High Performance RDMA-based Design of HDFS over InfiniBand

N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy* & D. K. Panda

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

**IBM TJ Watson Research Center*
Yorktown Heights, NY, USA

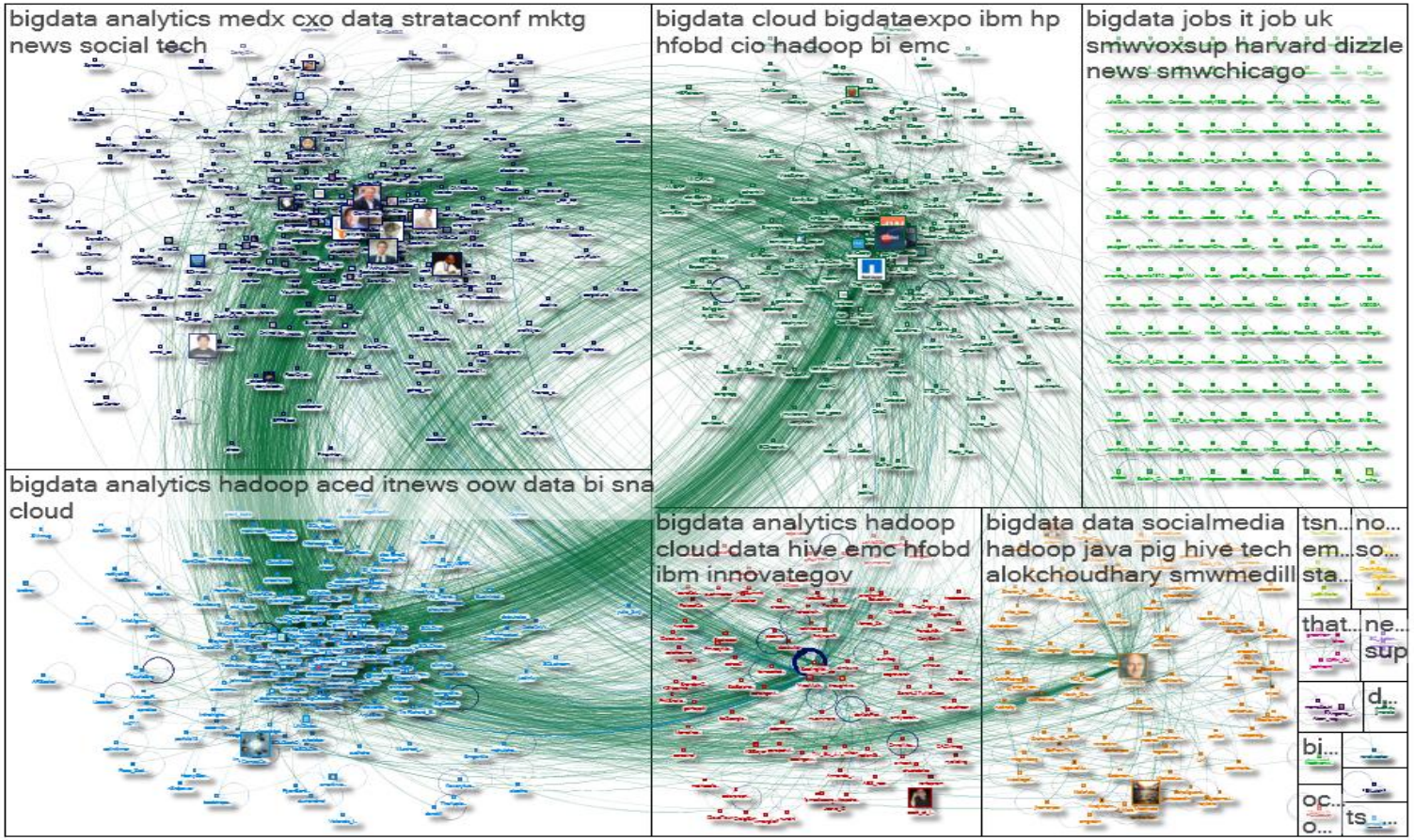
Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
- Conclusion & Future work

Introduction

- **Big Data**: provides groundbreaking opportunities for enterprise information management and decision making
- The rate of information growth appears to be exceeding Moore's Law
- The amount of data is exploding; companies are capturing and digitizing more information that ever
- **35 zettabytes** of data will be generated and consumed by the end of this decade

Introduction



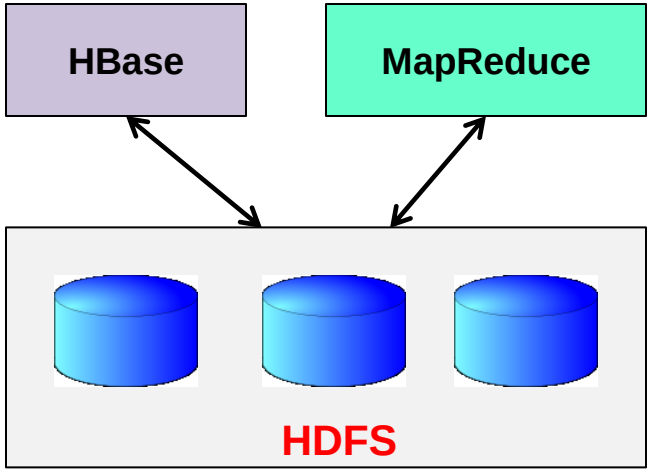
Graph of 1026 Twitter users whose tweets contain 'big data'

https://nodexlgraphgallery.org/Pages/Graph.aspx?graphID=1266

Tweets made over a period of 8 hours and 24 minutes

Big Data Technology

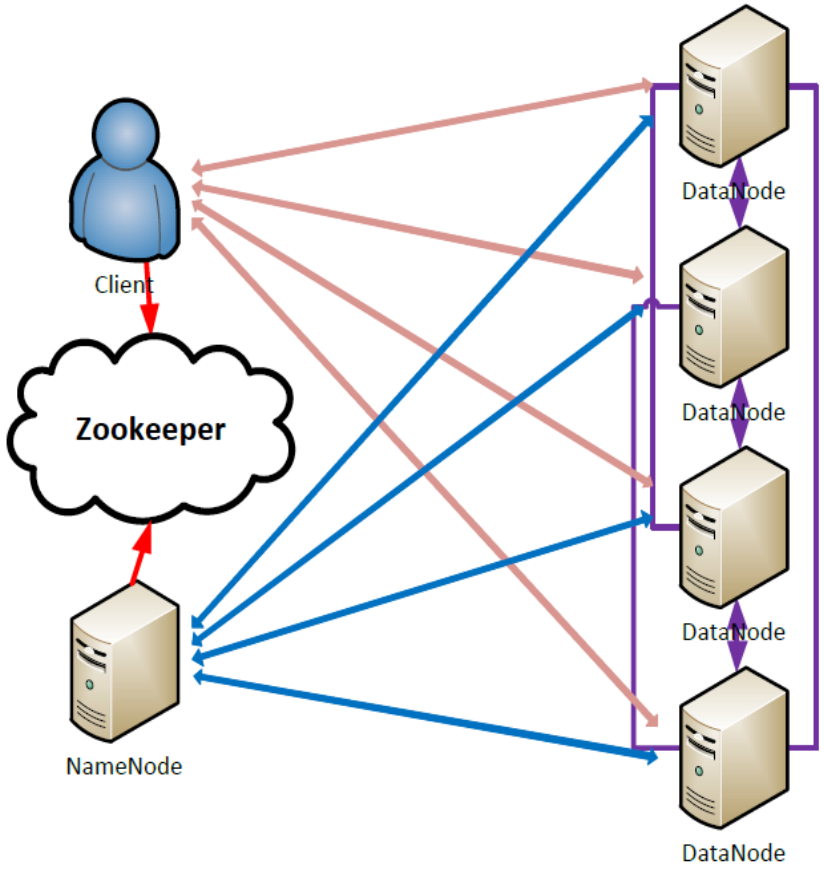
- Apache Hadoop is a popular Big Data technology
  - Provides framework for large-scale, distributed data storage and processing
- Hadoop is an open-source implementation of MapReduce programming model
- Hadoop Distributed File System (HDFS) (http://hadoop.apache.org/) is the underlying file system of Hadoop and Hadoop DataBase, HBase



