Accelerating Spark with RDMA for Big Data Processing: Early Experiences

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

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

Outline

- Introduction
- Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future work

Digital Data Explosion in the Society

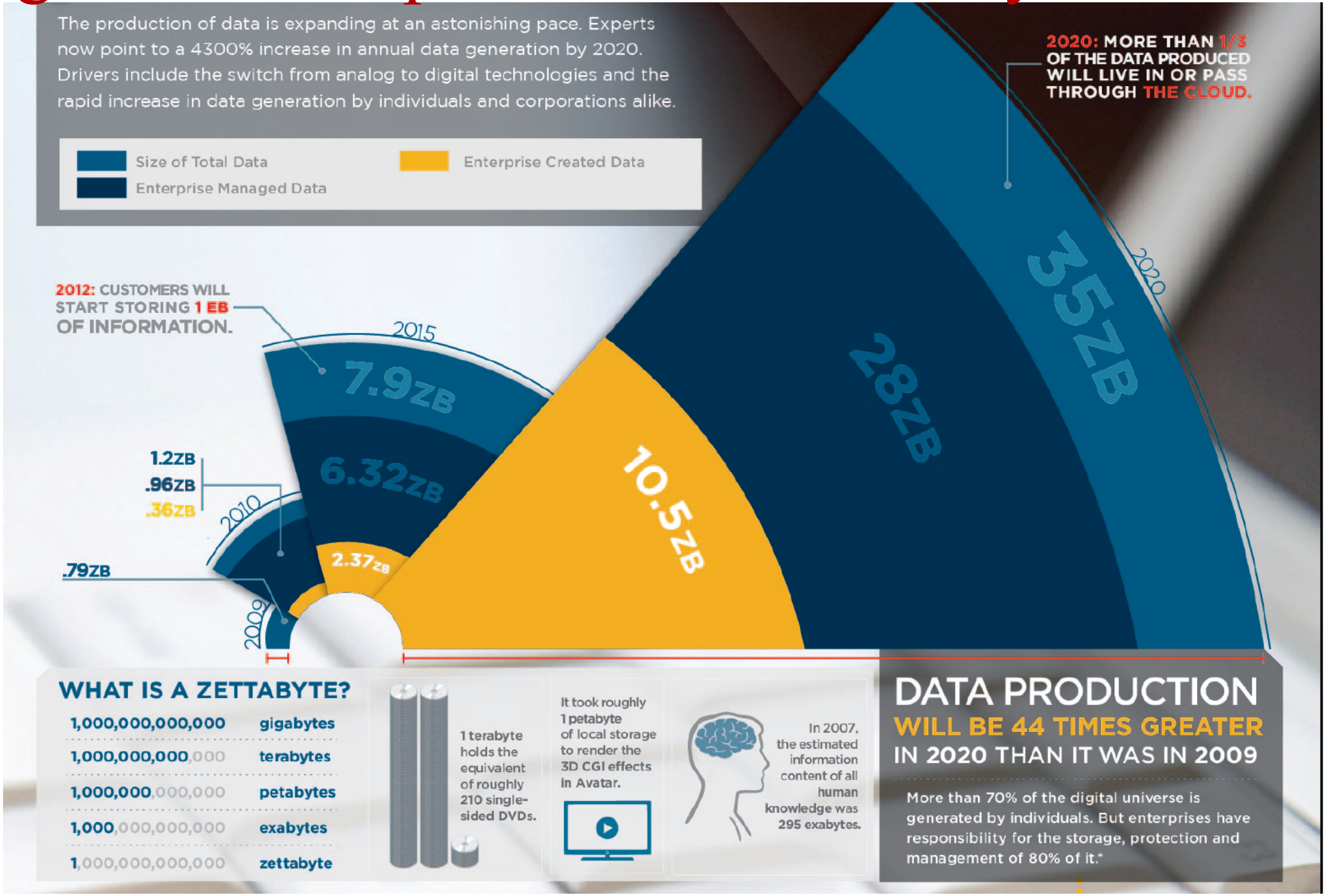
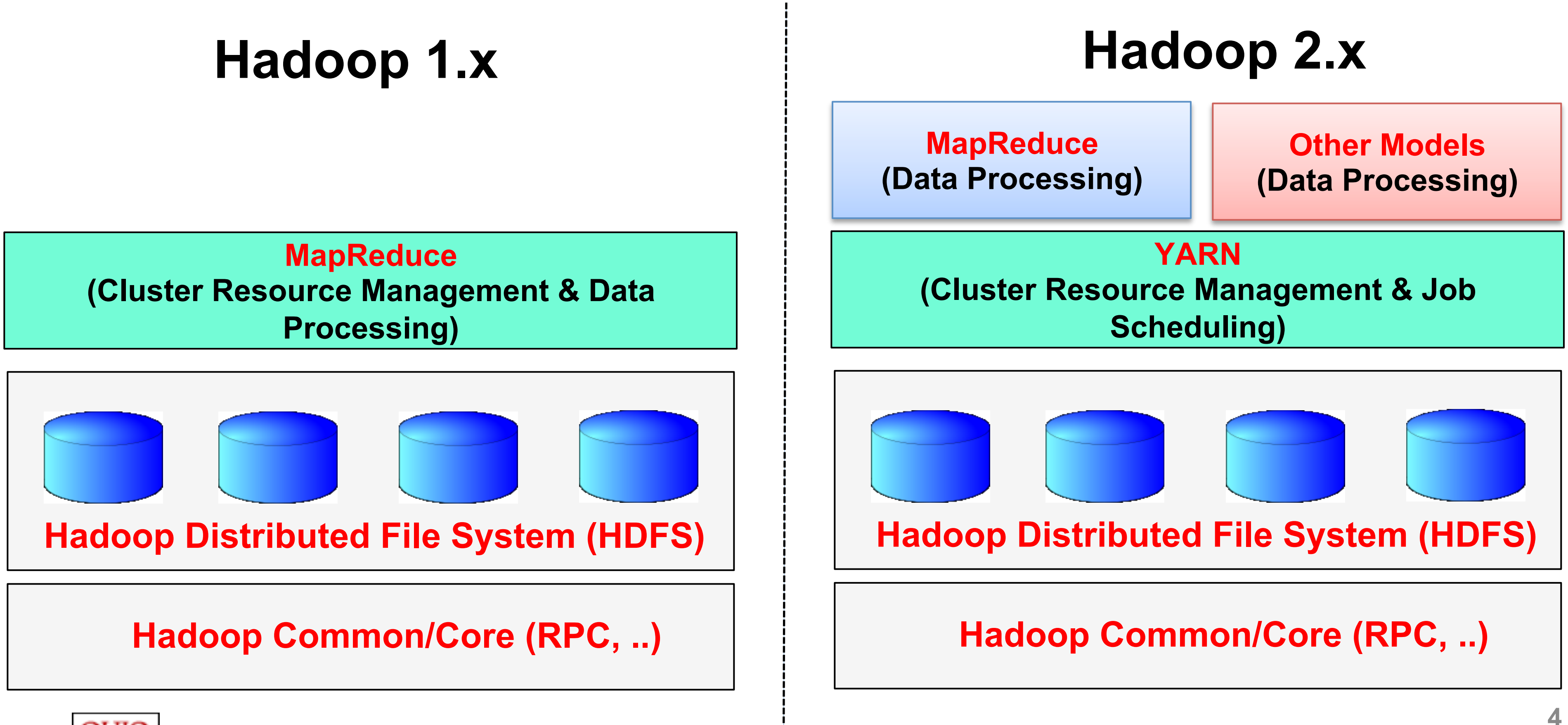


