Slides are available from

http://www.cse.ohio-state.edu/~panda/hoti15-bigdata-tut.pdf

Accelerating Big Data Processing with Hadoop, Spark, and Memcached over High-Performance Interconnects

Tutorial at HotI '15

by

Dhabaleswar K. (DK) Panda

The Ohio State University

E-mail: panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda

Xiaoyi Lu

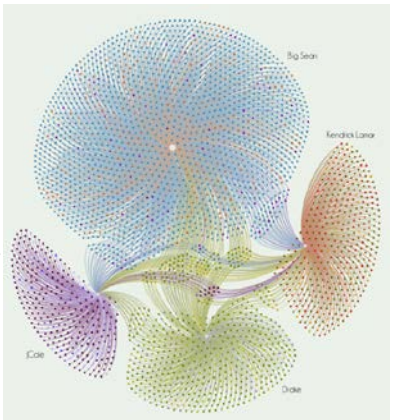
The Ohio State University

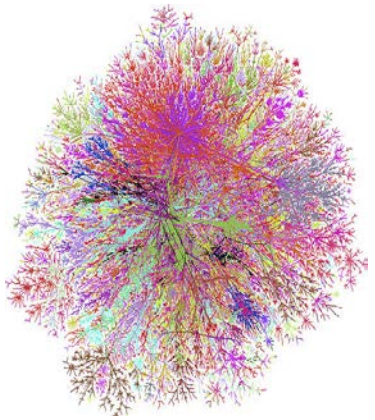
E-mail: luxi@cse.ohio-state.edu

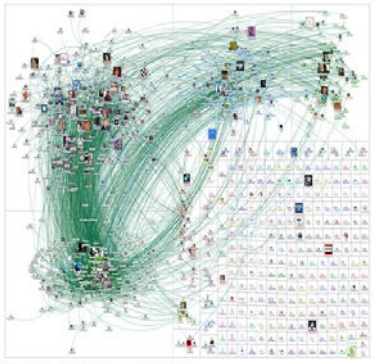
http://www.cse.ohio-state.edu/~luxi

Introduction to Big Data Applications and Analytics

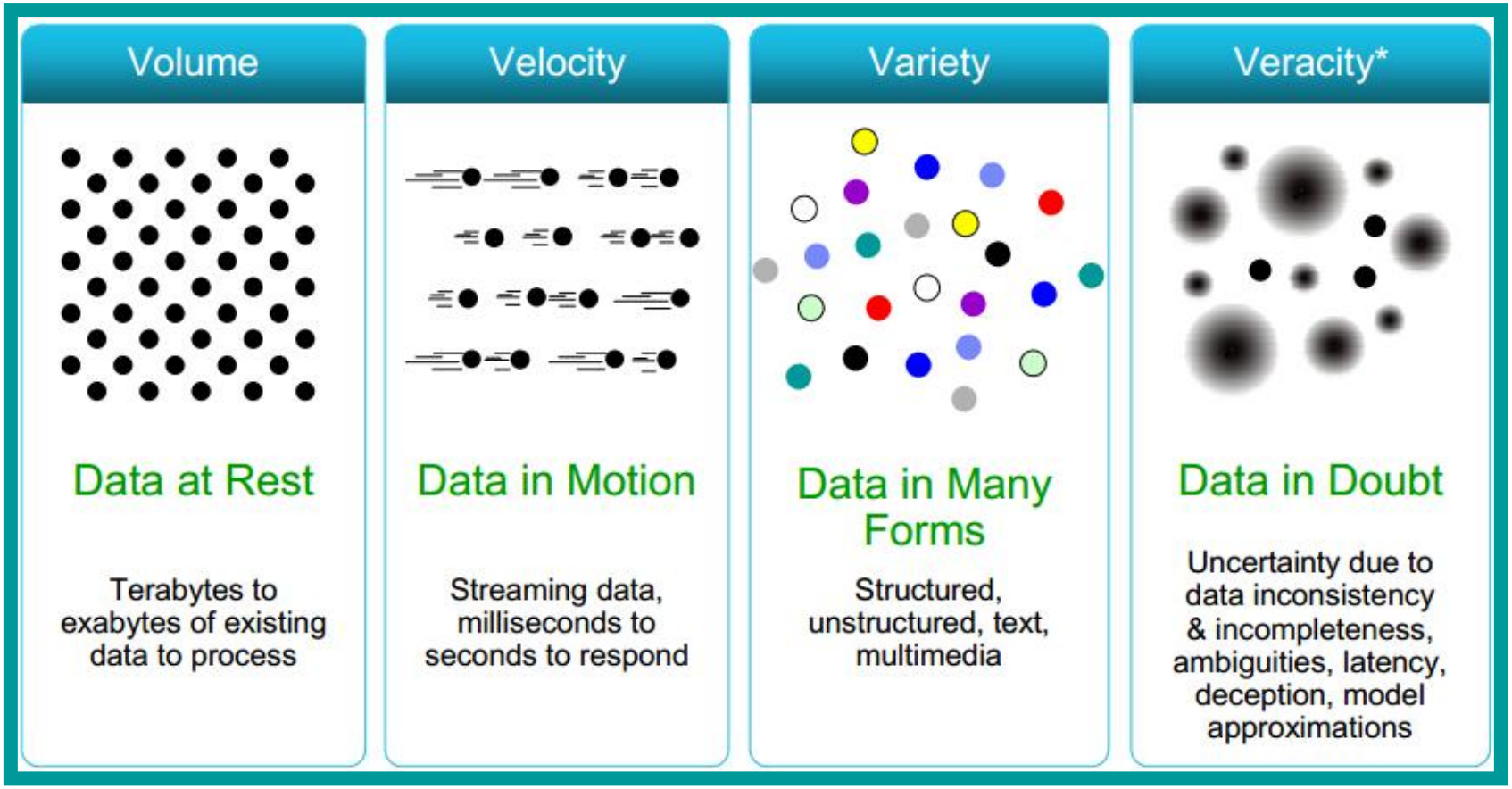
- Big Data has become the one of the most important elements of business analytics
- Provides groundbreaking opportunities for enterprise information management and decision making
- The amount of data is exploding; companies are capturing and digitizing more information than ever
- The rate of information growth appears to be exceeding Moore's Law







4V Characteristics of Big Data

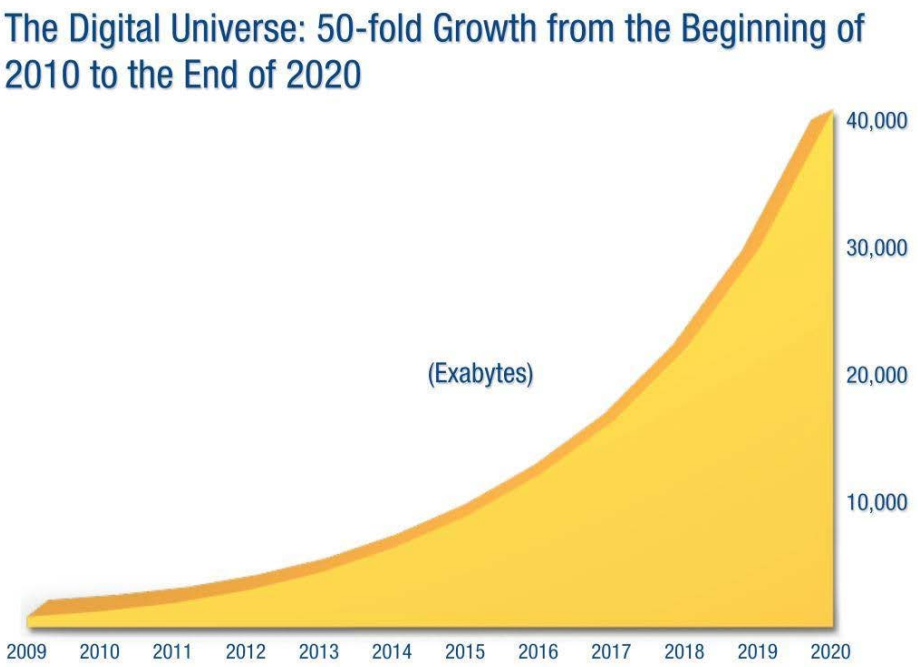


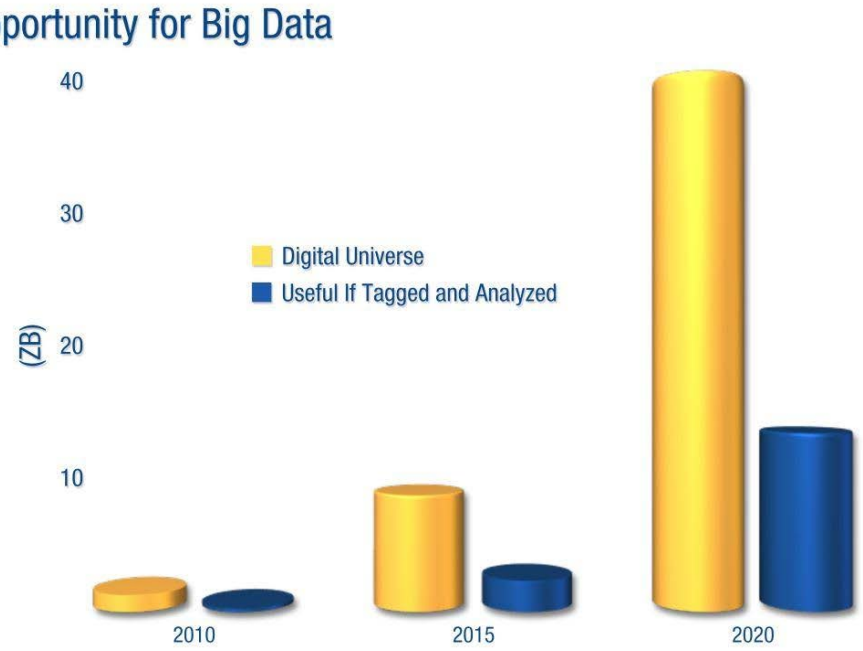
- Commonly accepted **3V**'s of Big Data
  - **Volume, Velocity, Variety**

  Michael Stonebraker: Big Data Means at Least Three Different Things, http://www.nist.gov/itl/ssd/is/upload/NIST-stonebraker.pdf
- **4/5V**'s of Big Data – **3V** + ***Veracity, *Value**

Courtesy: http://api.ning.com/files/tRHkwQN7s-Xz5cxylXG004GLGJdjoPd6bVfVBwvgu*F5MwDDUCiHHdmBW-JTEz0cfJjGurJucBMTkIUNdL3jcZT8IPfNWfN9/dv1.jpg

Big Volume of Data by the End of this Decade



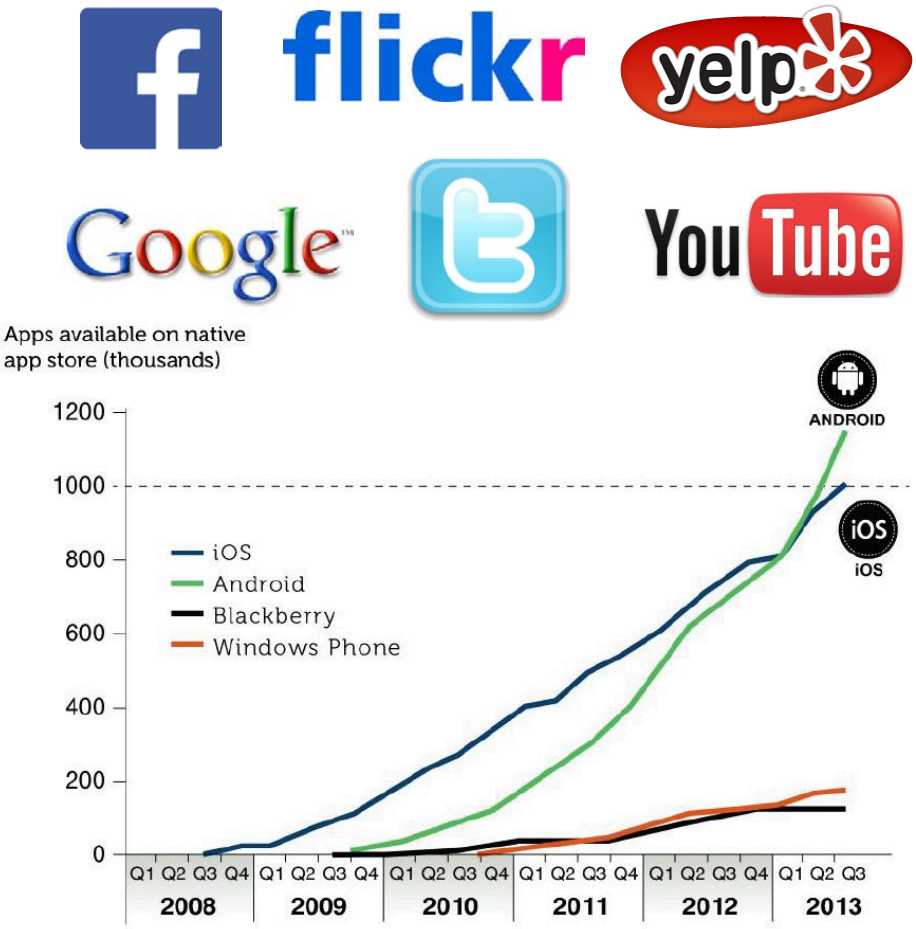


- From 2005 to 2020, the digital universe will grow by a factor of 300, from 130 exabytes to **40,000 exabytes**.
- By 2020, **a third of** the data in the digital universe (more than **13,000 exabytes**) will have **Big Data Value**, but only if it is tagged and analyzed.