Hadoop Framework

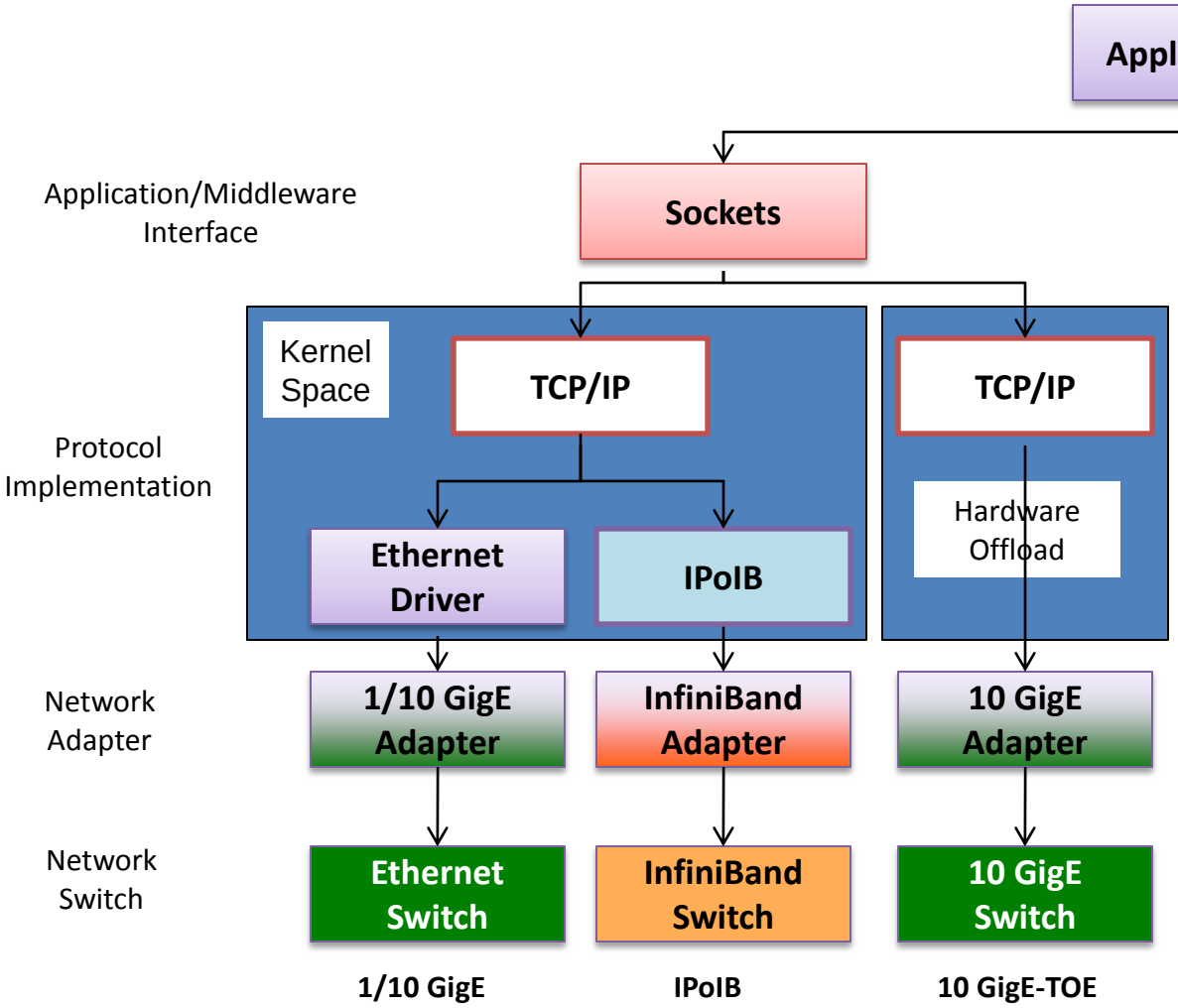
Hadoop Distributed File System (HDFS)

- Adopted by many reputed organizations
  - eg:- Facebook, Yahoo!
- Highly reliable and fault-tolerant - replication
- NameNode: stores the file system namespace
- DataNode: stores data blocks
- Developed in Java for platform-independence and portability
- Uses Java sockets for communication



(HDFS Architecture)

Modern High Performance Interconnects
(Socket Interface)



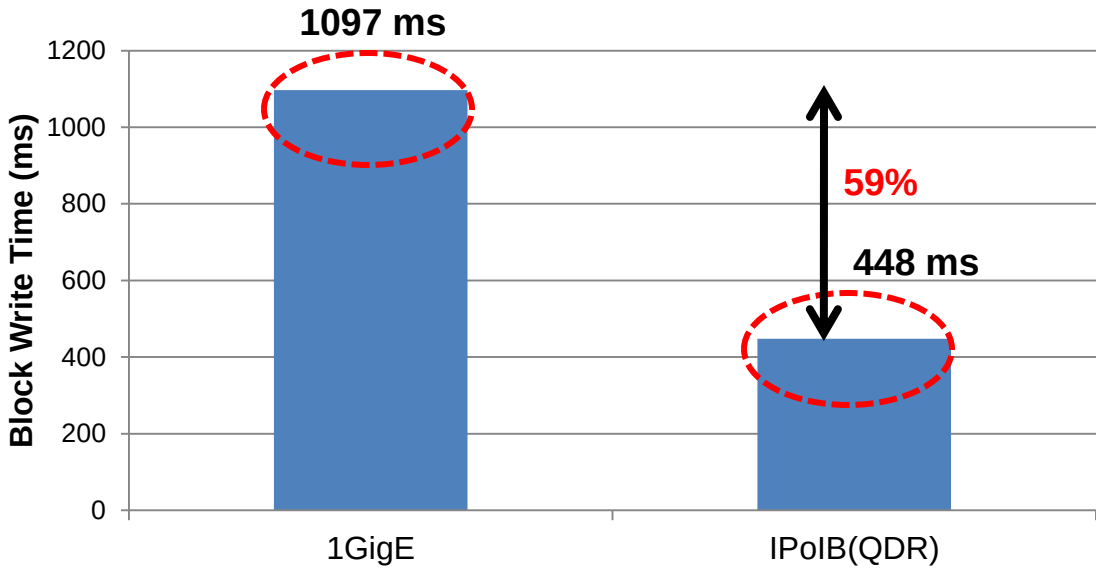
Cloud Computing systems are being widely used on High Performance Computing (HPC) Clusters

Hadoop middleware components do not leverage HPC cluster features for communication

Commodity high performance networks like InfiniBand can provide low latency and high throughput data transmission

For data-intensive applications network performance becomes key component for HDFS