Figure Source (http://www.csc.com/insights/flxwd/78931-big_data_growth_just_beginning_to_explode)

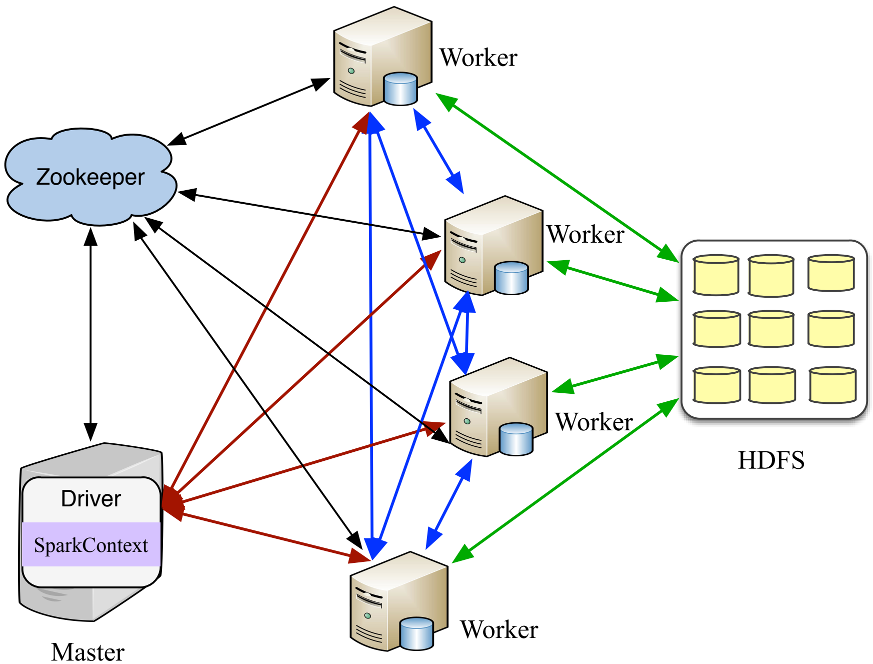
Big Data Technology - Hadoop

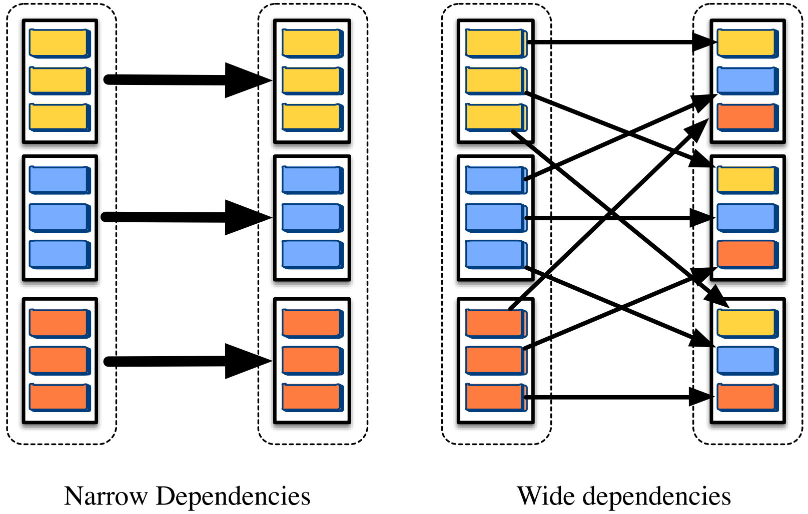
- Apache Hadoop is one of the most popular Big Data technology
  - Provides frameworks for large-scale, distributed data storage and processing
  - MapReduce, HDFS, YARN, RPC, etc.