Courtesy: John Gantz and David Reinsel, "The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East", IDC's Digital Universe Study, sponsored by EMC, December 2012.

Data Generation in Internet Services and Applications

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect, etc.)
- e-mails (Hotmail, Y!Mail, Gmail, etc.)
- Photos, Movies (Flickr, YouTube, Video, etc.)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market, App Store, etc.)
- Location (Latitude, Loopt, Foursquared, Google Now, etc.)
- User generated content (Wikipedia & co, etc.)
- Ads (display, text, DoubleClick, Yahoo, etc.)
- Comments (Discuss, Facebook, etc.)
- Reviews (Yelp, Y!Local, etc.)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook, etc.)
- Purchase decisions (Netflix, Amazon, etc.)
- Instant Messages (YIM, Skype, Gtalk, etc.)
- Search terms (Google, Bing, etc.)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News, etc.)
- Blog posts (Tumblr, Wordpress, etc.)
- Microblogs (Twitter, Jaiku, Meme, etc.)
- Link sharing (Facebook, Delicious, Buzz, etc.)
- Network traffic



Number of Apps in the Apple App Store, Android Market, Blackberry, and Windows Phone (2013)

- Android Market: <1200K
- Apple App Store: ~1000K

Courtesy: http://dazeinfo.com/2014/07/10/apple-inc-aapl-ios-google-inc-goog-android-growth-mobile-ecosystem-2014/

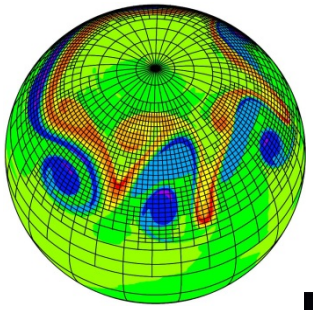
Velocity of Big Data – How Much Data Is Generated Every Minute on the Internet?

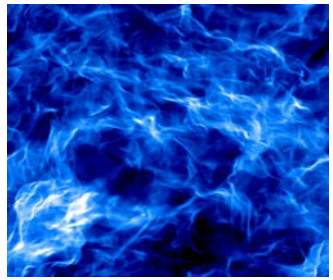
The global Internet population grew 14.3% from 2011 to 2013 and now represents 2.4 Billion People.

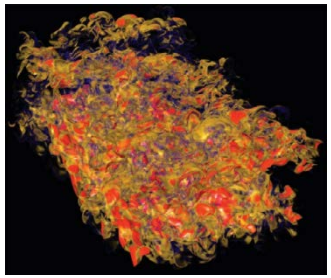
Courtesy: http://www.domo.com/blog/2014/04/data-never-sleeps-2-0/

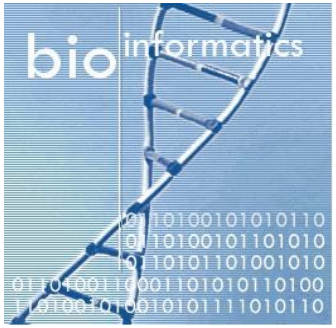
Not Only in Internet Services - Big Data in Scientific Domains

- Scientific Data Management, Analysis, and Visualization
- Applications examples
  - Climate modeling
  - Combustion
  - Fusion
  - Astrophysics
  - Bioinformatics









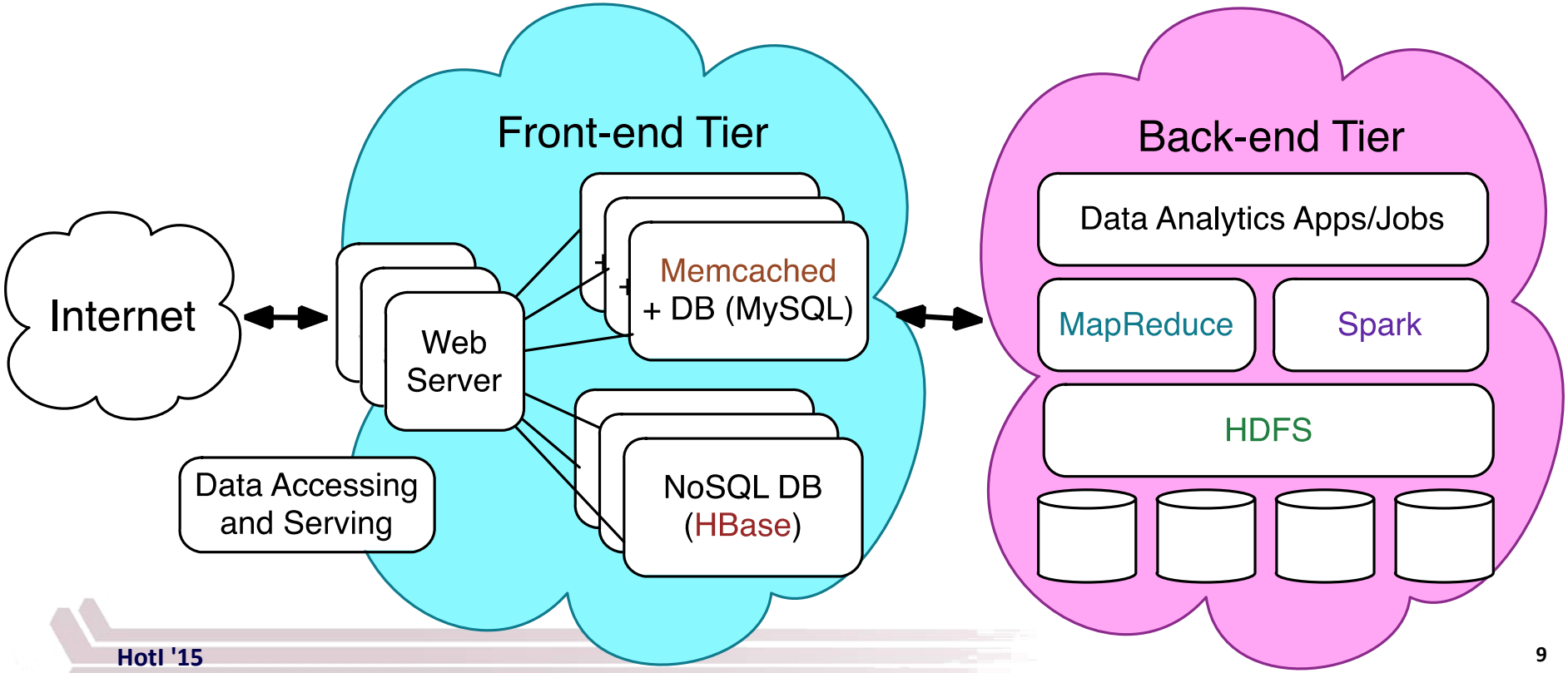
- Data Intensive Tasks
  - Runs large-scale simulations on supercomputers
  - Dump data on parallel storage systems
  - Collect experimental / observational data
  - Move experimental / observational data to analysis sites
  - Visual analytics – help understand data visually

Typical Solutions or Architectures for Big Data Analytics

- Hadoop: http://hadoop.apache.org
  - The most popular framework for Big Data Analytics
  - HDFS, MapReduce, HBase, RPC, Hive, Pig, ZooKeeper, Mahout, etc.
- Spark: http://spark-project.org
  - Provides primitives for in-memory cluster computing; Jobs can load data into memory and query it repeatedly
- Storm: http://storm-project.net
  - A distributed real-time computation system for real-time analytics, online machine learning, continuous computation, etc.
- S4: http://incubator.apache.org/s4
  - A distributed system for processing continuous unbounded streams of data
- GraphLab: http://graphlab.org
  - Consists of a core C++ GraphLab API and a collection of high-performance machine learning and data mining toolkits built on top of the GraphLab API.
- Web 2.0: RDBMS + Memcached (http://memcached.org)
  - Memcached: A high-performance, distributed memory object caching systems

Data Management and Processing on Modern Clusters

- Substantial impact on designing and utilizing modern data management and processing systems in multiple tiers
  - Front-end data accessing and serving (Online)
    - Memcached + DB (e.g. MySQL), HBase
  - Back-end data analytics (Offline)
    - HDFS, MapReduce, Spark



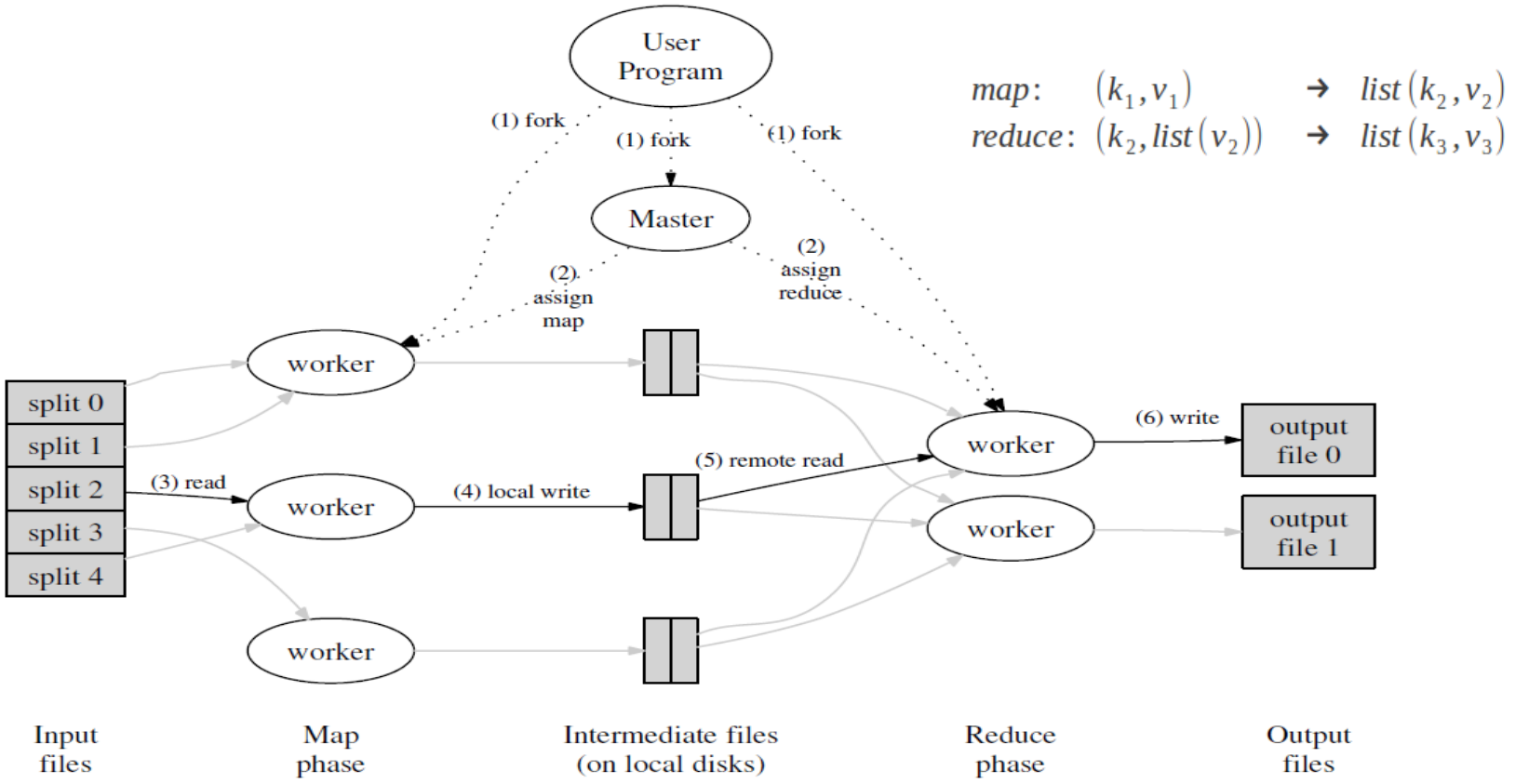
Who Are Using Hadoop?