HDFS Block Write Time



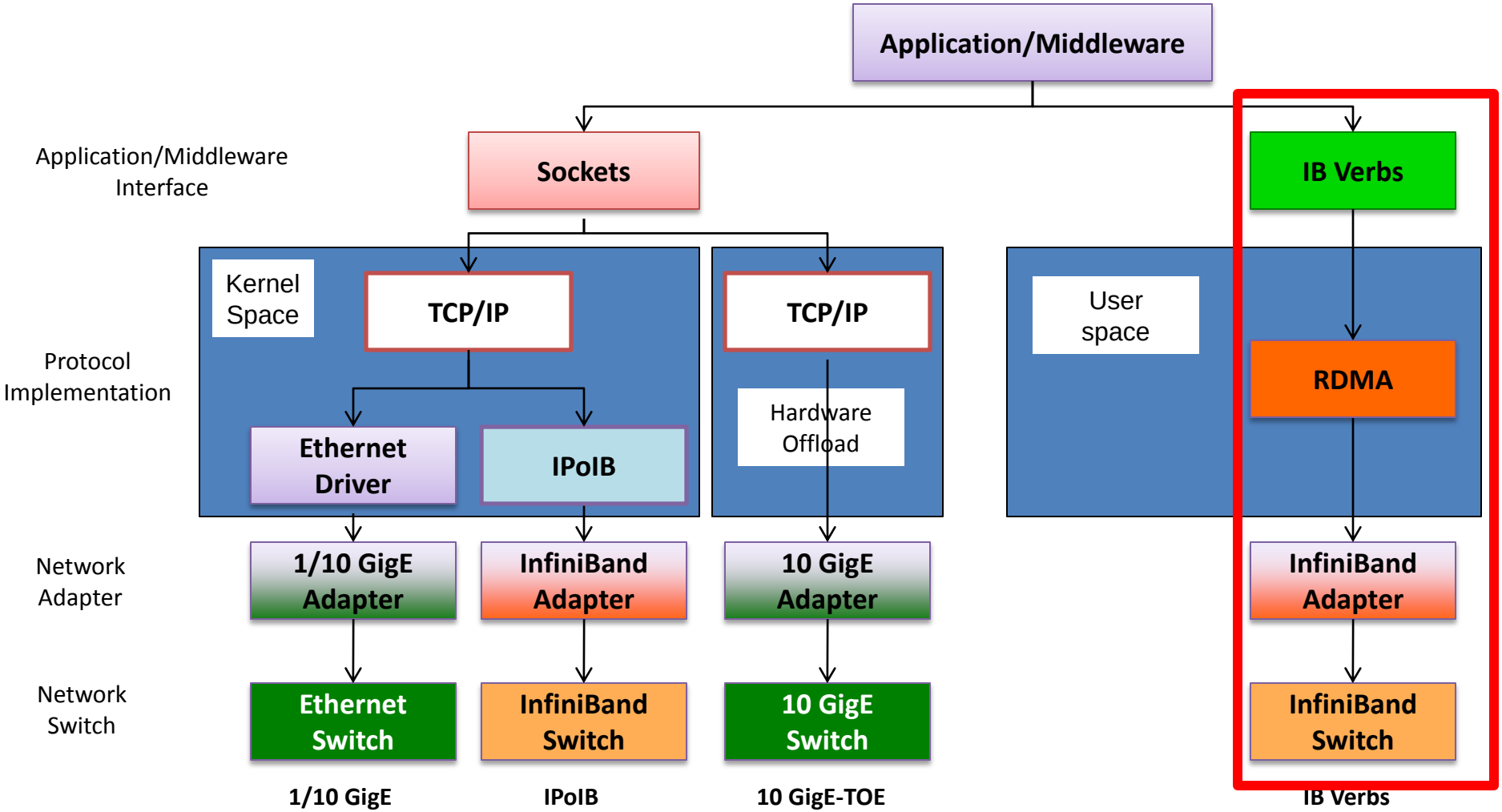
4 DataNodes each with single HDD
Block size = 64MB
Replication factor = 3

- HDFS block write time:
  - **1097** ms for 1GigE
  - **448** ms for IPoIB (QDR) (TCP/IP emulation over IB)
- IPoIB (QDR) improves the write time by **59%**

The byte stream communication nature of TCP/IP requires multiple data copies

Can we improve further?

Modern High Performance Interconnects



Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
- Conclusion & Future work

Problem Statement

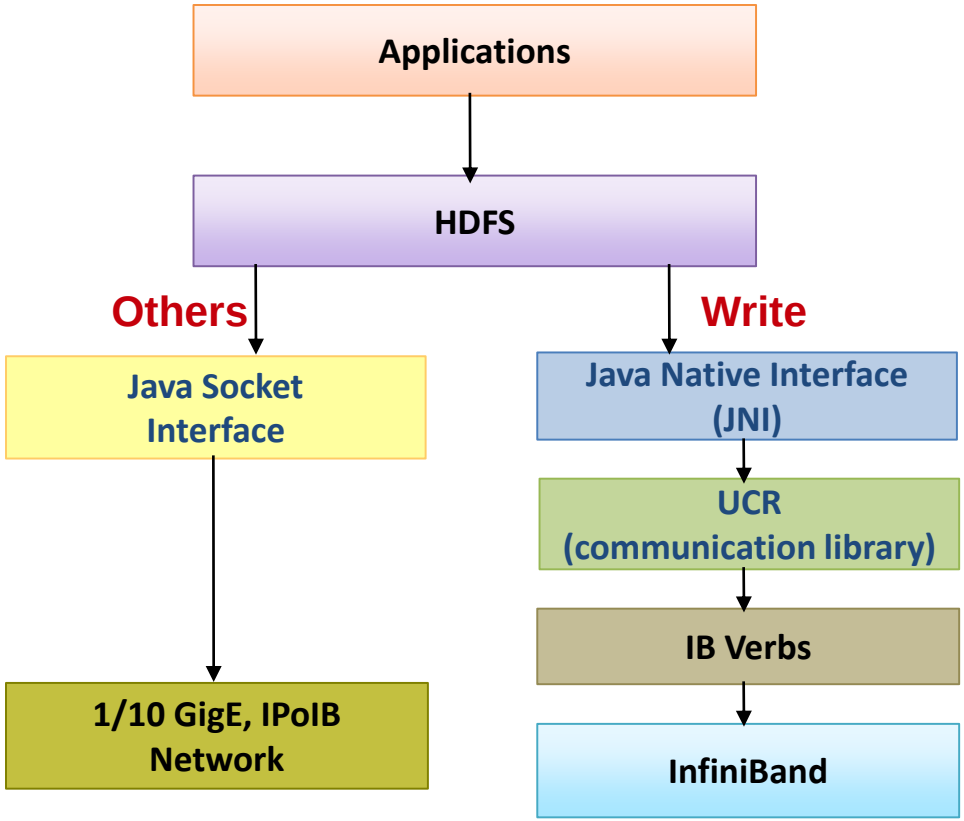
- How does the network performance impact the overall HDFS performance?
- Can we re-design HDFS to take advantage of high performance networks and exploit advanced features such as RDMA?
- What will be the performance improvement of HDFS with the RDMA-based design over InfiniBand?
- Can we observe the performance improvement for other cloud computing middleware such as HBase?

Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
- Conclusion & Future Work

HDFS Hybrid Design Overview

Enables high performance RDMA communication, while supporting traditional socket interface



HDFS Write involves replication; more network intensive

HDFS Read is mostly node-local

- JNI Layer bridges Java based HDFS with communication library written in native code
- **Only the communication part of HDFS Write is modified; No change in HDFS architecture**

Unified Communication Runtime (UCR)

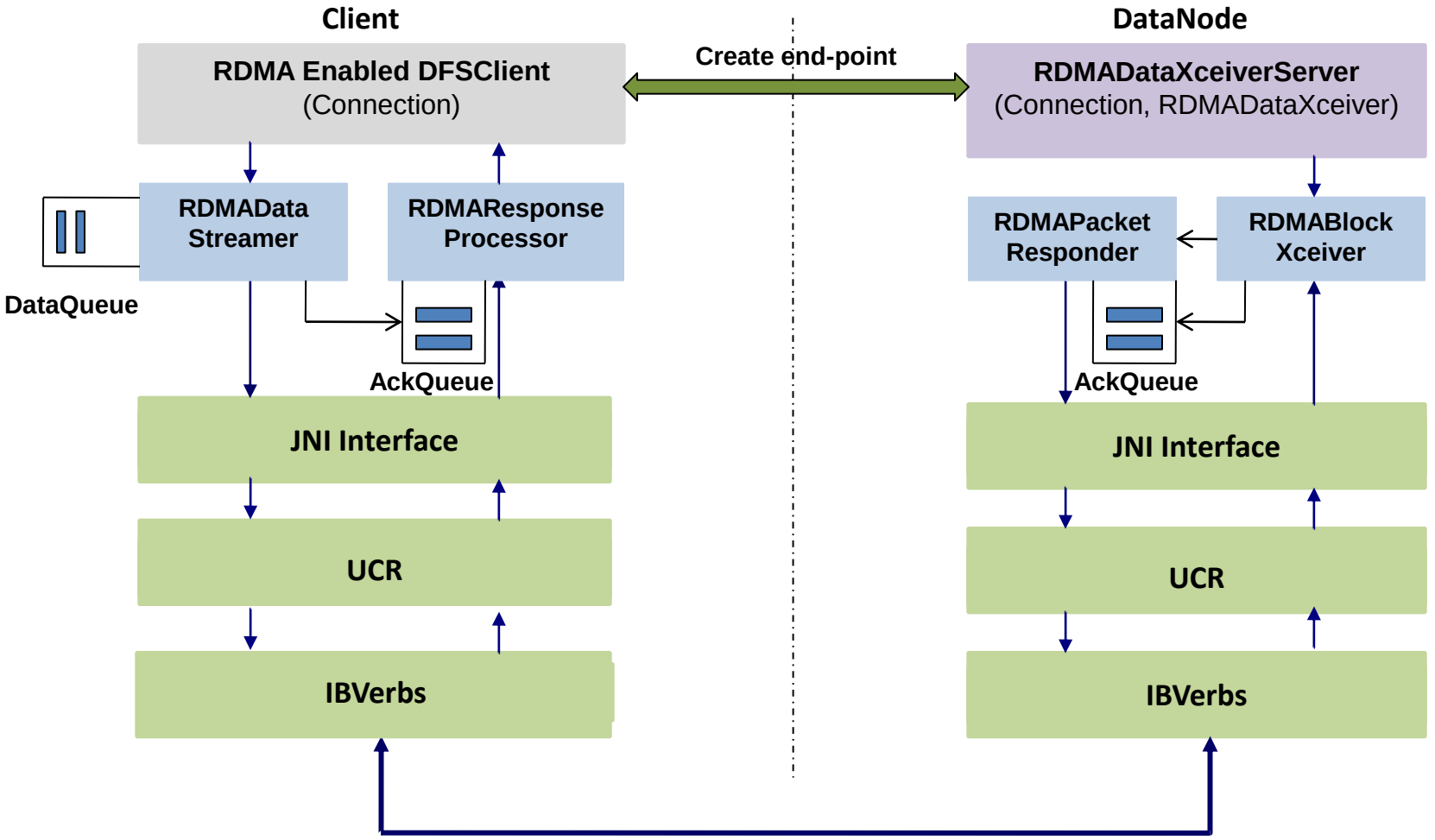
- Light-weight, high performance communication runtime
- Design of UCR evolved from MVAPICH/MVAPICH2 software stacks (http://mvapich.cse.ohio-state.edu/)
- Communications based on endpoints, analogous to sockets
- Designed as a native library to extract high performance from advanced network technologies
- Enhanced APIs to support data center middlewares such as HBase, Memcached, etc.

Design Challenges

- Socket-based HDFS
  - new socket connection for each block
  - New receiver thread per block in the DataNode
  - Multiple data copies

- RDMA-based HDFS
  - Creating a new UCR connection per block is expensive
  - Reduce data copy overhead
  - Keep low memory footprint

Components and Communication Flow



HDFS-JNI Interaction

- The new design does not change the user-level APIs
- Communication part of HDFS write API is modified keeping the socket-based design intact
- A new configuration parameter *dfs.ib.enabled* is added to select the communication protocol;
  - dfs.ib.enabled = true -> write will go over RDMA
  - dfs.ib.enabled = false -> write will go over socket

Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
- Conclusion & Future Work

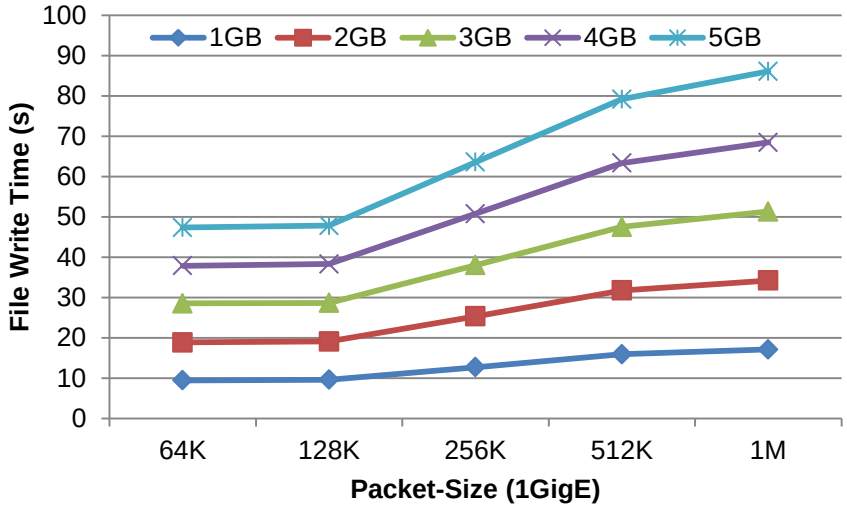
Experimental Setup

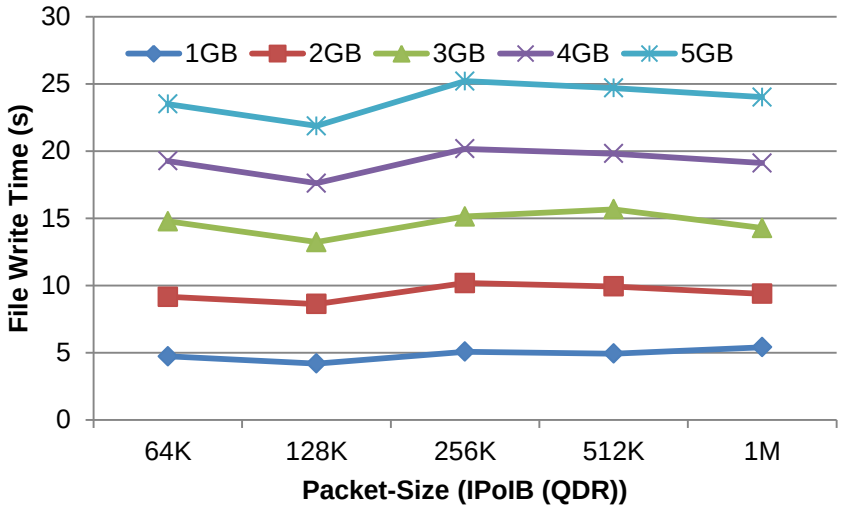
- Hardware
  - **Intel Clovertown (Cluster A)**
    - Each node has 8 processor cores on 2 Intel Xeon 2.33 GHz Quad-core CPUs, 6 GB main memory, 250 GB hard disk
    - Network: 1GigE, IPoIB, 10GigE TOE and IB-DDR (16Gbps)
  - **Intel Westmere (Cluster B)**
    - Each node has 8 processor cores on 2 Intel Xeon 2.67 GHz Quad-core CPUs, 12 GB main memory, 160 GB hard disk
    - 4 storage nodes with two 1 TB HDD per node, 24 GB RAM
    - 4 storage nodes with 300GB OCZ VeloDrive PCIe SSD
    - Network: 1GigE, IPoIB and IB-QDR (32Gbps)
- Software
  - Hadoop 0.20.2, HBase 0.90.3 and Sun Java SDK 1.7.
  - Yahoo! Cloud Serving Benchmark (YCSB)

