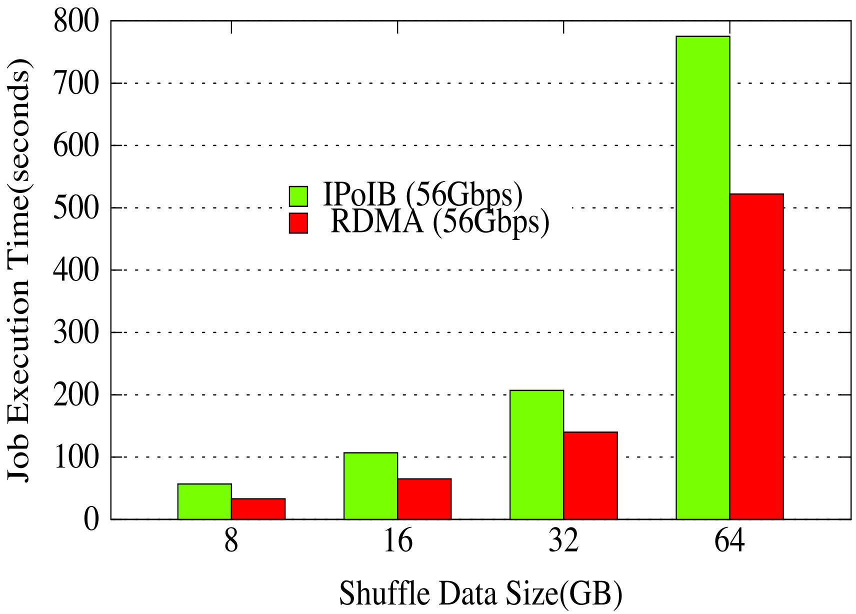
Network Throughput on one slave node

- MR-AVG on Cluster A, intermediate data size of 16 GB, key/value pair size of 1 KB on 4 node cluster,16 maps and 8 reduces
- IPoIB (32Gbps) gives best peak bandwidth at 950 MB/s
- CPU trends are similar for different interconnects

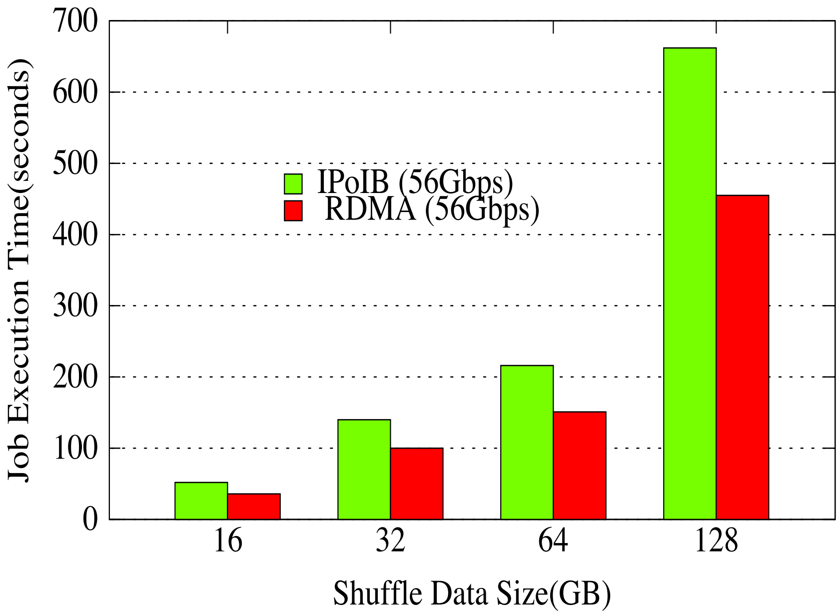
Outline

- Introduction & Motivation
- Design Considerations
- Micro-benchmark Suite
- Performance Evaluation
- Case Study with RDMA
- Conclusion &Future Work