- Focuses on large data and data analysis
- Hadoop (e.g. HDFS, MapReduce, RPC, HBase) environment is gaining a lot of momentum
- http://wiki.apache.org/hadoop/PoweredBy

Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

The MapReduce Model

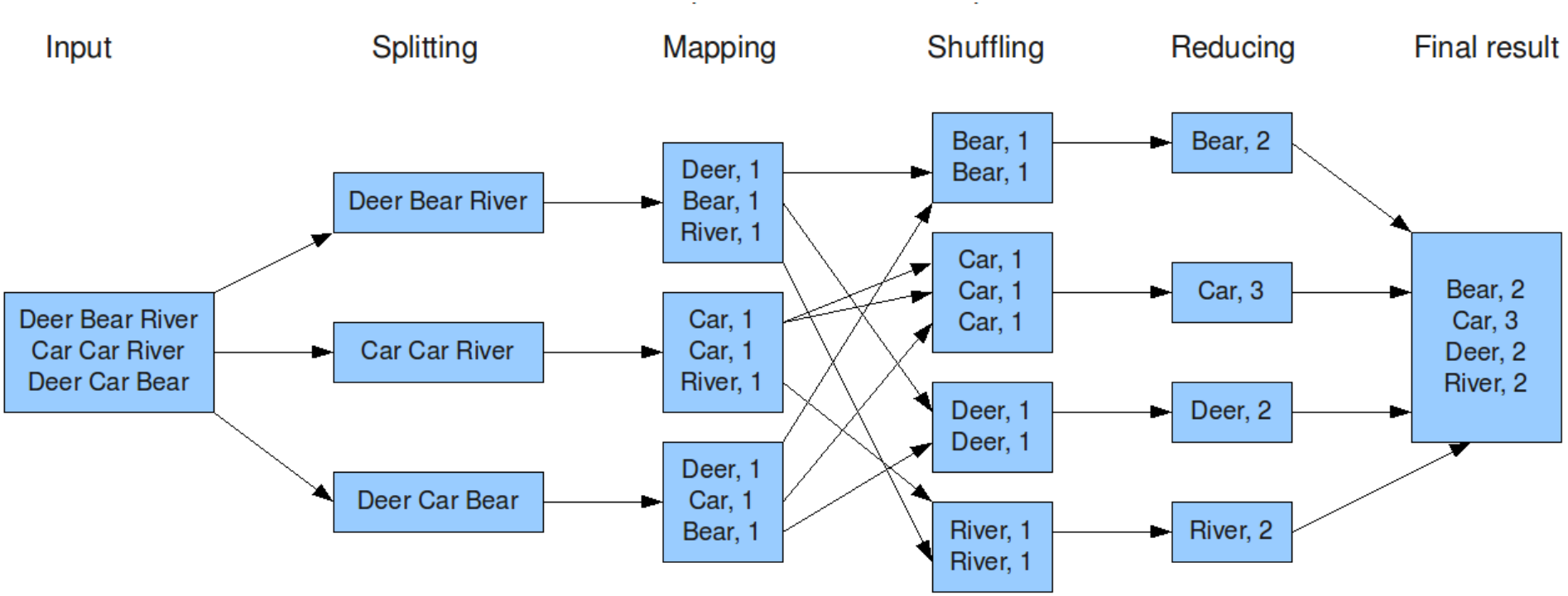


$$map: \quad (k_1, v_1) \rightarrow list(k_2, v_2)$$
$$reduce: \quad (k_2, list(v_2)) \rightarrow list(k_3, v_3)$$

J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. In *Proceedings of the 6th Symposium on Operating Systems Design & Implementation (OSDI'04)*, 2004.

WordCount Execution

- The overall execution process of WordCount in MapReduce



A Hadoop MapReduce Example - WordCount

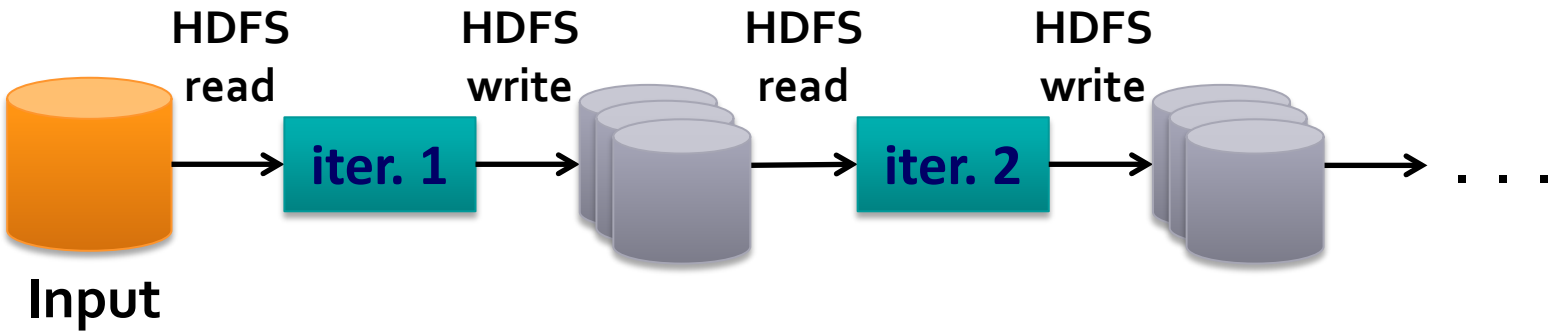
```
public class WordCount {
 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Longw     table ke    Tex     alue, Context context) throws
    IOException, Interrupted
        String line = value
        StringTokenizer to                          izer(line);
        while (tokenizer.h
            word.set(tokenizer
            context.write(word, one
        }
    }
 }
 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>
    {
    public voi   reduc    Text key, Iterator<IntWritable> values, Con   xt context)
      ows    ce         erruptedException {

                                )) {
                                get();
        }
        contex   writ   key, new IntWritable(sum));
    }
 }
}
```

Productive

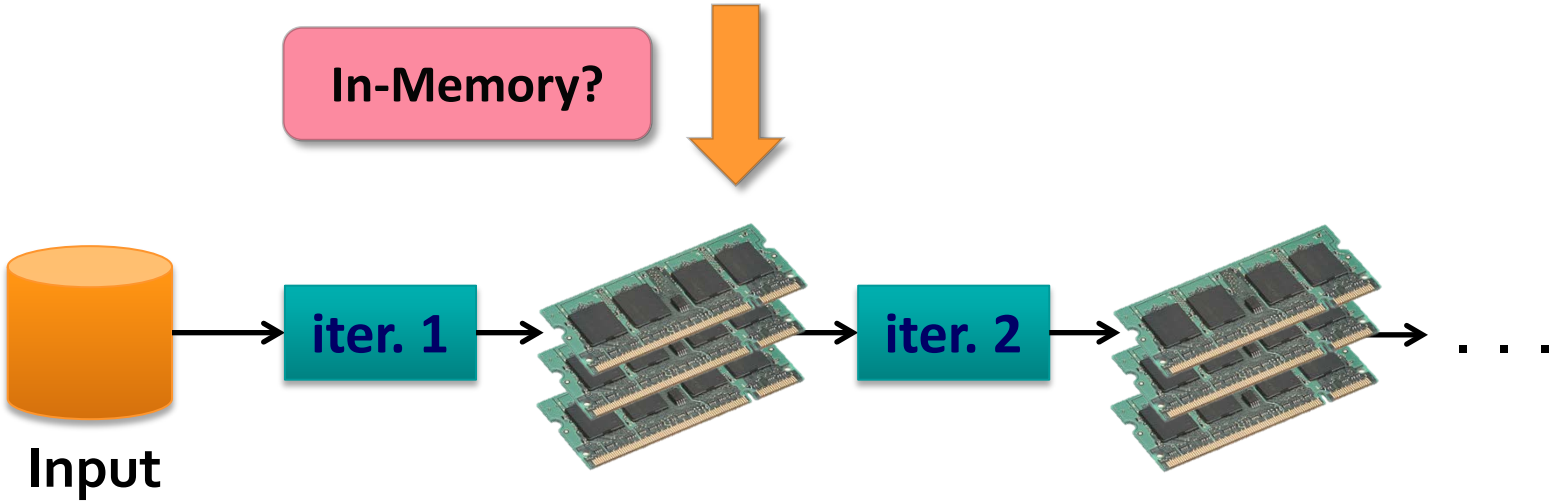
Scalable

Fault-Tolerant

Data Sharing Problems in MapReduce



Slow due to replication, serialization, and disk IO



10-100× faster than network and disk

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 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
…	…

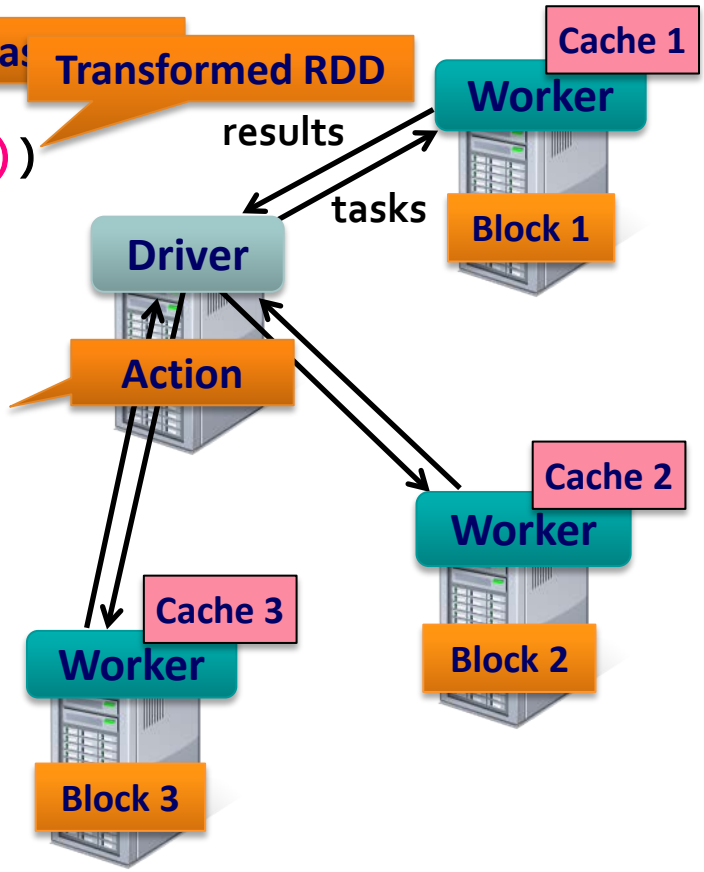
More Information:

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

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
. . .
```



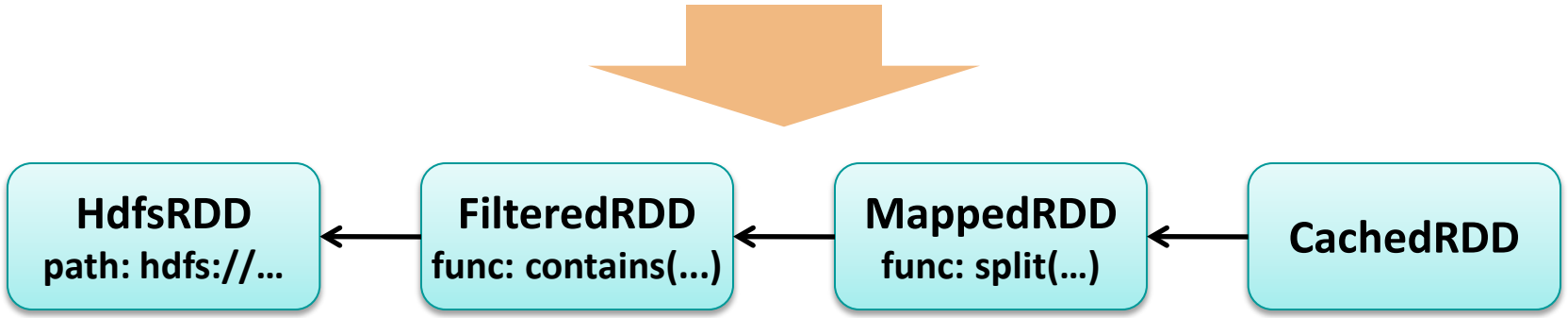
Result: scaled to 1 TB data in 5-7 sec (vs 170 sec for on-disk data)

Courtesy: https://spark.apache.org/

Lineage-based Fault Tolerance

- RDDs maintain ***lineage*** information that can be used to reconstruct lost partitions
- Example

```
cachedMsgs = textFile(...).filter(_.contains("error"))
                          .map(_.split('\t')(2))
                          .cache()
```



RDD Example: Word Count in Spark!