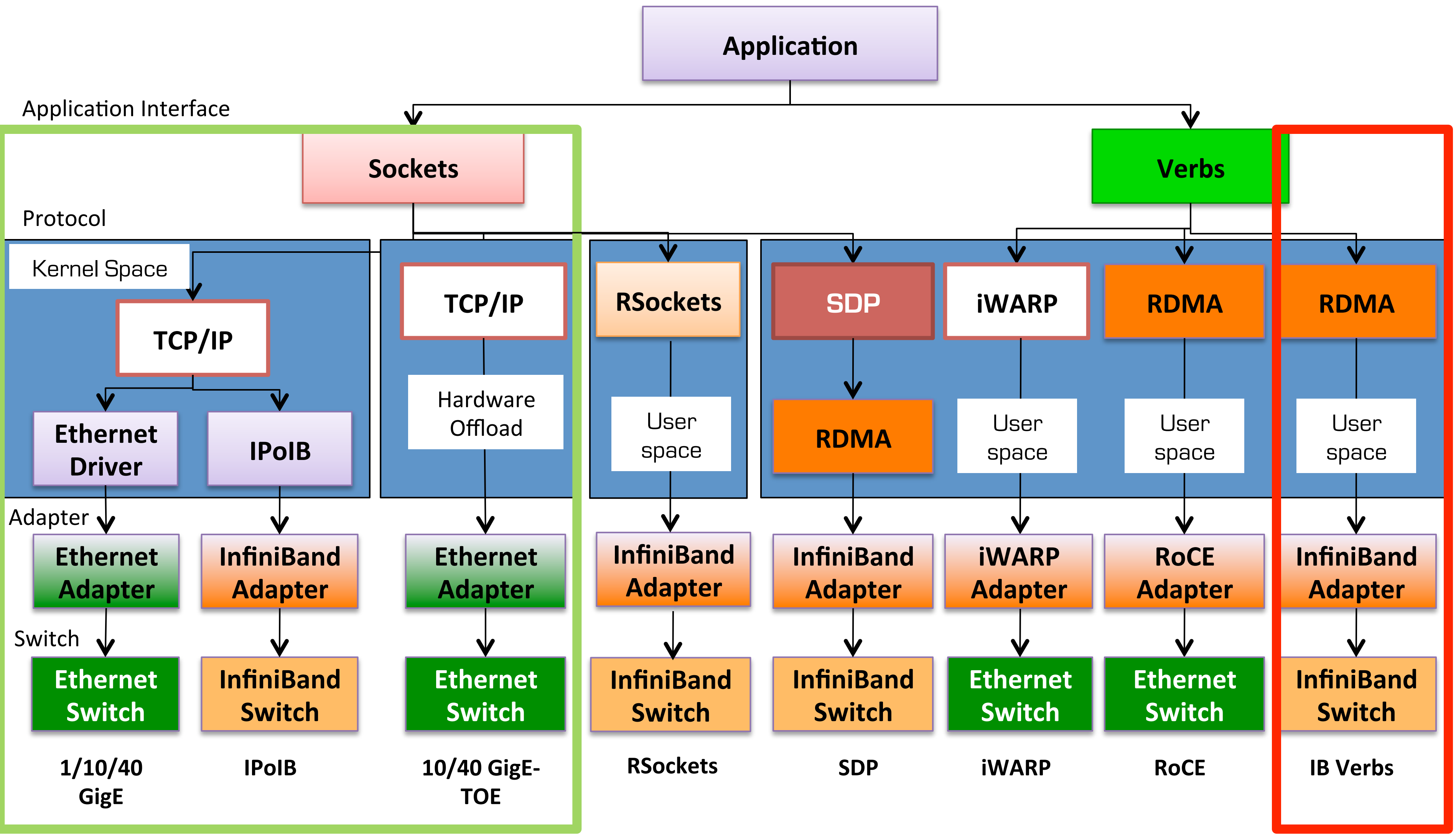
Big Data Technology - Spark

- An emerging in-memory processing framework
  - Iterative machine learning
  - Interactive data analytics
  - Scala based Implementation
  - Master-Slave; HDFS, Zookeeper
- Scalable and communication intensive
  - Wide dependencies between Resilient Distributed Datasets (RDDs)
  - MapReduce-like shuffle operations to repartition RDDs
  - Same as Hadoop, Sockets-based communication





Common Protocols using Open Fabrics



Previous Studies

- Very good performance improvements for Hadoop (HDFS /MapReduce/RPC), HBase, Memcached over InfiniBand
  - 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
  - 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., 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