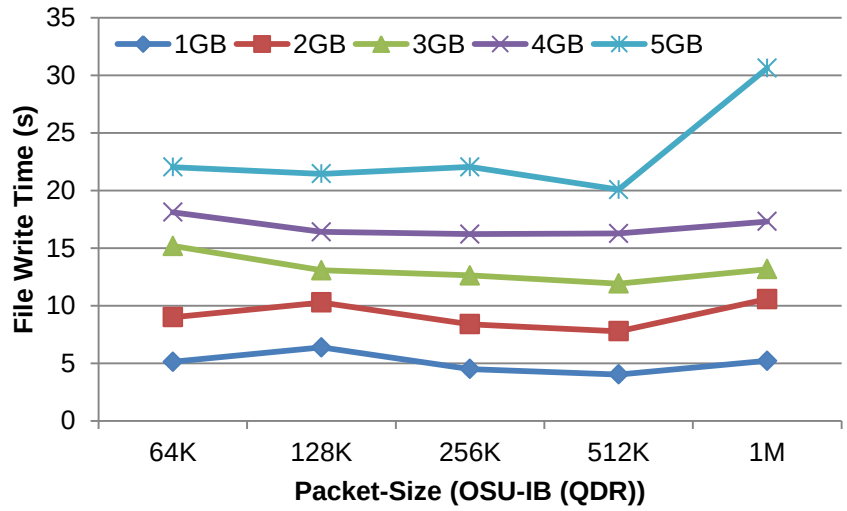
Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
  - Determining the optimal packet-size for different interconnects/protocols
  - Micro-benchmark level evaluations (measures the latency of file write)
  - Evaluations with TestDFSIO (measures the throughput of sequential file write)
  - Integration with HBase (TCP/IP) (measures latency and throughput of HBase put operation)
- Conclusion & Future Work

HDFS Optimal Packet-Size Evaluation





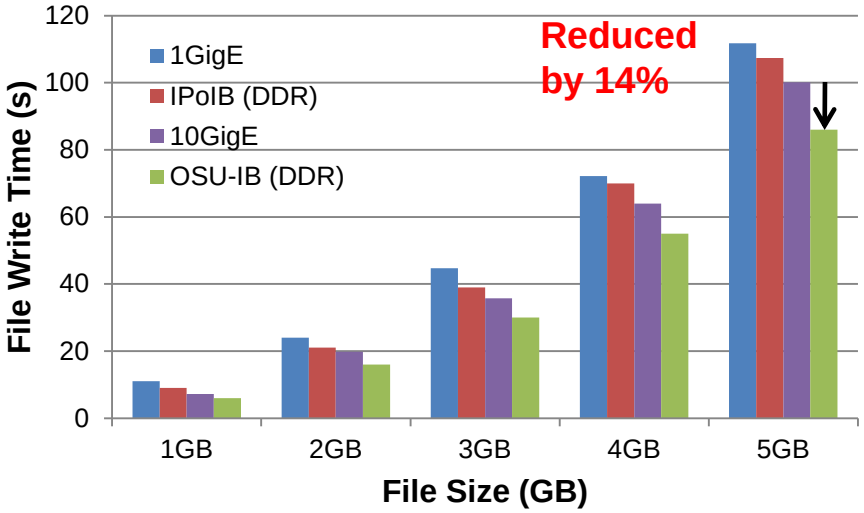


- For both the clusters (HDD/SSD)
  - Optimal packet-size for 1/10GigE: 64KB
  - Optimal packet-size for IPoIB: 128KB
  - Optimal packet-size for OSU-IB: 512KB

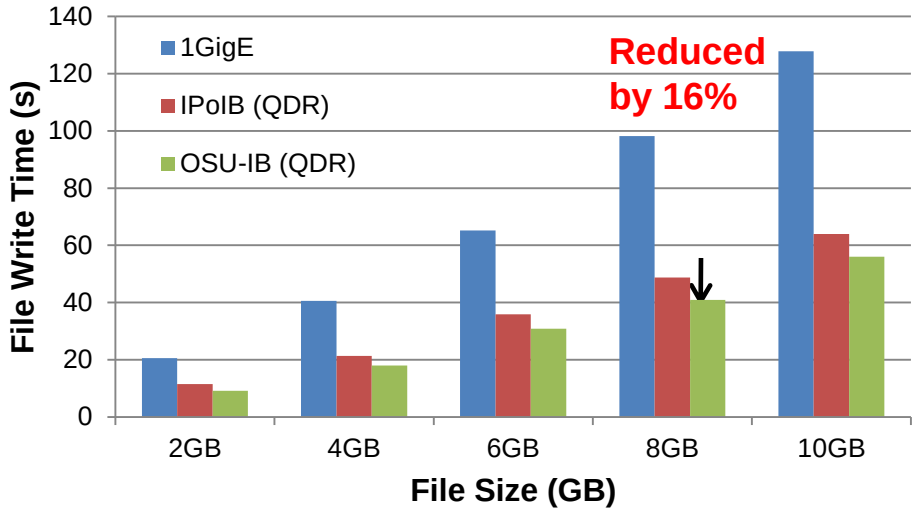
Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
  - Determining the optimal packet-size for different interconnects/protocols
  - Micro-benchmark level evaluations (measures the latency of file write)
  - Evaluations with TestDFSIO (measures the throughput of sequential file write)
  - Integration with HBase (TCP/IP) (measures latency and throughput of HBase put operation)
- Conclusion & Future Work

Evaluations using Micro-benchmark (DataNode Storage: HDD)