```
val f            extFile("hdfs:/
val cou          flatMap(line =                 ))
                .map(word => (wor           )
                .reduceByKey(_ + _)

counts.saveAsTextFile("hdfs://...")
```

Productive

High-Performance

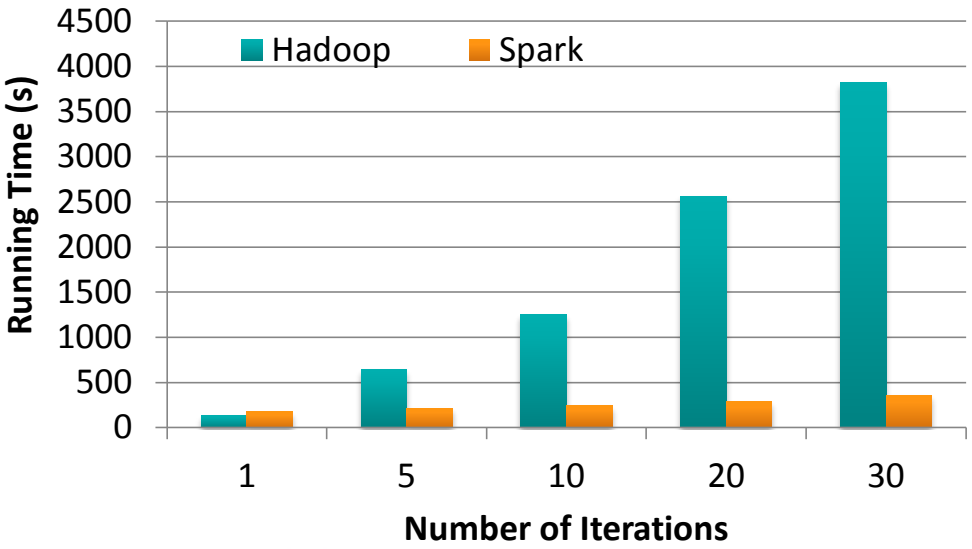
Scalable

Fault-Tolerant

Benefits of RDD Model

- Consistency is easy due to immutability
- Inexpensive fault tolerance (log lineage rather than replicating/checkpointing data)
- Locality-aware scheduling of tasks on partitions
- Despite being restricted, model seems applicable to a broad variety of applications
- Easy Programming
- High-Performance
- Scalable

- **Logistic Regression Performance**
  - Apache Hadoop (127s/iteration)
  - Apache Spark (first iteration : 174s, further iterations: 6s)



Courtesy: https://spark.apache.org/

Presentation Outline

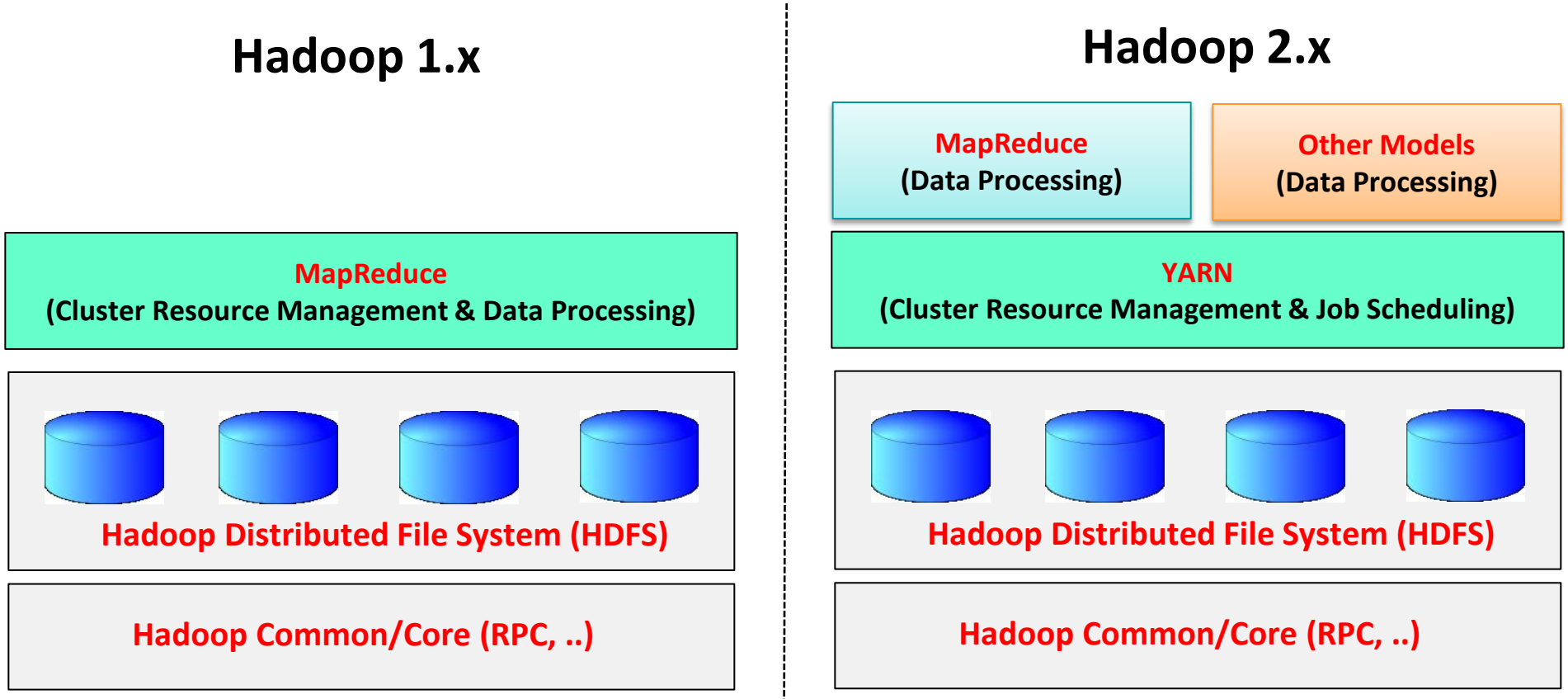
- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

Architecture Overview of Hadoop, Spark, and Memcached

- Overview of Apache Hadoop Architecture and its Components
  - MapReduce
  - HDFS
  - RPC
  - Spark
  - HBase
- Overview of Web 2.0 Architecture and Memcached

Overview of Apache Hadoop Architecture

- Open-source implementation of Google MapReduce, GFS, and BigTable for Big Data Analytics
  - Hadoop Common Utilities (RPC, etc.), HDFS, MapReduce, YARN
- http://hadoop.apache.org

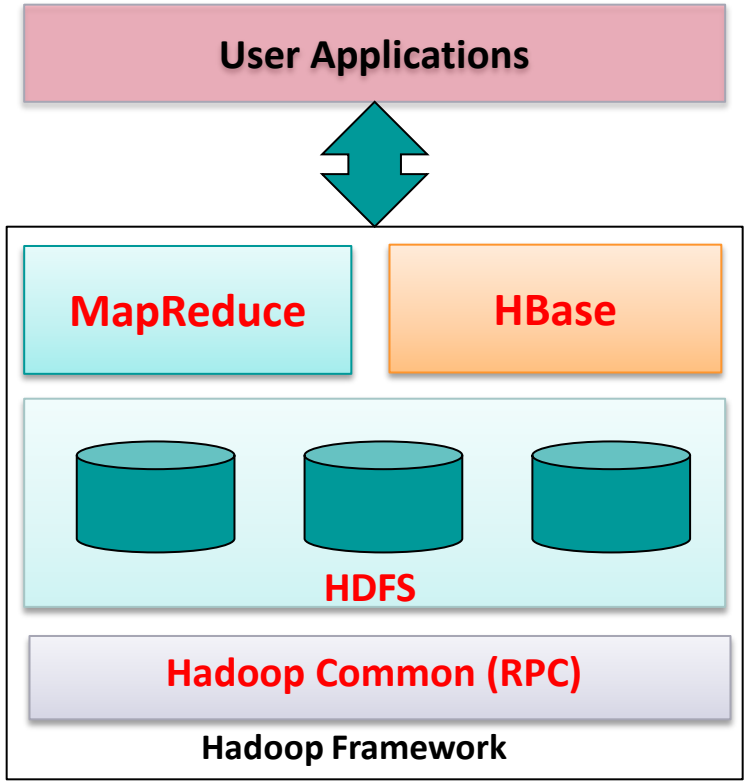


Projects Under Apache Hadoop