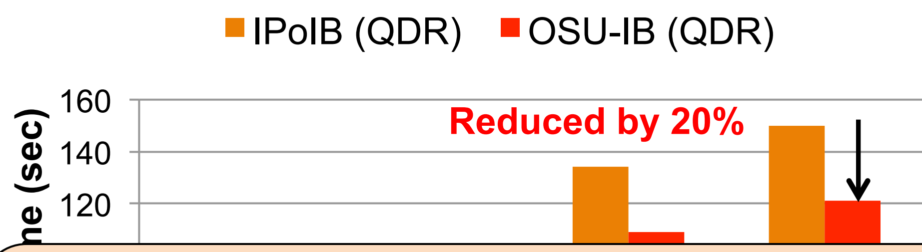
RDMA for Apache Hadoop

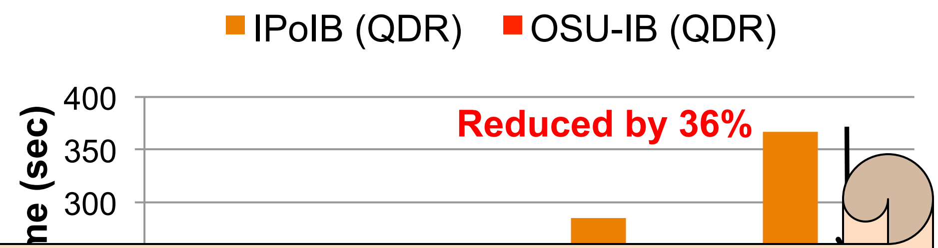
- High-Performance Design of Hadoop over RDMA-enabled Interconnects
  - High performance design with native InfiniBand and RoCE support at the verbs-level for HDFS, MapReduce, and RPC components
  - Easily configurable for native InfiniBand, RoCE and the traditional sockets -based support (Ethernet and 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

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

- RDMA for Apache Hadoop 2.x 0.9.1 is released in HiBD!

Performance Benefits – RandomWriter & Sort in SDSC-Gordon





Can **RDMA** benefit **Apache Spark** on **High-Performance Networks** also?

- **16%** improvement over IPoIB for 50GB in a cluster of 16 nodes
- **20%** improvement over IPoIB for 300GB in a cluster of 64 nodes

- 20% improvement over IPoIB for 50GB in a cluster of 16 nodes
- **36%** improvement over IPoIB for 300GB in a cluster of 64 nodes

Outline

- Introduction
- Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future work

Problem Statement

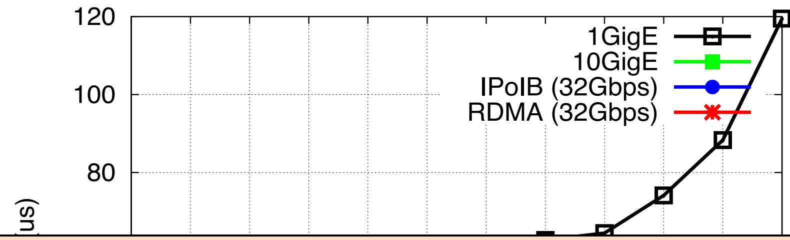
- Is it worth it?
  - Is the performance improvement potential high enough, if we can successfully adapt RDMA to Spark?
  - A few percentage points or orders of magnitude?
- How difficult is it to adapt RDMA to Spark?
  - Can RDMA be adapted to suit the communication needs of Spark?
  - Is it viable to have to rewrite portions of the Spark code with RDMA?
- Can Spark applications benefit from an RDMA-enhanced design?
  - What are the performance benefits that can be achieved by using RDMA for Spark applications on modern HPC clusters?
  - Can RDMA-based design benefit applications transparently?

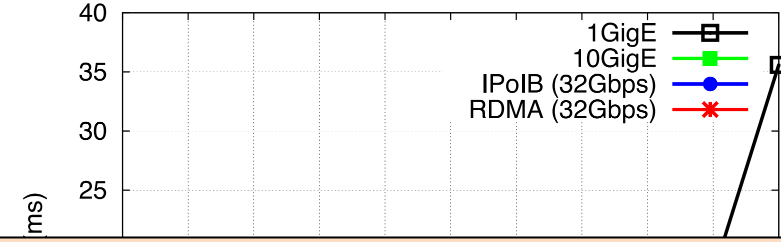
Assessment of Performance Improvement Potential

- How much benefit RDMA can bring in Apache Spark compared to other interconnects/protocols?

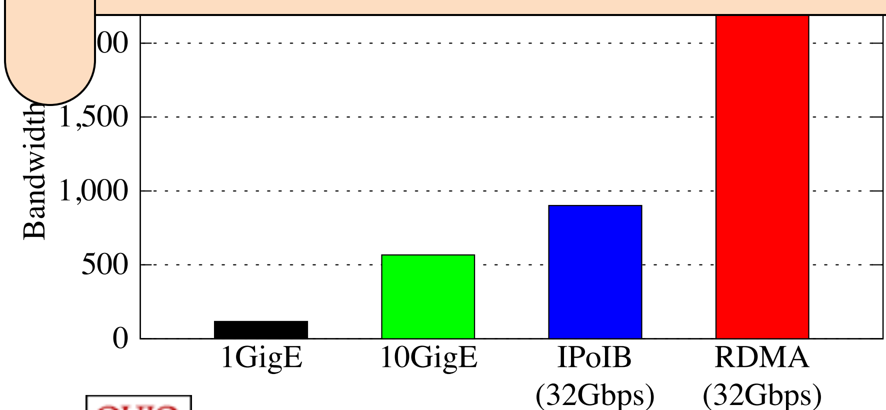
- Assessment Methodology
  – Evaluation on primitive-level micro-benchmarks
    - Latency, Bandwidth
    - 1GigE, 10GigE, IPoIB (32Gbps), RDMA (32Gbps)
    - Java/Scala-based environment
  – Evaluation on typical workloads
    - *GroupByTest*
    - 1GigE, 10GigE, IPoIB (32Gbps)
    - Spark 0.9.1

Evaluation on Primitive-level Micro-benchmarks



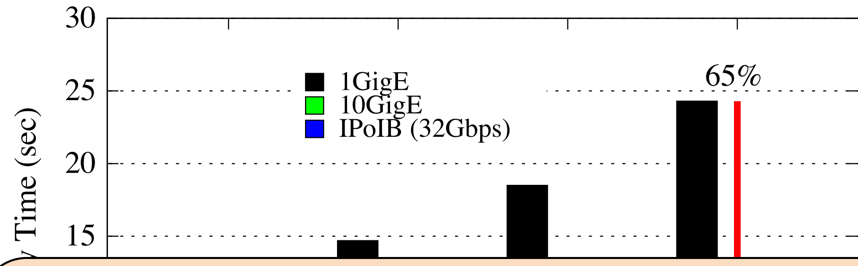


Can these benefits of High-Performance Networks be achieved in Apache Spark?