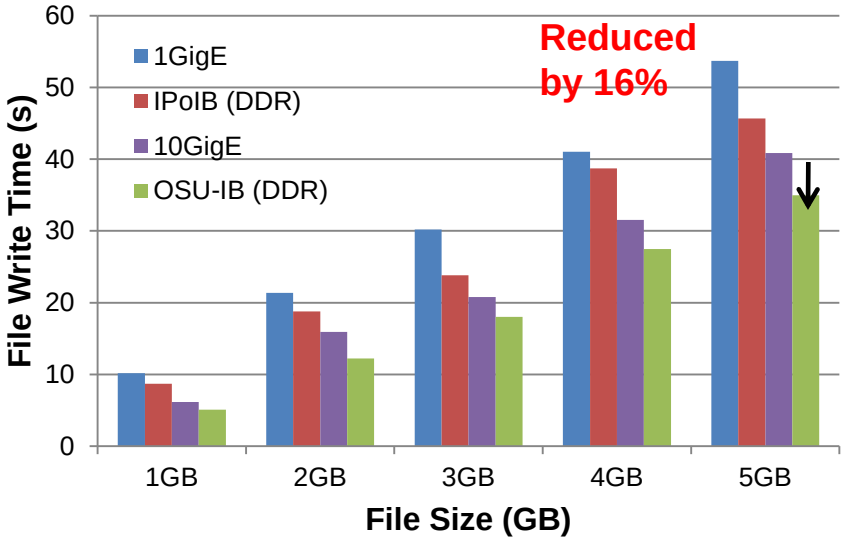
Cluster A with 4 DataNodes



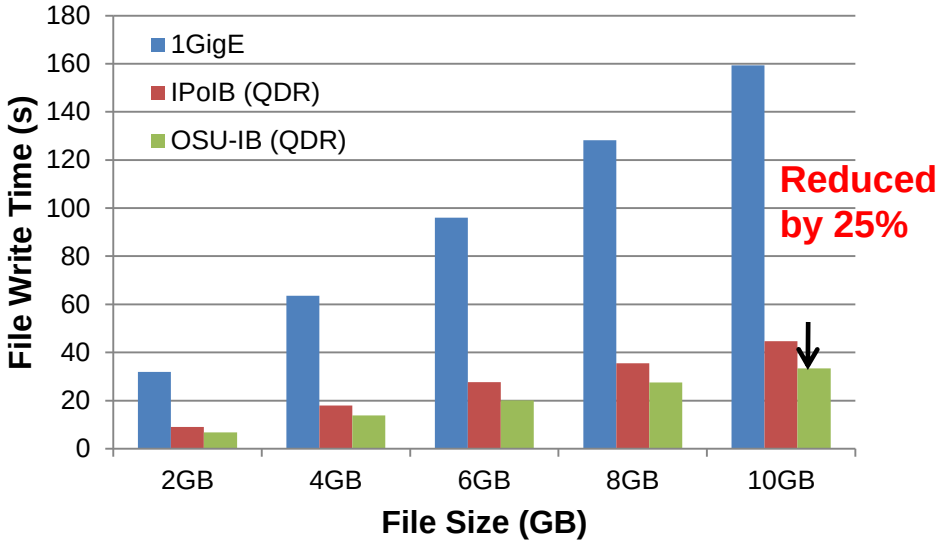
Cluster B with 32 DataNodes

- Cluster A with 4 DataNodes
  - **14%** improvement over 10GigE for 5 GB file size
  - **20%** improvement over IPoIB (16Gbps) for 5GB file size
- Cluster B with 32 DataNodes
  - **16%** improvement over IPoIB (32Gbps) for 8GB file size

Evaluations using Micro-benchmark (DataNode Storage: SSD)



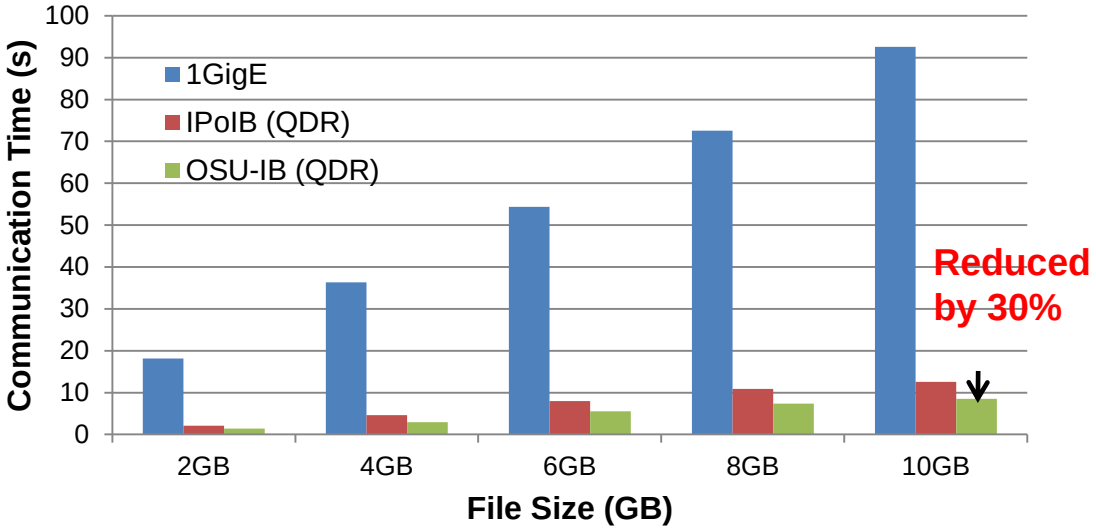
Cluster A with 4 DataNodes



Cluster B with 4 DataNodes

- Cluster A with 4 DataNodes
  - **16%** improvement over 10GigE for 5GB file size
  - **25%** improvement over IPoIB (16Gbps) for 5GB file size
- Cluster B with 4 DataNodes
  - **25%** improvement over IPoIB (32Gbps) for 10GB file size

Communication Times in HDFS



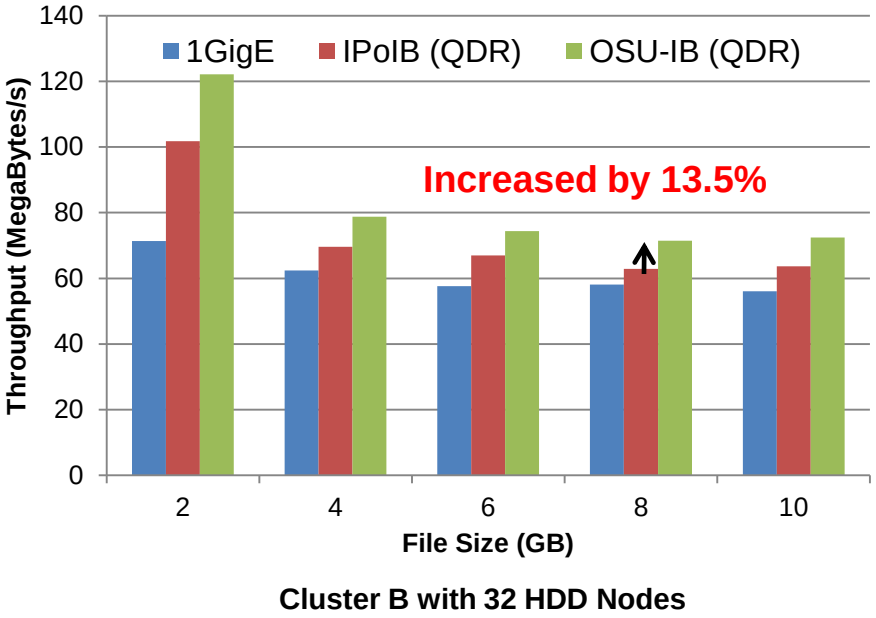
- Cluster B with 32 HDD DataNodes
  - **30%** improvement in communication time over IPoIB (32Gbps)
  - **87%** improvement in communication time over 1GigE
- Similar improvements are obtained for SSD DataNodes

Cluster B

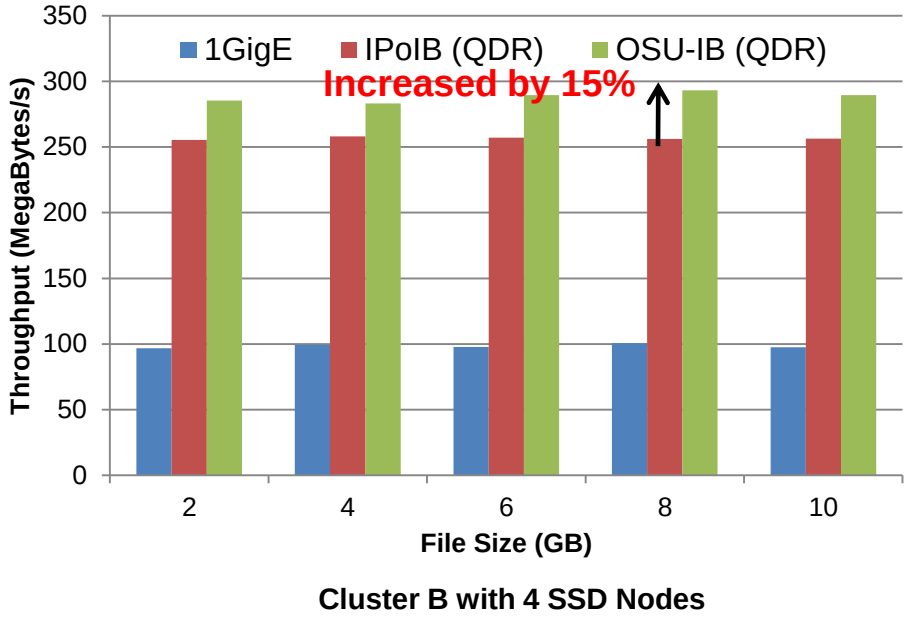
Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
  - Determining the optimal packet-size for different interconnects/protocols
  - Micro-benchmark level evaluations (measures the latency of file write)
  - Evaluations with TestDFSIO (measures the throughput of sequential file write)
  - Integration with HBase (TCP/IP) (measures latency and throughput of HBase put operation)
- Conclusion & Future Work