- **Ambari**: A web-based tool for provisioning, managing, and monitoring Hadoop clusters.
- **Avro**: A data serialization system.
- **Cassandra**: A scalable multi-master database with no single points of failure.
- **Chukwa**: A data collection system for managing large distributed systems.
- **HBase**: A scalable database that supports structured data storage for large tables.
- **Hive**: A data warehouse infrastructure that provides data summarization and ad-hoc querying.
- **Mahout**: A scalable machine learning and data mining library.
- **Pig**: A high-level data-flow language and execution framework for parallel computation.
- **Spark**: A fast and general compute engine for data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- **Tez**: A generalized data-flow programming framework, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases. Tez is being adopted to replace Hadoop MapReduce.
- **ZooKeeper**: A high-performance coordination service for distributed applications.

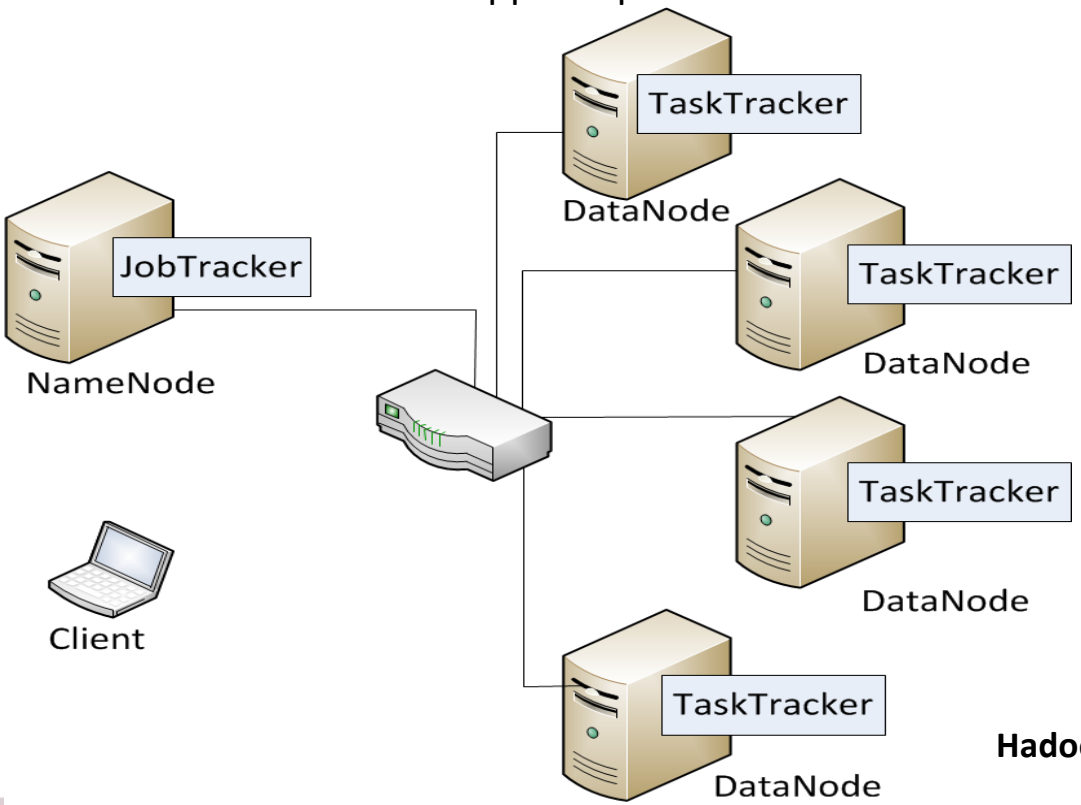
Big Data Processing with Hadoop Components

- Major components included in this tutorial:
  - MapReduce (Batch)
  - HBase (Query)
  - HDFS (Storage)
  - RPC (Inter-process communication)
- Underlying Hadoop Distributed File System (HDFS) used by both MapReduce and HBase
- Model scales but high amount of communication during intermediate phases can be further optimized