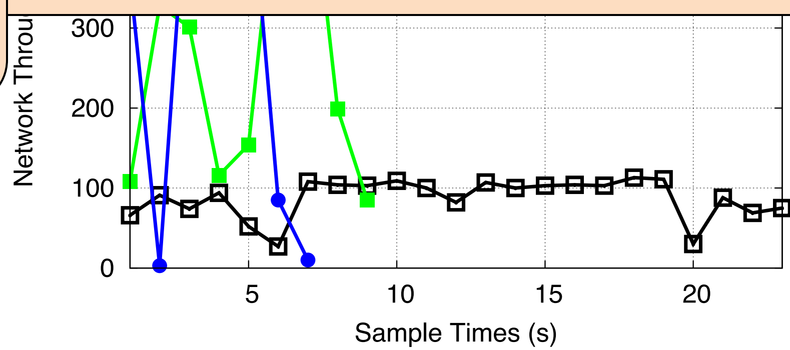
- Compared to other interconnects /protocols, RDMA can significantly
  - reduce the latencies for all the message sizes
  - improve the peak bandwidth

Evaluation on Typical Workloads

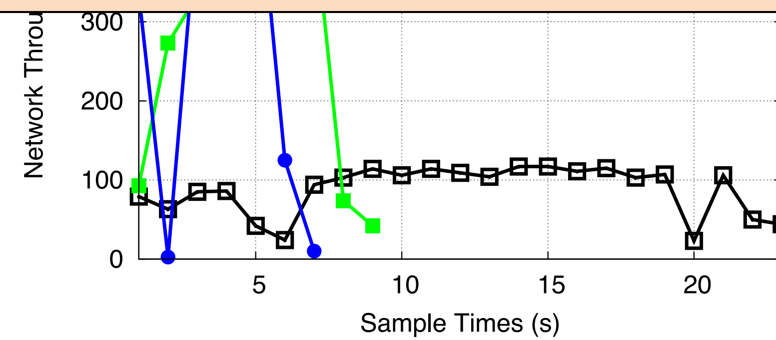


- For High-Performance Networks,
  - The execution time of GroupBy is significantly improved
  - The network throughputs are much

Can RDMA further benefit Spark performance compared with IPoIB and 10GigE?