Evaluations using TestDFSIO



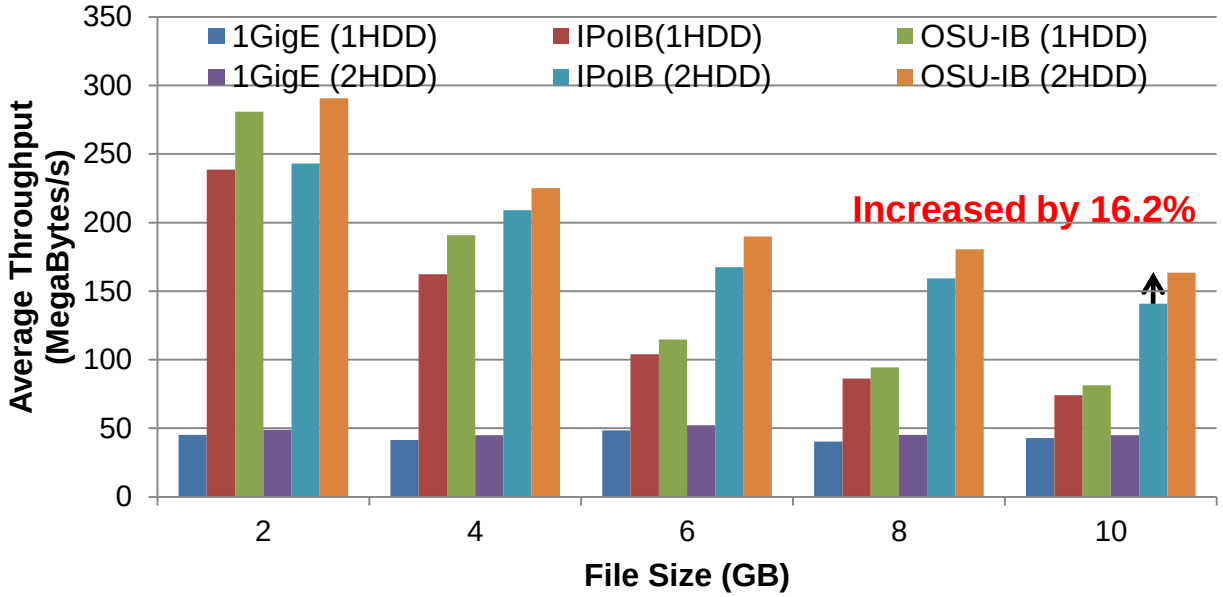
Cluster B with 32 HDD Nodes



Cluster B with 4 SSD Nodes

- Cluster B with 32 HDD DataNodes
  - **13.5%** improvement over IPoIB (32Gbps) for 8GB file size
- Cluster B with 4 SSD DataNodes
  - **15%** improvement over IPoIB (32Gbps) for 8GB file size

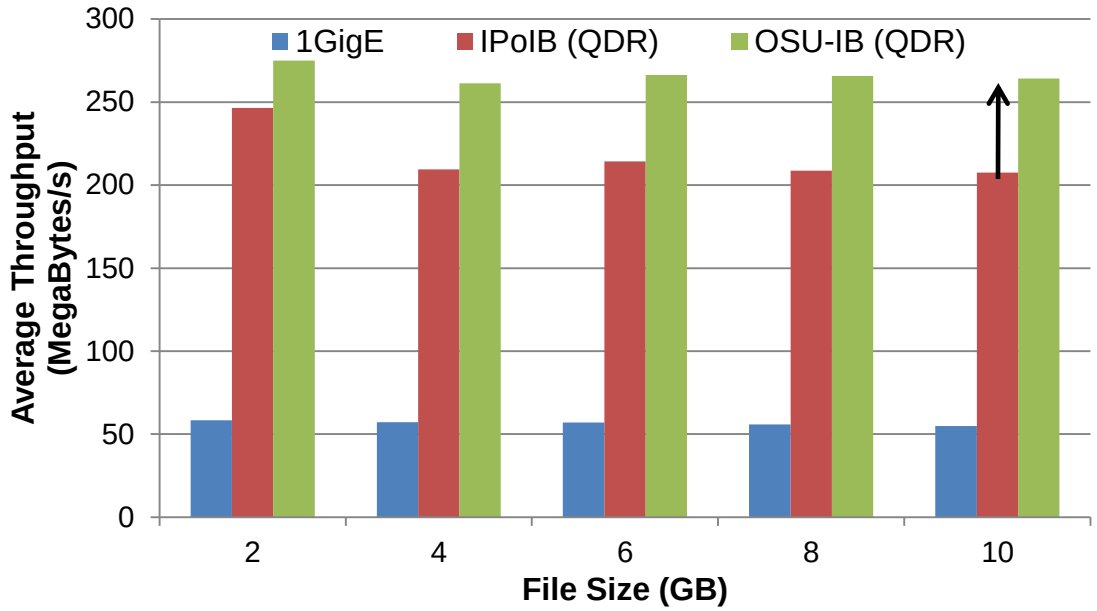
Evaluations using TestDFSIO



- Cluster B with 4 DataNodes, **2 HDD** per node
  - **16.2%** improvement over IPoIB (32Gbps) for 10GB file size
- Cluster B with 4 DataNodes, **1 HDD** per node
  - **10%** improvement over IPoIB (32Gbps) for 10GB file size
- 2 HDD vs 1 HDD
  - **2.01x** improvement for OSU-IB (32Gbps)
  - **1.8x** improvement for IPoIB (32Gbps)

Cluster B, 4 storage nodes

Evaluations using TestDFSIO



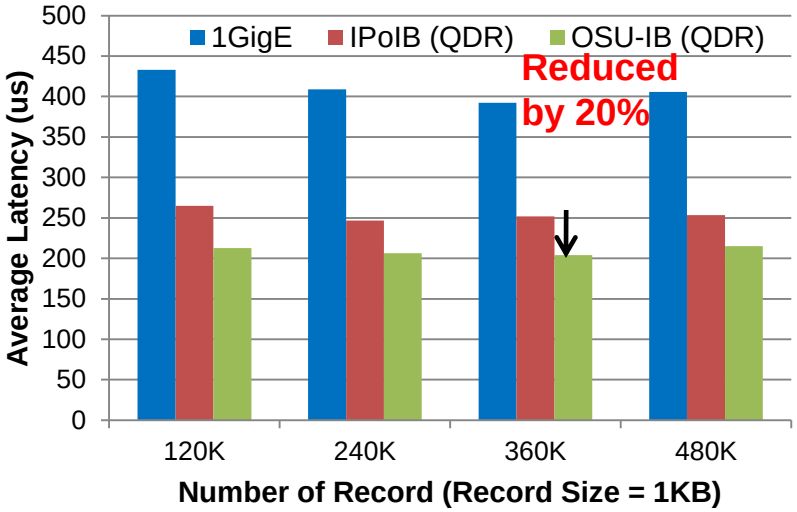
- Cluster B with 4 DataNodes, **1 SSD** per node
  - **28%** improvement over IPoIB (32Gbps) for 10GB file size