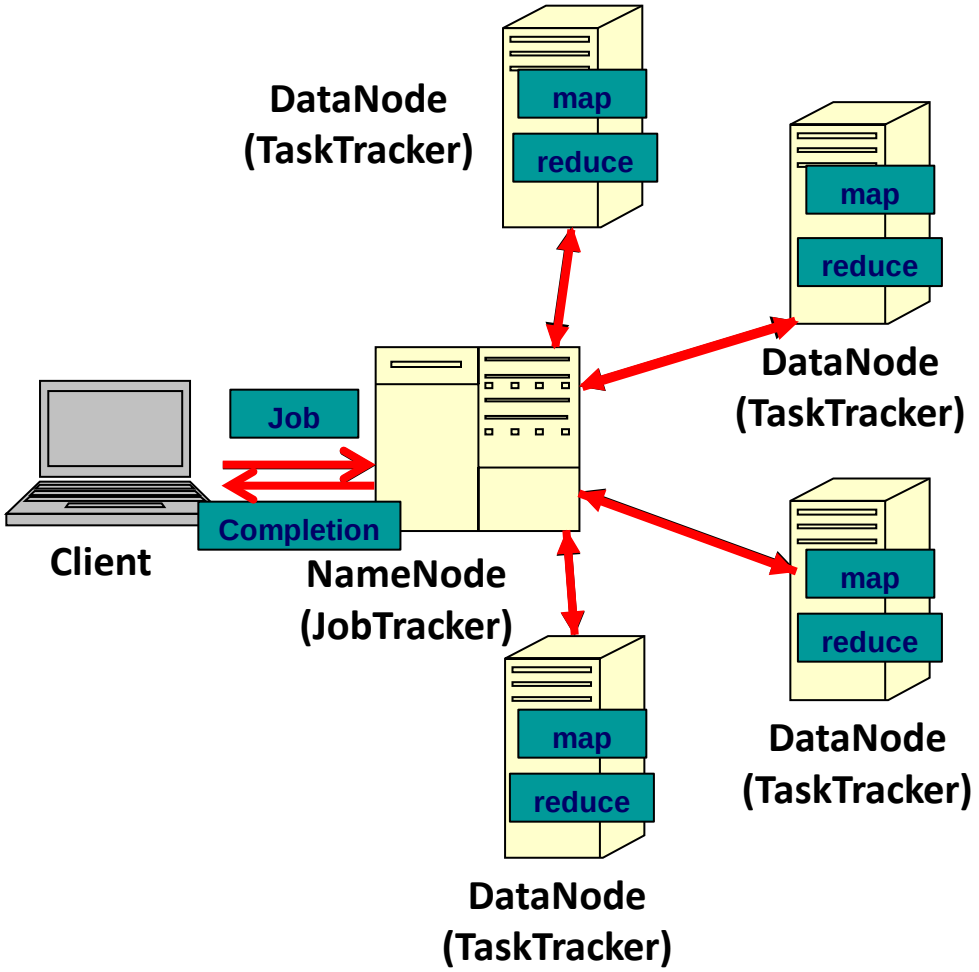
Typical Hadoop 1.x Cluster

- MapReduce Framework
  - JobTracker: track, manage jobs and detect failure
  - TaskTrackers: host map/reduce tasks computation
- HDFS
  - NameNode: stores meta information of data blocks
  - DataNodes: store blocks and support operations on blocks



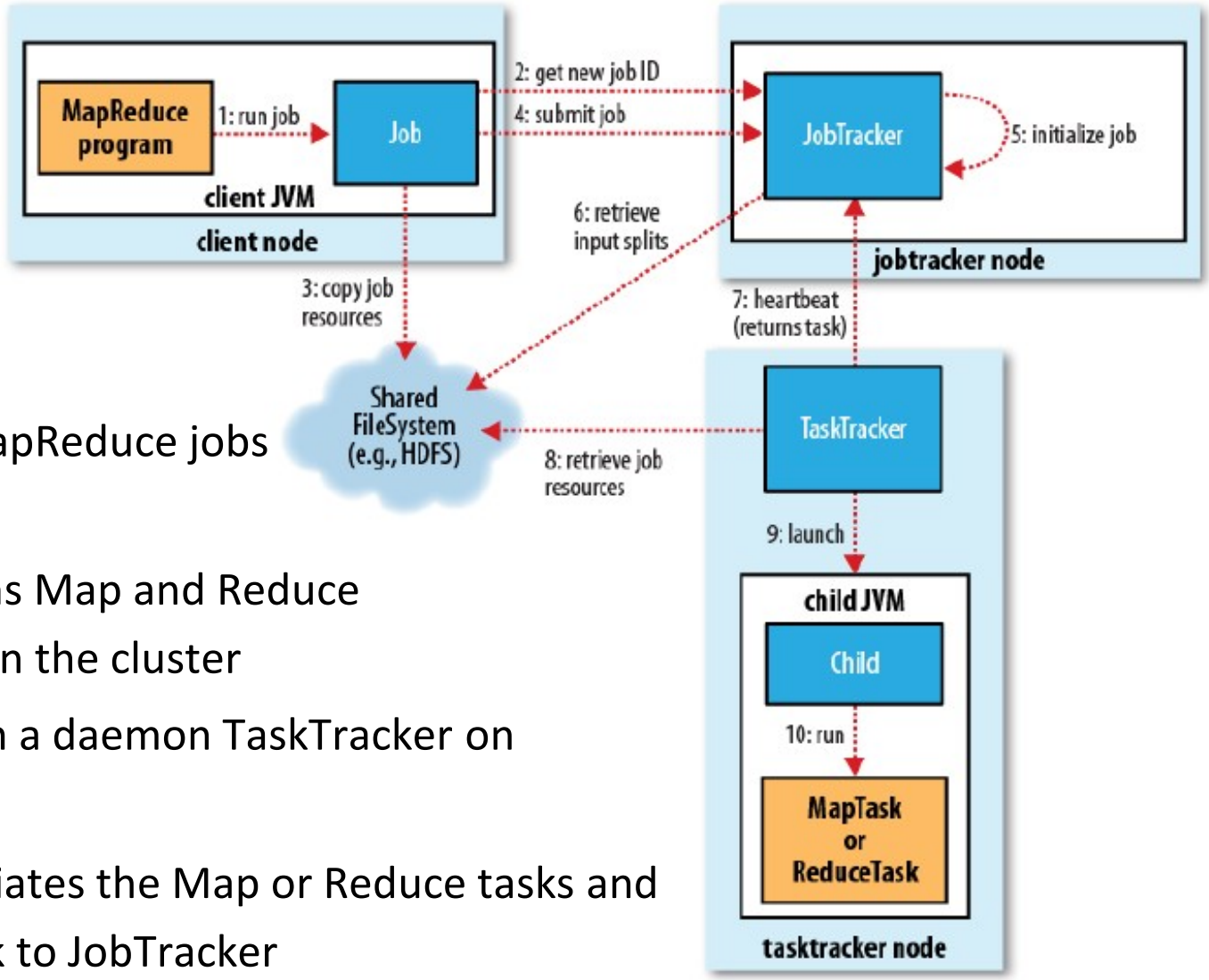
Hadoop Cluster

Hadoop 1.x MapReduce Job Execution



- **Main Features**
  - Replication (e.g. 3)
  - Data locality for Maps
  - HTTP-based Shuffle
  - Speculative execution
  - Independence among tasks
  - …
- **Goals**
  - Fault Tolerance
  - Scalability

MapReduce on Hadoop 1.x

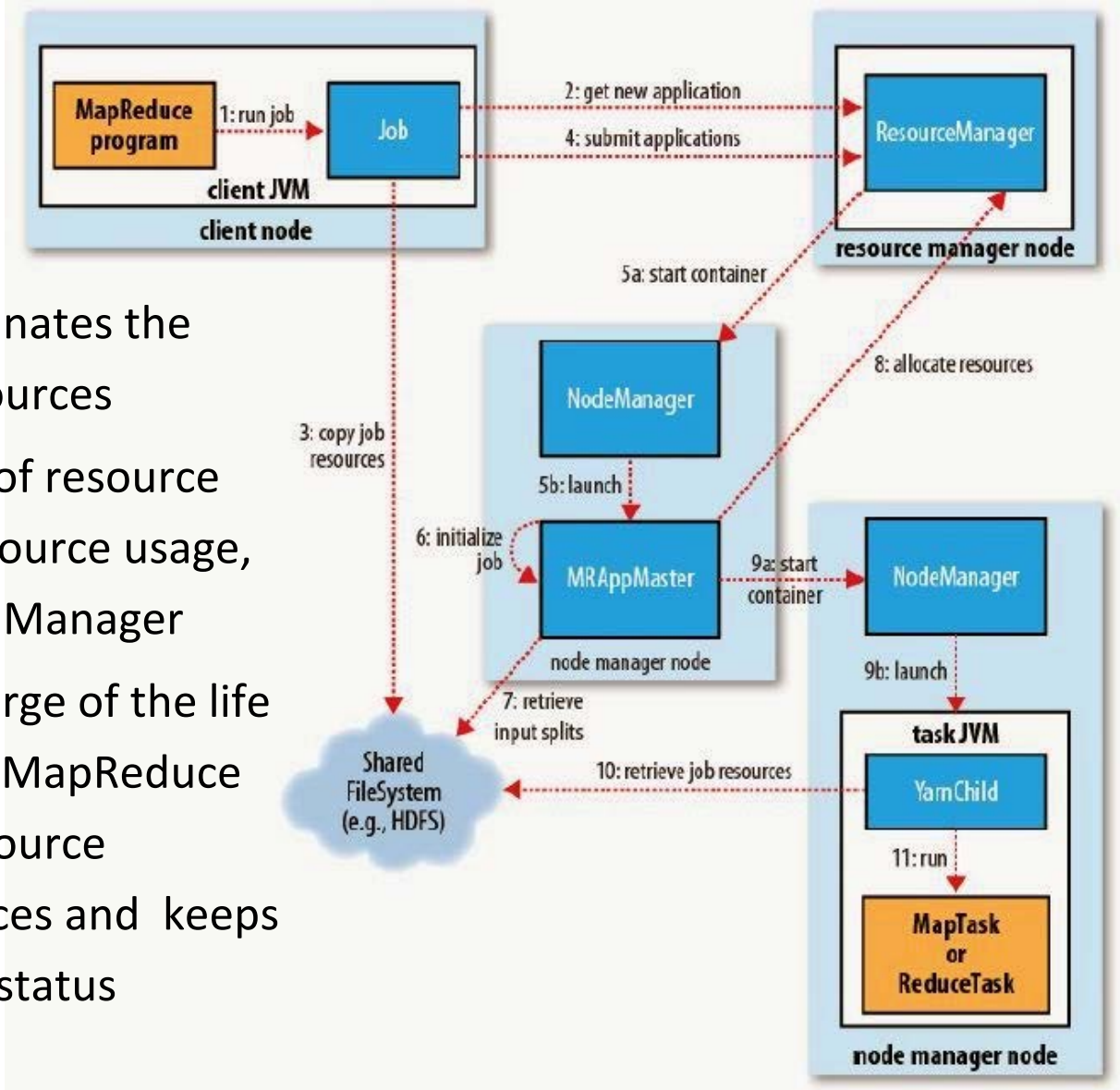


- The clients submit MapReduce jobs to JobTracker
- The JobTracker assigns Map and Reduce tasks to other nodes in the cluster
- These nodes each run a daemon TaskTracker on separate JVM
- Each TaskTracker initiates the Map or Reduce tasks and reports progress back to JobTracker

Courtesy: http://www.cyanny.com/2013/12/05/hadoop-mapreduce-1-framework

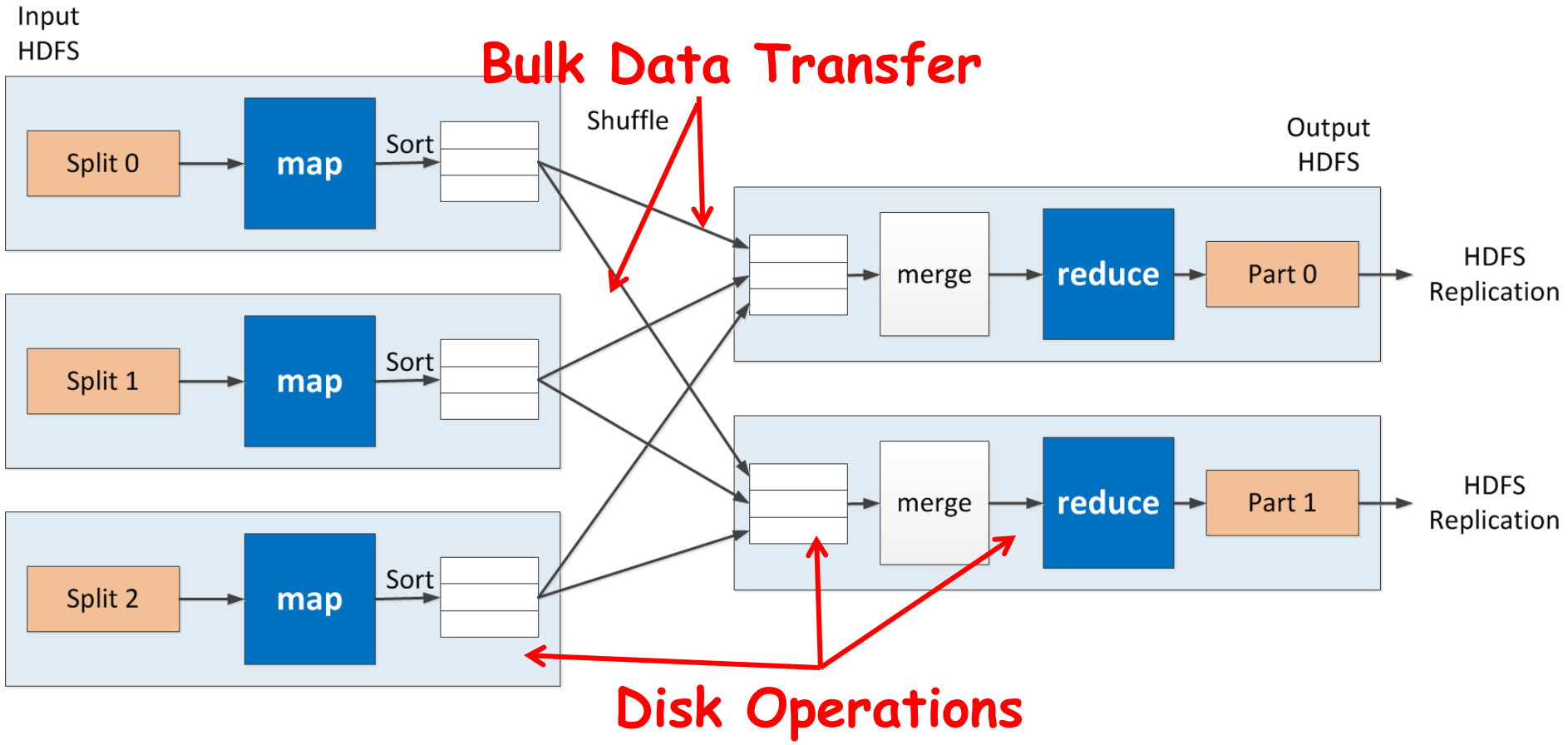
MapReduce on Hadoop 2.x -- YARN Architecture

- **Resource Manager**: coordinates the allocation of compute resources
- **Node Manager**: in charge of resource containers, monitoring resource usage, and reporting to Resource Manager
- **Application Master**: in charge of the life cycle an application, like a MapReduce Job. It negotiates with Resource Manager of cluster resources and keeps track of task progress and status