Network Throughput in Recv



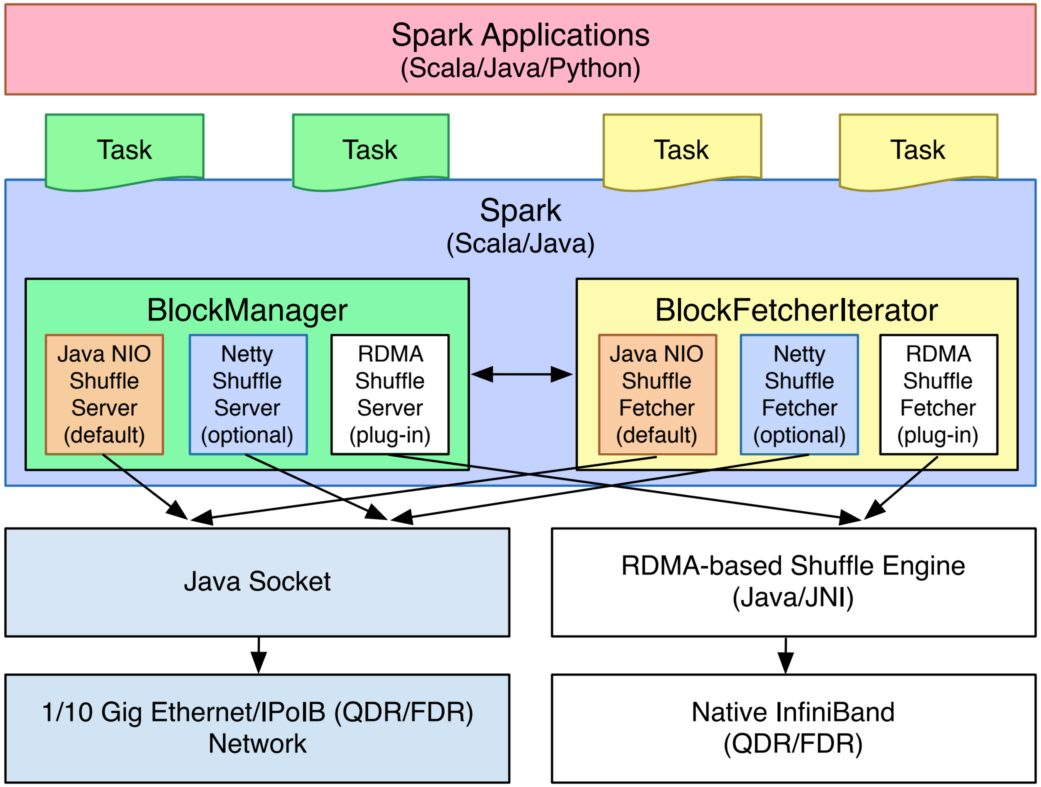
Network Throughput in Send

Outline

- Introduction
- Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future work

Architecture Overview

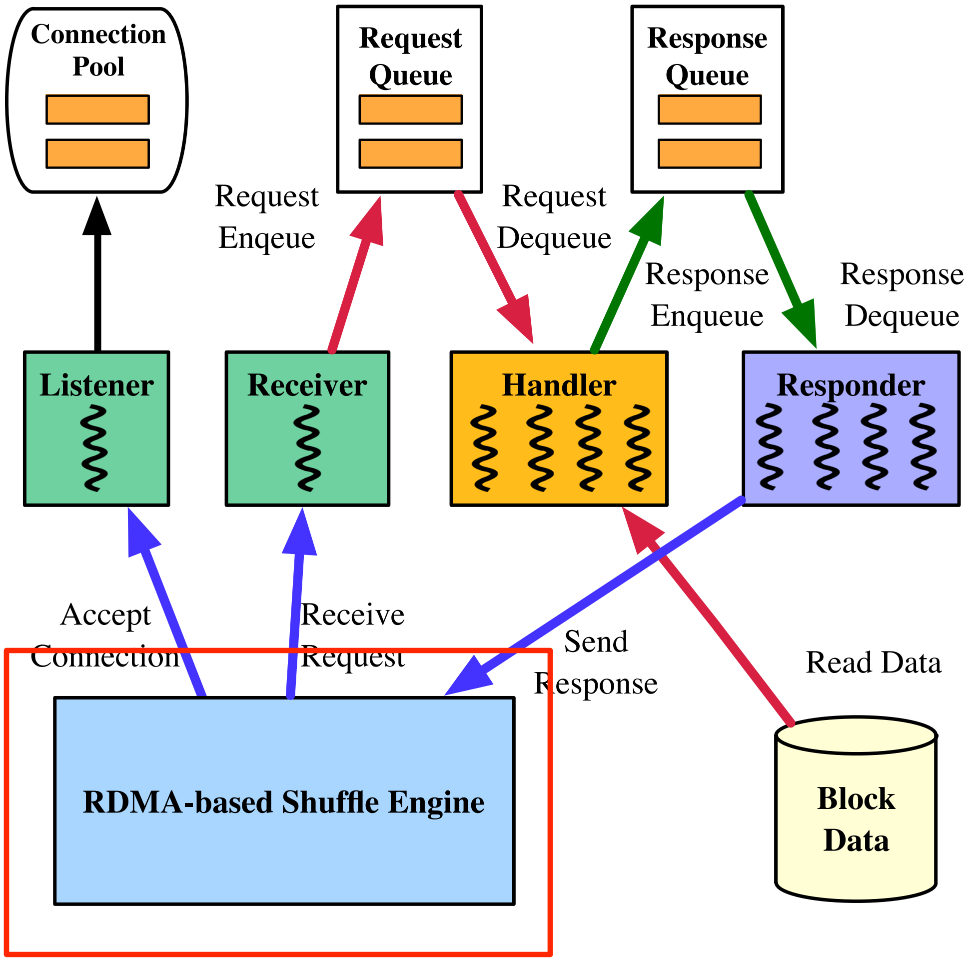
- Design Goals
  - High Performance
  - Keeping the existing Spark architecture and interface intact
  - Minimal code changes



- Approaches
  - Plug-in based approach to extend shuffle framework in Spark
    - RDMA Shuffle Server + RDMA Shuffle Fetcher
    - **100 lines of code changes inside Spark original files**
  - RDMA-based Shuffle Engine

SEDA-based Data Shuffle Plug-ins

- SEDA - Staged Event-Driven Architecture
- A set of stages connected by queues
- A dedicated thread pool will be in charge of processing events on the corresponding queue
  - Listener, Receiver, Handlers, Responders
- Performing admission controls on these event queues
- High throughput through maximally overlapping different processing stages as well as maintain default task-level parallelism in Spark



RDMA-based Shuffle Engine

- Connection Management
  - Alternative designs
    - Pre-connection
      - Hide the overhead in the initialization stage
      - Before the actual communication, pre-connect processes to each other
      - Sacrifice more resources to keep all of these connections alive
    - Dynamic Connection Establishment and Sharing
      - A connection will be established if and only if an actual data exchange is going to take place
      - A naive dynamic connection design is not optimal, because we need to allocate resources for every data block transfer
      - Advanced dynamic connection scheme that reduces the number of connection establishments
      - Spark uses multi-threading approach to support multi-task execution in a single JVM → Good chance to share connections!
      - How long should the connection be kept alive for possible re-use?
      - Time out mechanism for connection destroy

RDMA-based Shuffle Engine

- Data Transfer
  - Each connection is used by multiple tasks (threads) to transfer data concurrently
  - Packets over the same communication lane will go to different entities in both server and fetcher sides
  - Alternative designs
    - Perform sequential transfers of complete blocks over a communication lane → Keep the order
      - Cause long wait times for some tasks that are ready to transfer data over the same connection
    - Non-blocking and Out-of-order Data Transfer
      - Chunking data blocks
      - Non-blocking sending over shared connections
      - Out-of-order packet communication
      - Guarantee both performance and ordering
      - Efficiently work with the dynamic connection management and sharing mechanism