Cluster B, 4 storage nodes

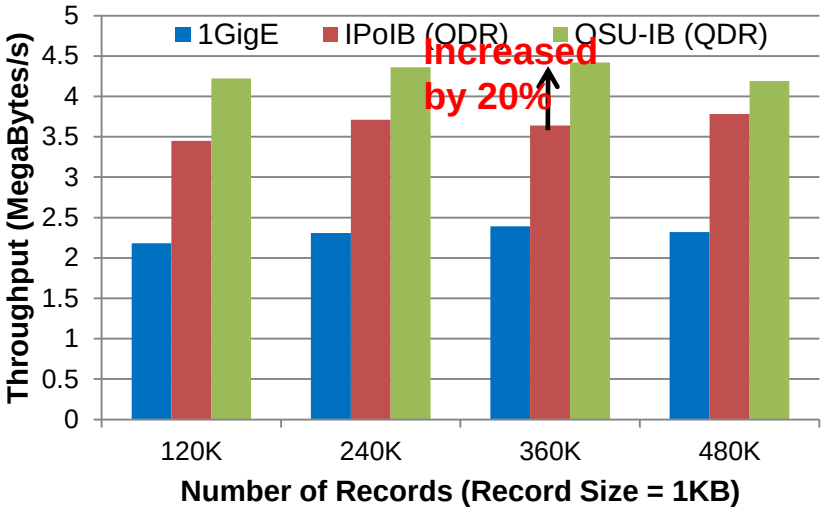
Outline

- Introduction and Motivation
- Problem Statement
- Design
- Performance Evaluation
  - Determining the optimal packet-size for different interconnects/protocols
  - Micro-benchmark level evaluations (measures the latency of file write)
  - Evaluations with TestDFSIO (measures the throughput of sequential file write)
  - Integration with HBase (TCP/IP) (measures latency and throughput of HBase put operation)
- Conclusion & Future Work

Evaluations using YCSB (Single Region Server: 100% Update)



Put Latency

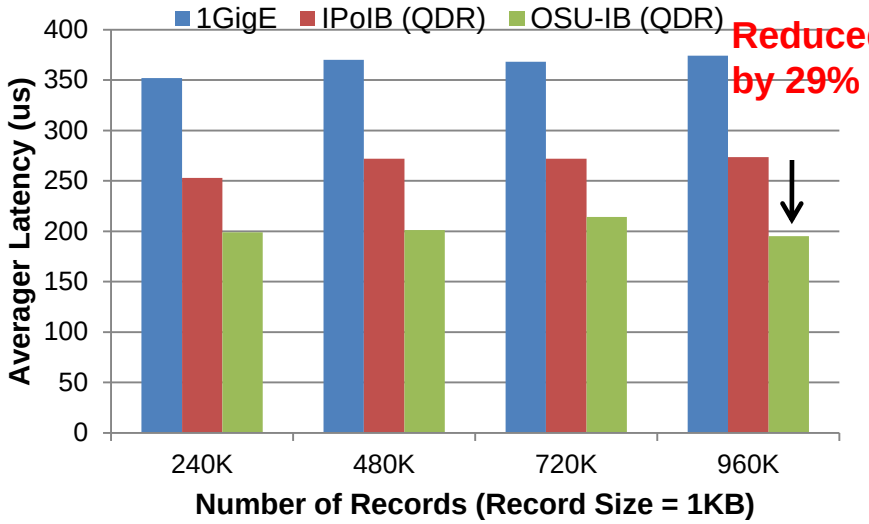


Put Throughput

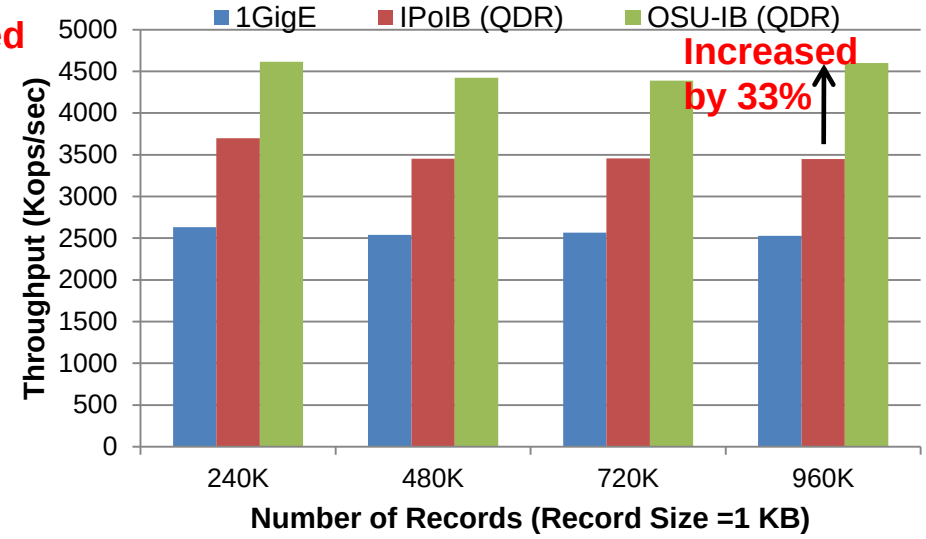
- HBase *Put* latency for 360K records
  - 204 us for OSU Design; 252 us for IPoIB (32Gbps)
- HBase *Put* throughput for 360K records
  - 4.42 Kops/sec for OSU Design; 3.63 Kops/sec for IPoIB (32Gbps)
- **20%** improvement in both average latency and throughput

Cluster B

Evaluations using YCSB (32 Region Servers: 100% Update)



Put Latency



Put Throughput

- HBase *Put* latency for 960K records
  - 195 us for OSU Design; 273 us for IPoIB (32Gbps)
- HBase *Put* throughput for 960K records
  - 4.60 Kops/sec for OSU Design; 3.45 Kops/sec for IPoIB (32Gbps)
- **29%** improvement in average latency; **33%** improvement in throughput

Cluster B

Outline

- Introduction and Motivation
- Problem Statement
- Design using Hybrid Transports
- Performance Evaluation
- Conclusion & Future Work

Conclusion

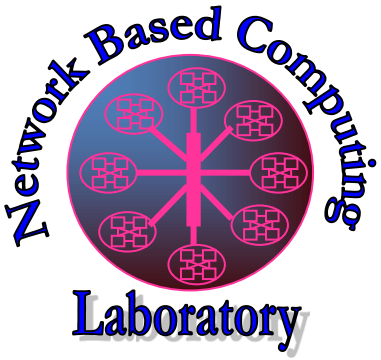
- Detailed Profiling and Analysis of HDFS
- RDMA-based Design of HDFS over InfiniBand
  - First design of HDFS over InfiniBand network
- Comprehensive Evaluation of the RDMA-based design of HDFS
- Integration with HBase leads performance improvement of HBase Put operation

Future Works

- Identify architectural bottlenecks of higher level HDFS designs and propose enhancements to work with high performance communication schemes
- Investigate on faster recovery on DataNode Failure
- Other HDFS operations (e.g. HDFS Read) will be implemented over RDMA
- Integration with other Hadoop components designed over InfiniBand

Thank You!

{islamn, rahmanmd, jose, rajachan,
wangh, subramon, panda}@cse.ohio-state.edu,
chet@watson.ibm.com



Network-Based Computing Laboratory

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

MVAPICH Web Page

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