Courtesy: http://www.cyanny.com/2013/12/05/hadoop-mapreduce-2-yarn/

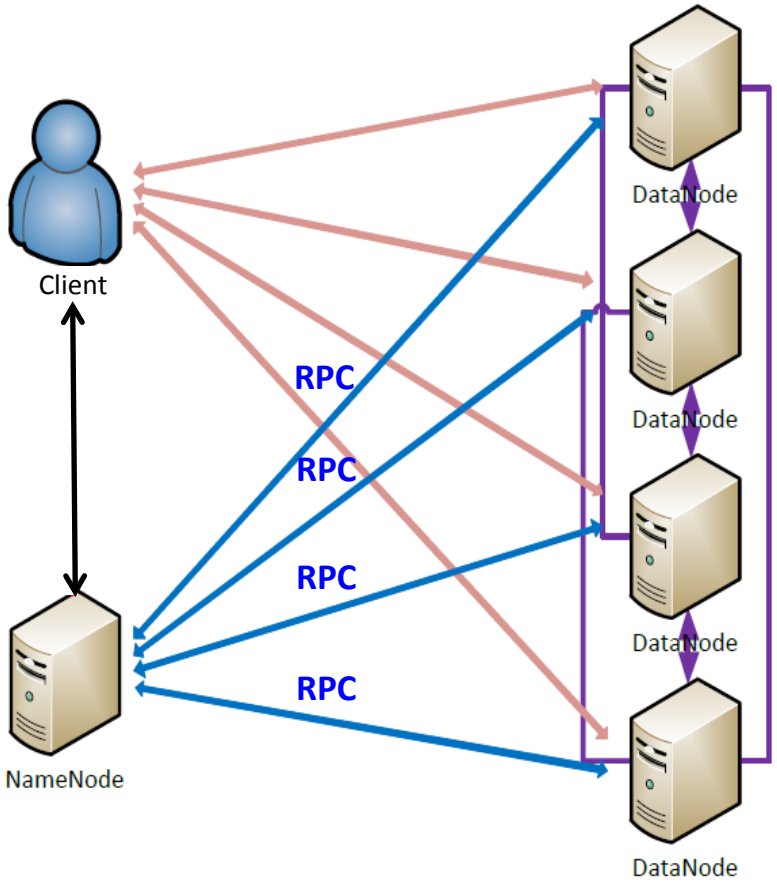
Data Movement in Hadoop MapReduce



- 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
- Communication in shuffle phase uses HTTP over Java Sockets

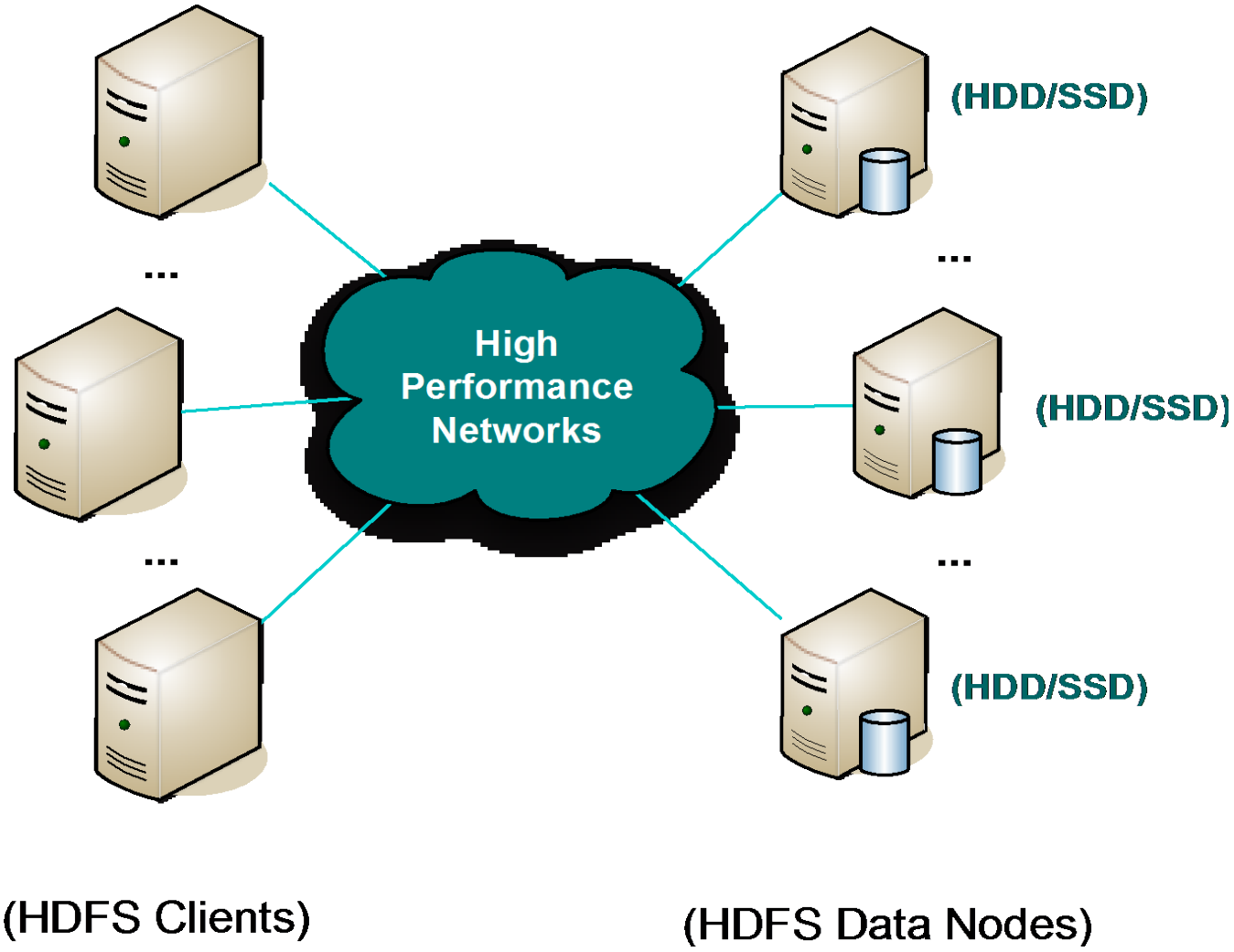
Hadoop Distributed File System (HDFS)

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

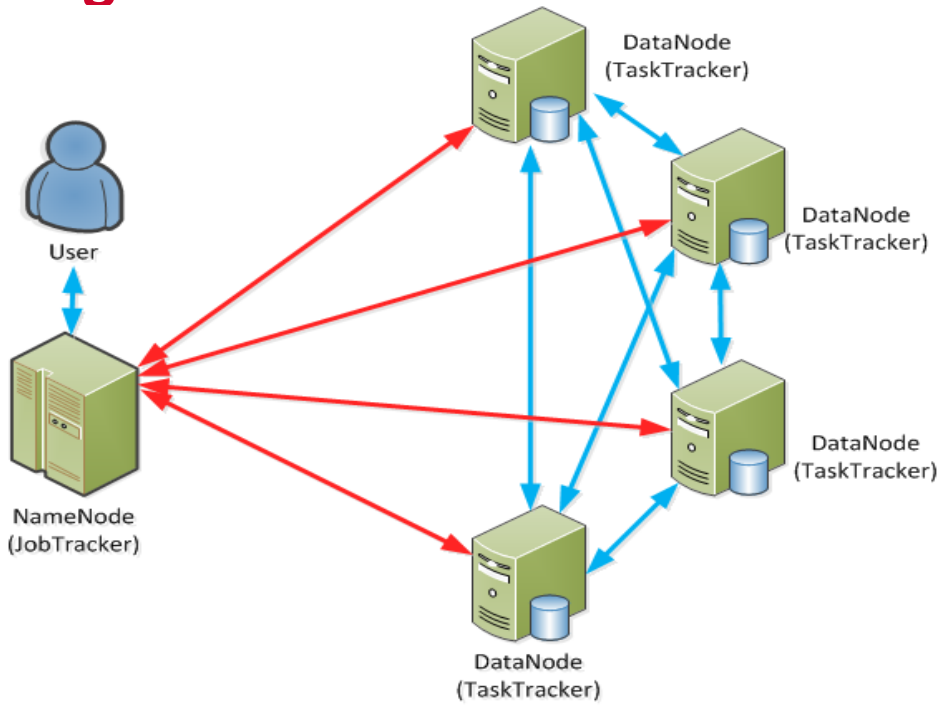


(HDFS Architecture)

Network-Level Interaction Between Clients and Data Nodes in HDFS



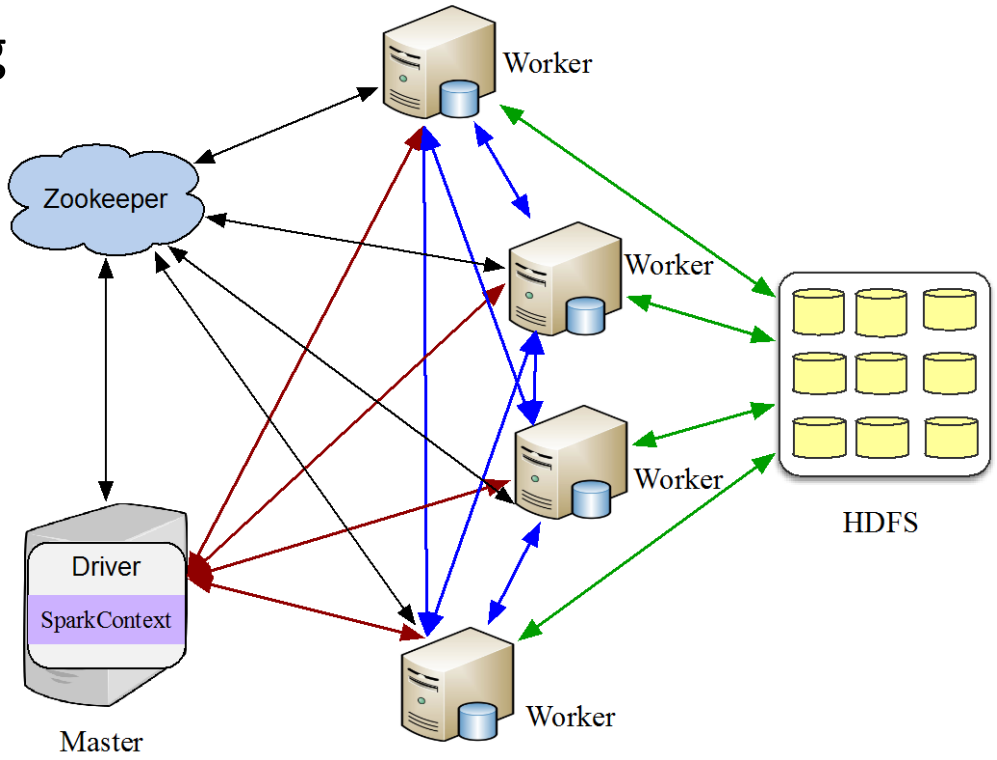
Hadoop RPC Usage



- RPC in HDFS
  - Report the current load and all the information of stored blocks between DNs and NN
  - The HDFS clients use RPC to communicate with NN
- RPC in MapReduce
  - Exchanges information between JT and TTs, e.g. heartbeat messages, task completion events, error reports, etc.
  - Submit a Job for execution and get the current system status
- RPC in HBase
  - A core communication scheme for HBase Put/Get operations

Spark Architecture Overview

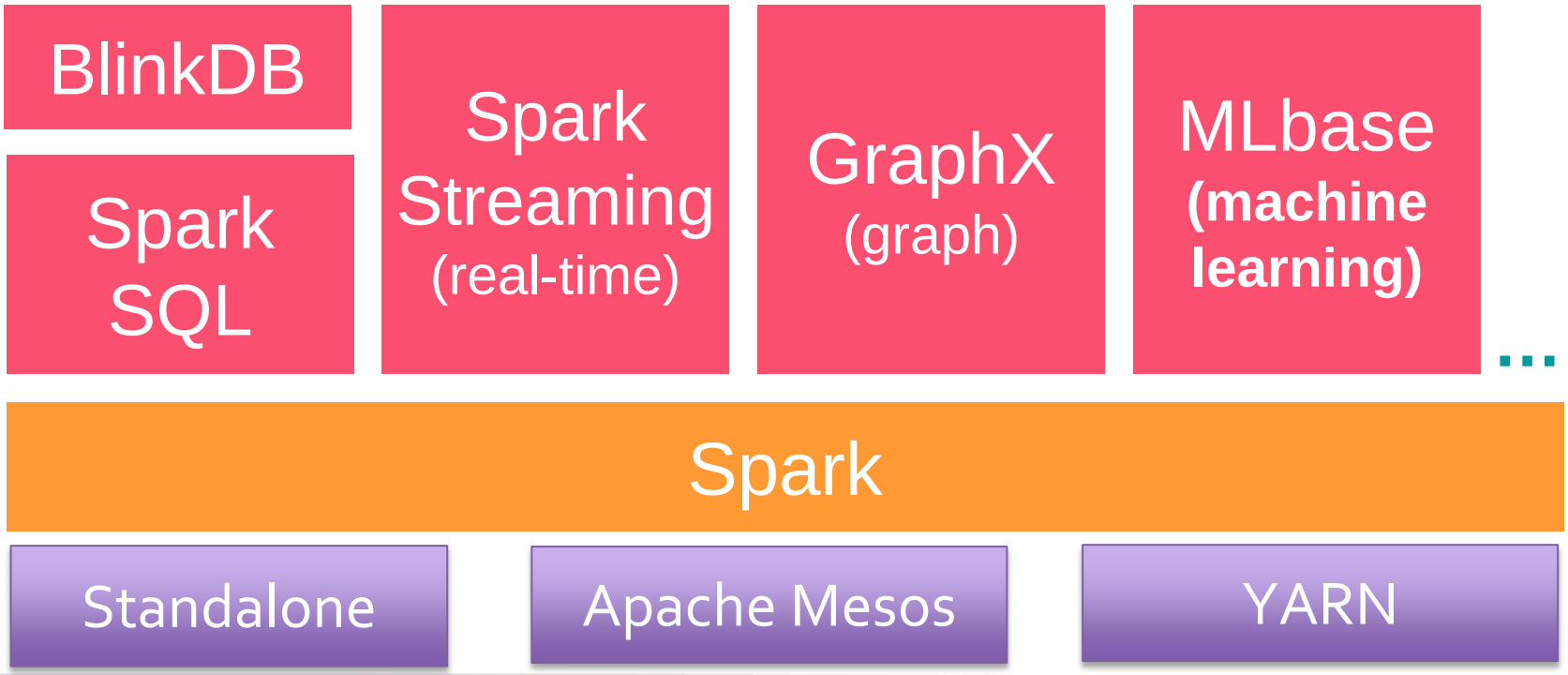
- An in-memory data-processing framework
  - Iterative machine learning jobs
  - Interactive data analytics
  - Scala based Implementation
  - Standalone, YARN, Mesos
- Scalable and communication intensive
  - Wide dependencies between Resilient Distributed Datasets (RDDs)