RDMA-based Shuffle Engine

- Buffer Management
  - On-JVM-Heap vs. Off-JVM-Heap Buffer Management
  - Off-JVM-Heap
    - High-Performance through native IO
    - Shadow buffers in Java/Scala
    - Registered for RDMA communication
  - Flexibility for upper layer design choices
    - Support connection sharing mechanism → Request ID
    - Support packet processing in order → Sequence number
    - Support non-blocking send → Buffer flag + callback

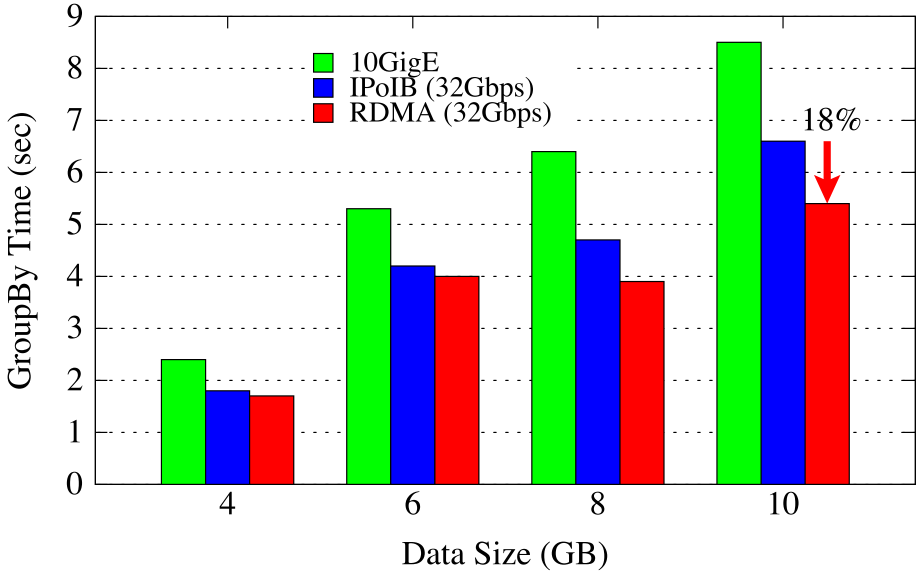
Outline

- Introduction
- Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future work

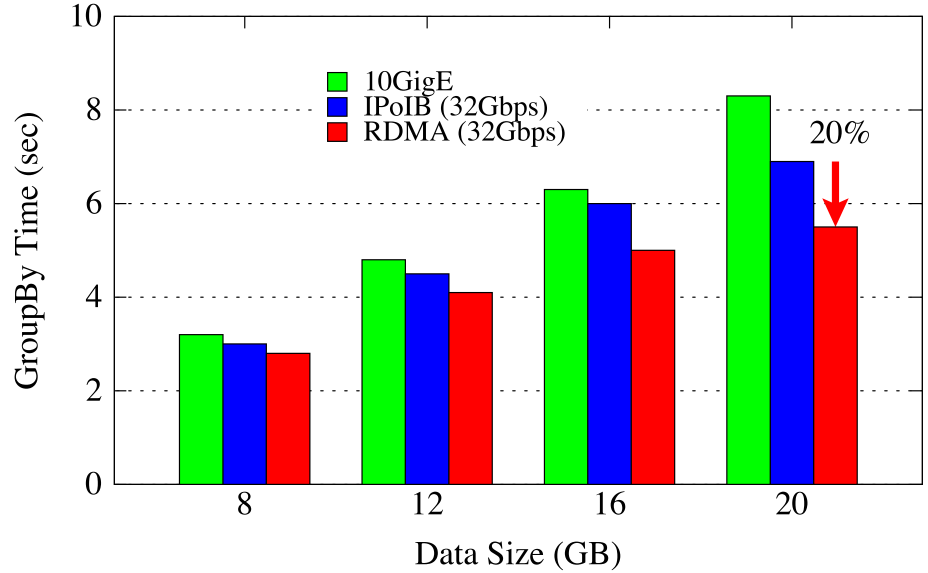
Experimental Setup

- 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 (32Gbps) + 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 (56Gbps)
- Software
  - Spark 0.9.1, Scala 2.10.4 and JDK 1.7.0
  - GroupBy Test

Performance Evaluation on Cluster A



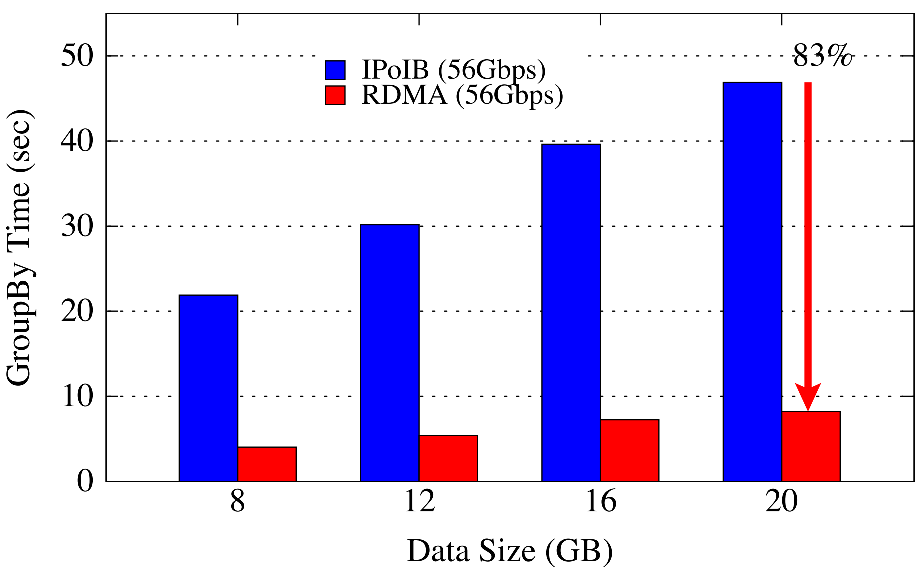
4 Worker Nodes, 32 Cores, (32M 32R)