Performance of IPoIB Vs. RDMA – MR-AVG



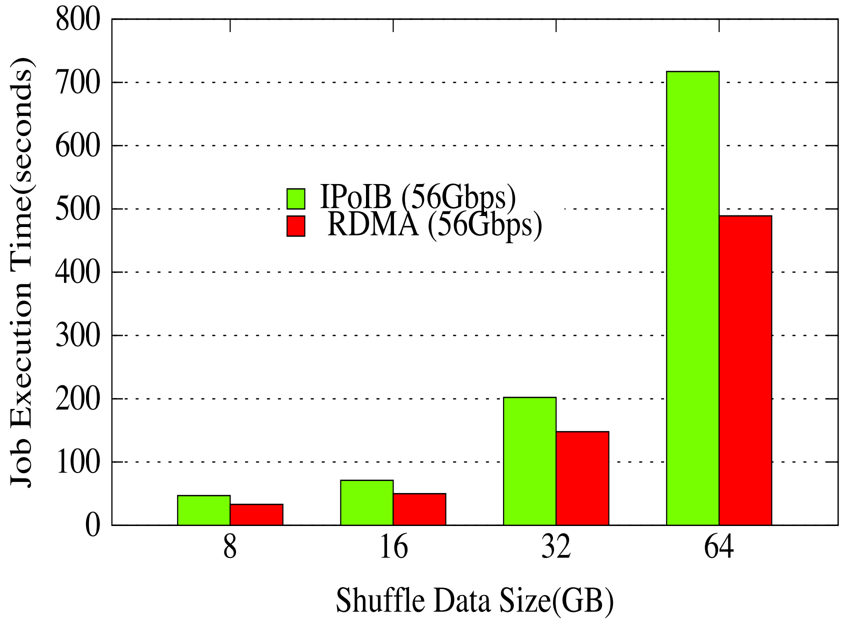
MR-AVG with 8 slave nodes on Cluster B



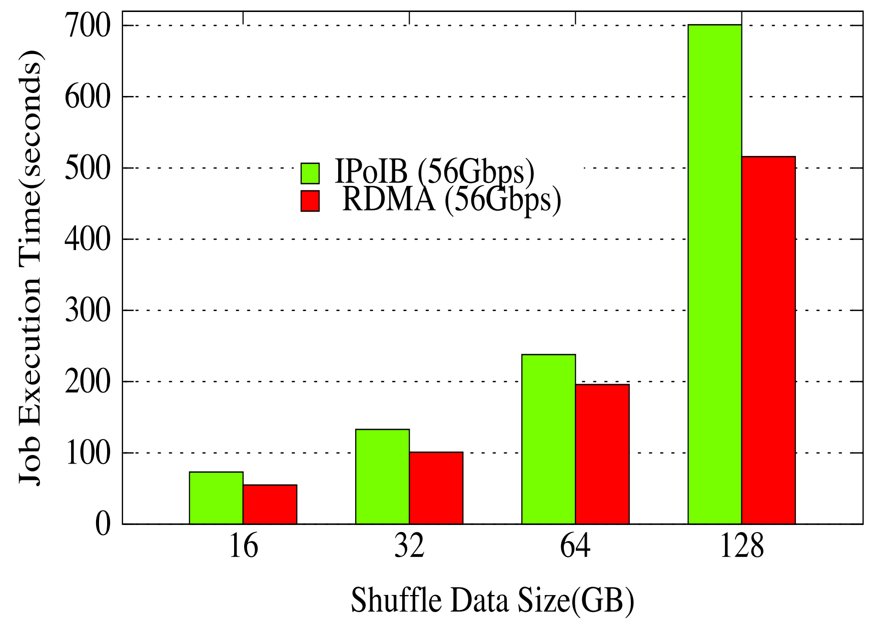
MR-AVG with 16 slave nodes on Cluster B

- Cluster B, key/value pair size of 1 KB, 32 maps and 16 reducers, on 8 and 16 node cluster
- For MR-AVG, RDMA-based MapReduce over over IPoIB (56Gbps) gives an improvement of
  – Up to **30%** for 8 slave nodes (128 cores)
  – Up to **28%** for 16 slave nodes (256 cores)

Performance of IPoIB Vs. RDMA – MR-RAND



MR-RAND with 8 slave nodes on Cluster B



MR-RAND with 16 slave nodes on Cluster B

- For MR-RAND, RDMA-based MapReduce over IPoIB (56Gbps) gives an improvement of
  - Up to **28%** for 8 slave nodes (128 cores)
  - Up to **25%** for 16 slave nodes (256 cores)
- Illustration performance improvement obtained by leveraging RDMA for shuffle and sort phase of MapReduce

Outline

- Introduction & Motivation
- Design Considerations
- Micro-benchmark Suite
- Performance Evaluation
- Case Study with RDMA
- Conclusion & Future work

Conclusion and Future Work

- Study of factors that can significantly impact performance of Hadoop MapReduce (like network protocol etc.)
- Design, development and implementation of a micro-benchmark suite for stand-alone MapReduce
  - Help developers enhance their MapReduce designs
  - Support for both Hadoop 1.x and 2.x
- Performance evaluations with our micro-benchmark suite over different interconnects on modern clusters
- Future Work
  - Enhance micro-benchmark suite to real-world workloads
  - Investigate more data types
  - Make the micro-benchmarks publicly available through the HiBD project (OHB)

Thank You!

{shankard, luxi, rahmanmd, islamn, panda}
@cse.ohio-state.edu

Network-Based Computing Laboratory

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

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