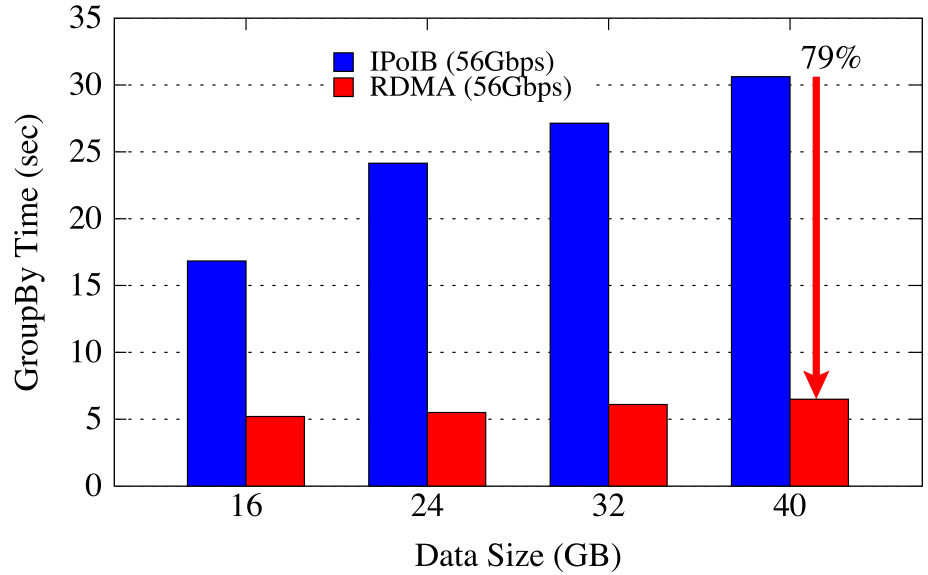
8 Worker Nodes, 64 Cores, (64M 64R)

- For 32 cores, up to 18% over IPoIB (32Gbps) and up to 36% over 10GigE
- For 64 cores, up to 20% over IPoIB (32Gbps) and 34% over 10GigE

Performance Evaluation on Cluster B



8 Worker Nodes, 128 Cores, (128M 128R)



16 Worker Nodes, 256 Cores, (256M 256R)

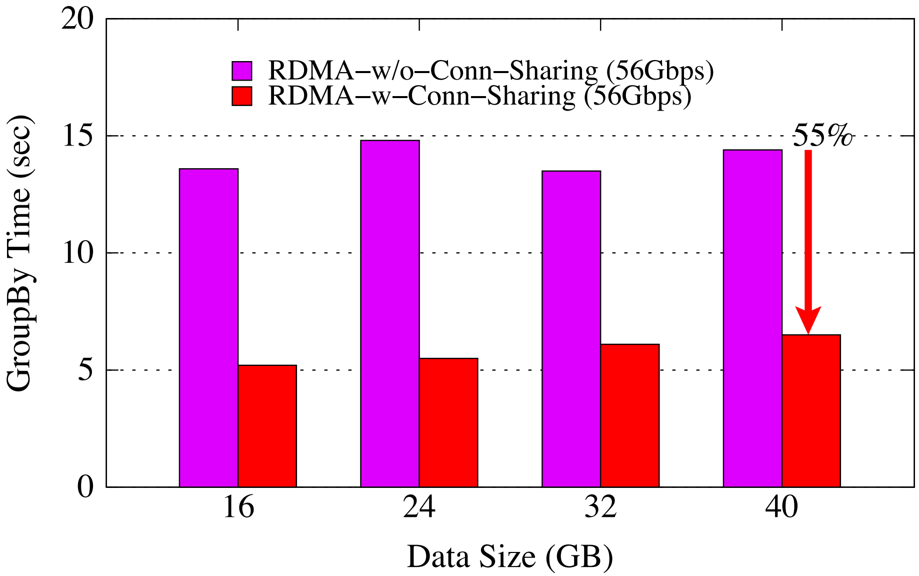
- For 128 cores, up to 83% over IPoIB (56Gbps)
- For 256 cores, up to 79% over IPoIB (56Gbps)

Performance Analysis on Cluster B

- Java-based micro-benchmark comparison

	IPoIB (56Gbps)	RDMA (56Gbps)
Peak Bandwidth	1741.46MBps	5612.55MBps

- Benefit of RDMA Connection Sharing Design



By enabling connection sharing, achieve 55% performance benefit for 40GB data size on 16 nodes

Outline

- Introduction
- Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future work

Conclusion and Future Work

- **Three major conclusions**
  - **RDMA and high-performance interconnects can benefit Spark.**
  - **Plug-in based approach with SEDA-/RDMA-based designs provides both performance and productivity.**
  - **Spark applications can benefit from an RDMA -enhanced design.**
- Future Work
  - Continuously update this package with newer designs and carry out evaluations with more Spark applications on systems with high-performance networks
  - Will make this design publicly available through the HiBD project

Thank You!

{luxi, rahmanmd, islamn, shankard, 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/