  - MapReduce-like shuffle operations to repartition RDDs
  - Sockets based communication



http://spark.apache.org

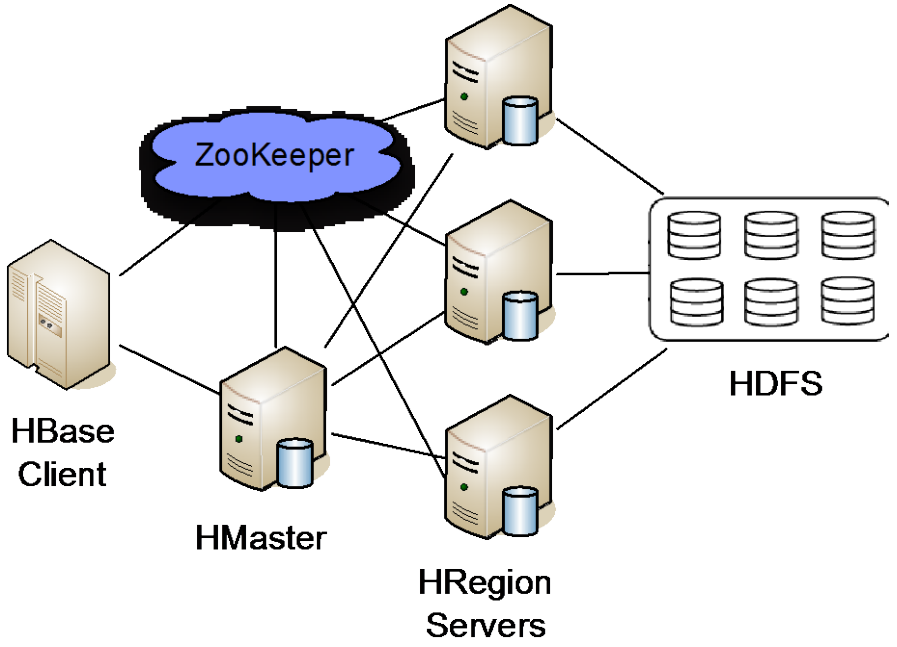
Spark Ecosystem

- Generalize MapReduce to support new apps in same engine
- Two Key Observations
  - General task support with DAG
  - Multi-stage and interactive apps require faster **data sharing** across parallel jobs



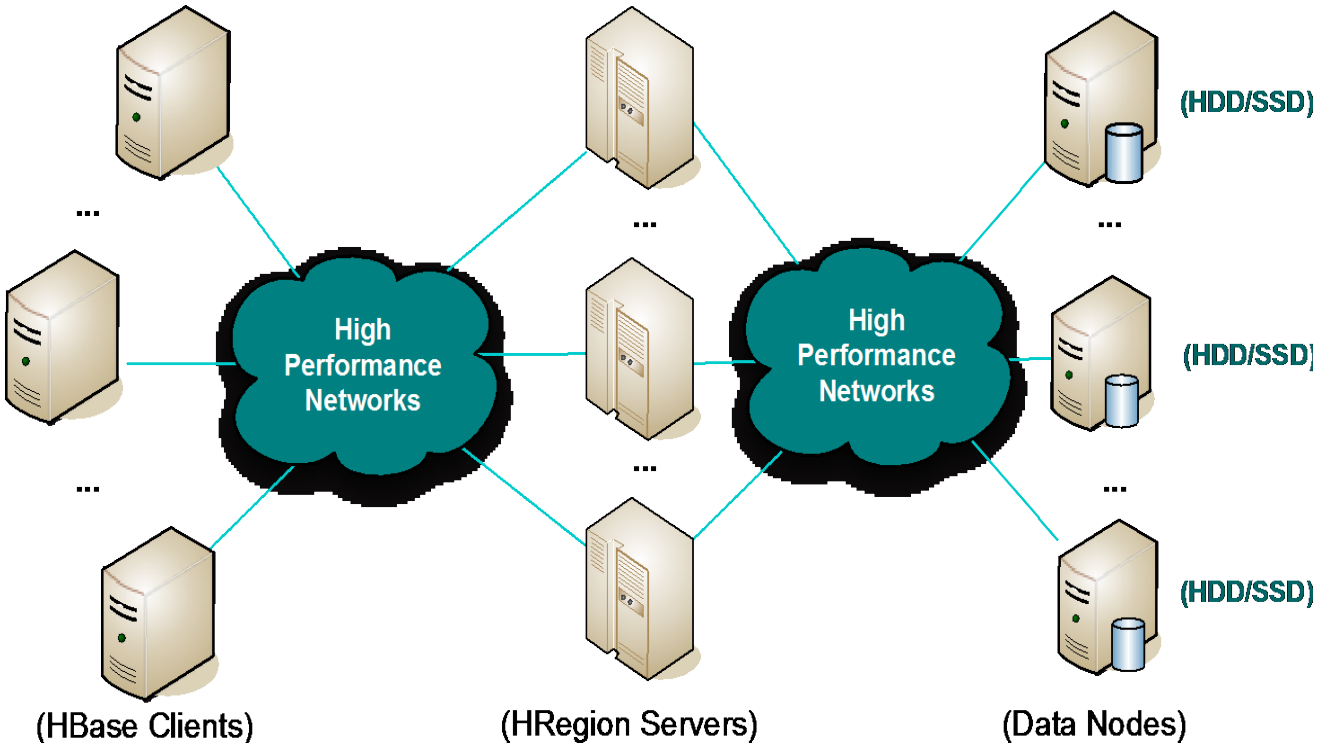
HBase Overview

- Apache Hadoop Database (http://hbase.apache.org/)
- Semi-structured database, which is highly scalable
- Integral part of many datacenter applications
  - eg: Facebook Social Inbox
- Developed in Java for platform-independence and portability
- Uses sockets for communication!



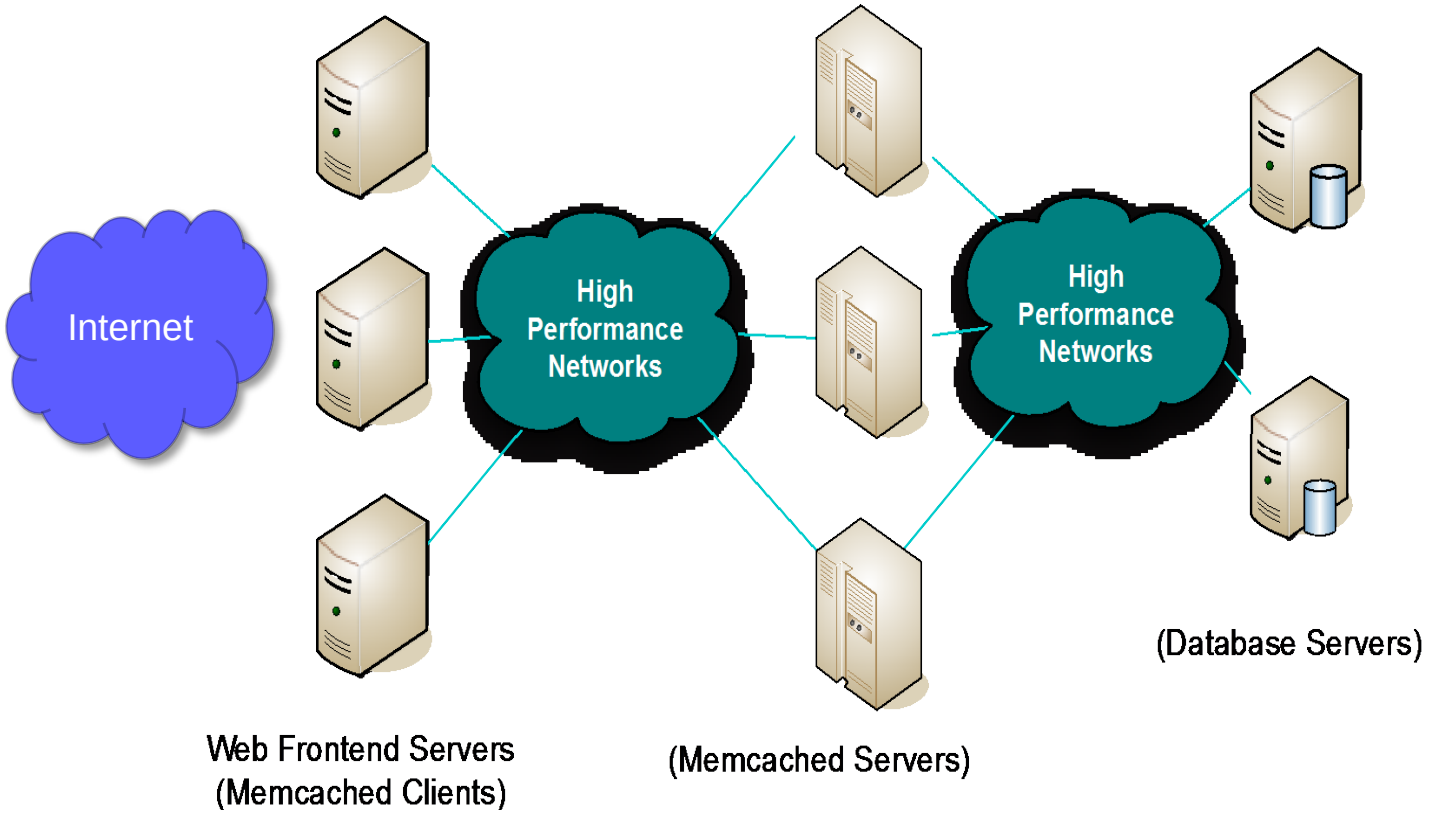
(HBase Architecture)

Network-Level Interaction Between HBase Clients, Region Servers and Data Nodes

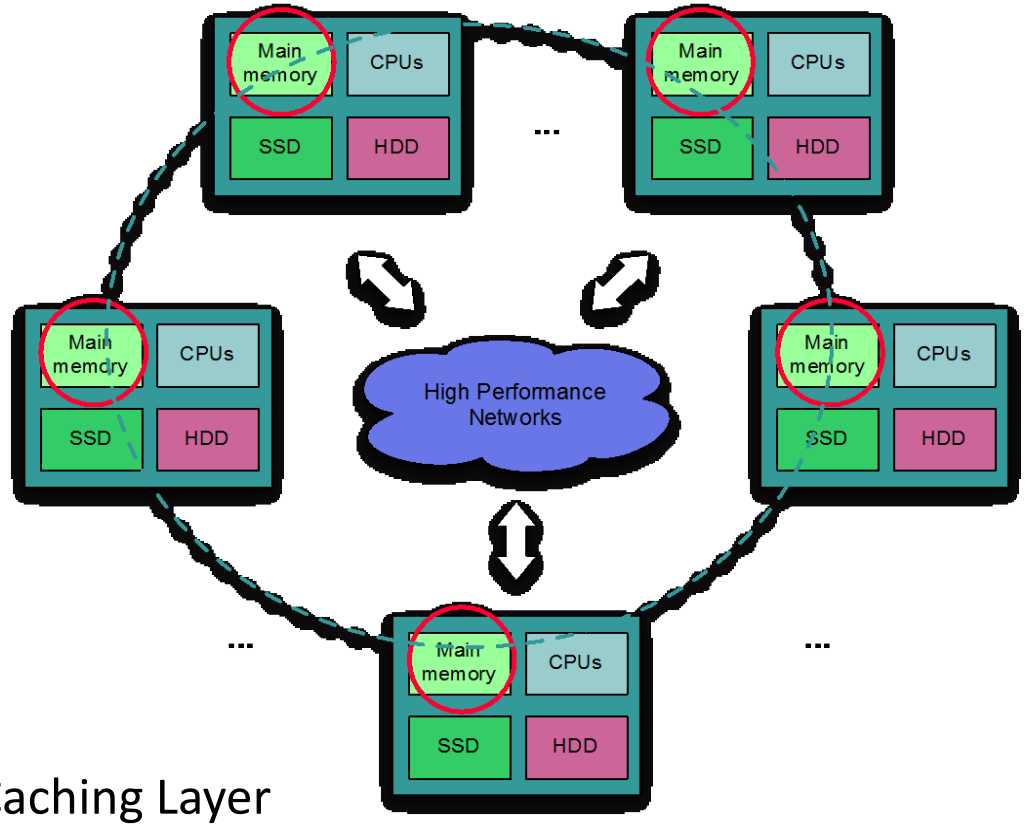


Overview of Web 2.0 Architecture and Memcached

- Three-layer architecture of Web 2.0
  - Web Servers, Memcached Servers, Database Servers
- Memcached is a core component of Web 2.0 architecture



Memcached Architecture

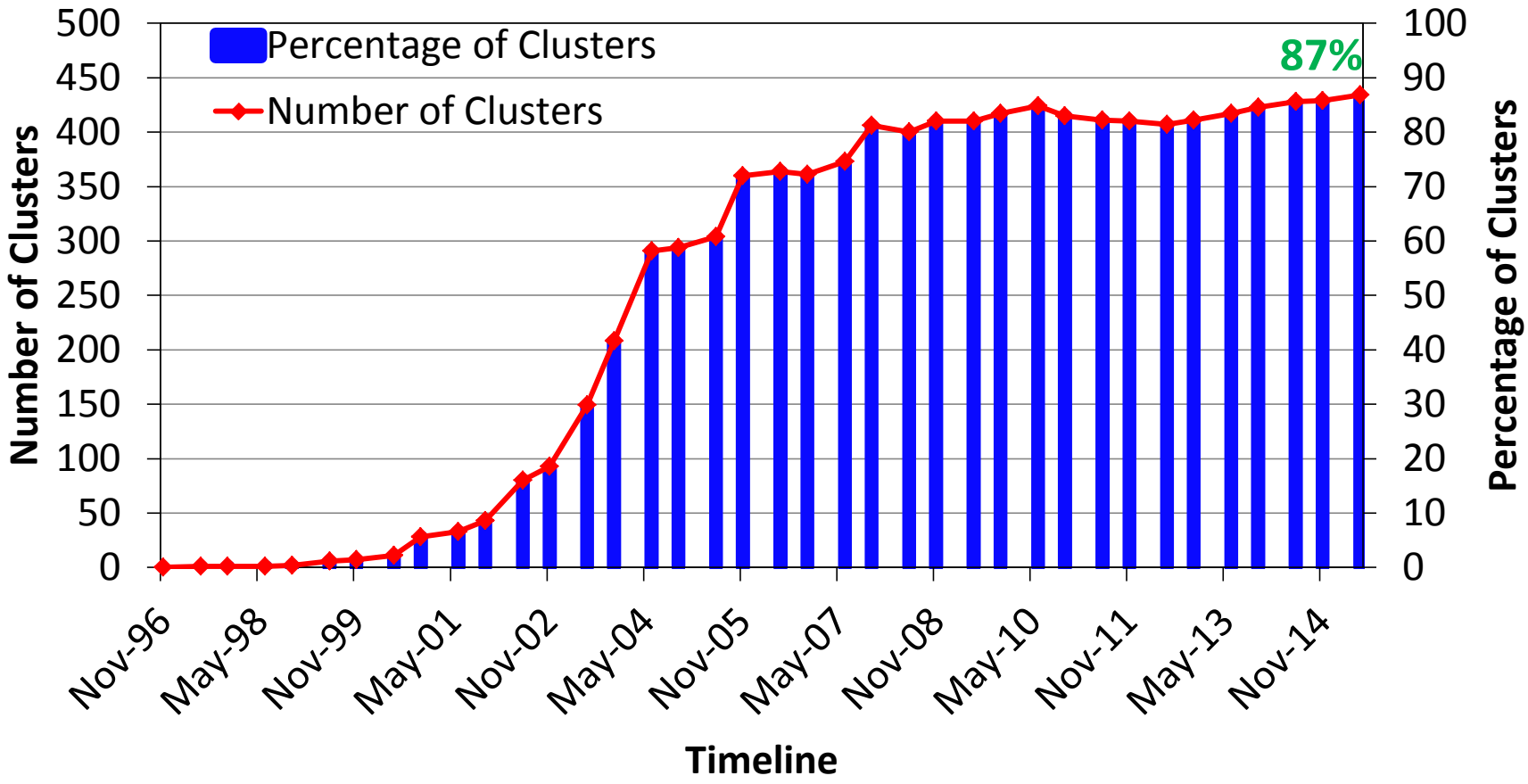


- Distributed Caching Layer
  - Allows to aggregate spare memory from multiple nodes
  - General purpose
- Typically used to cache database queries, results of API calls
- Scalable model, but typical usage very network intensive

Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

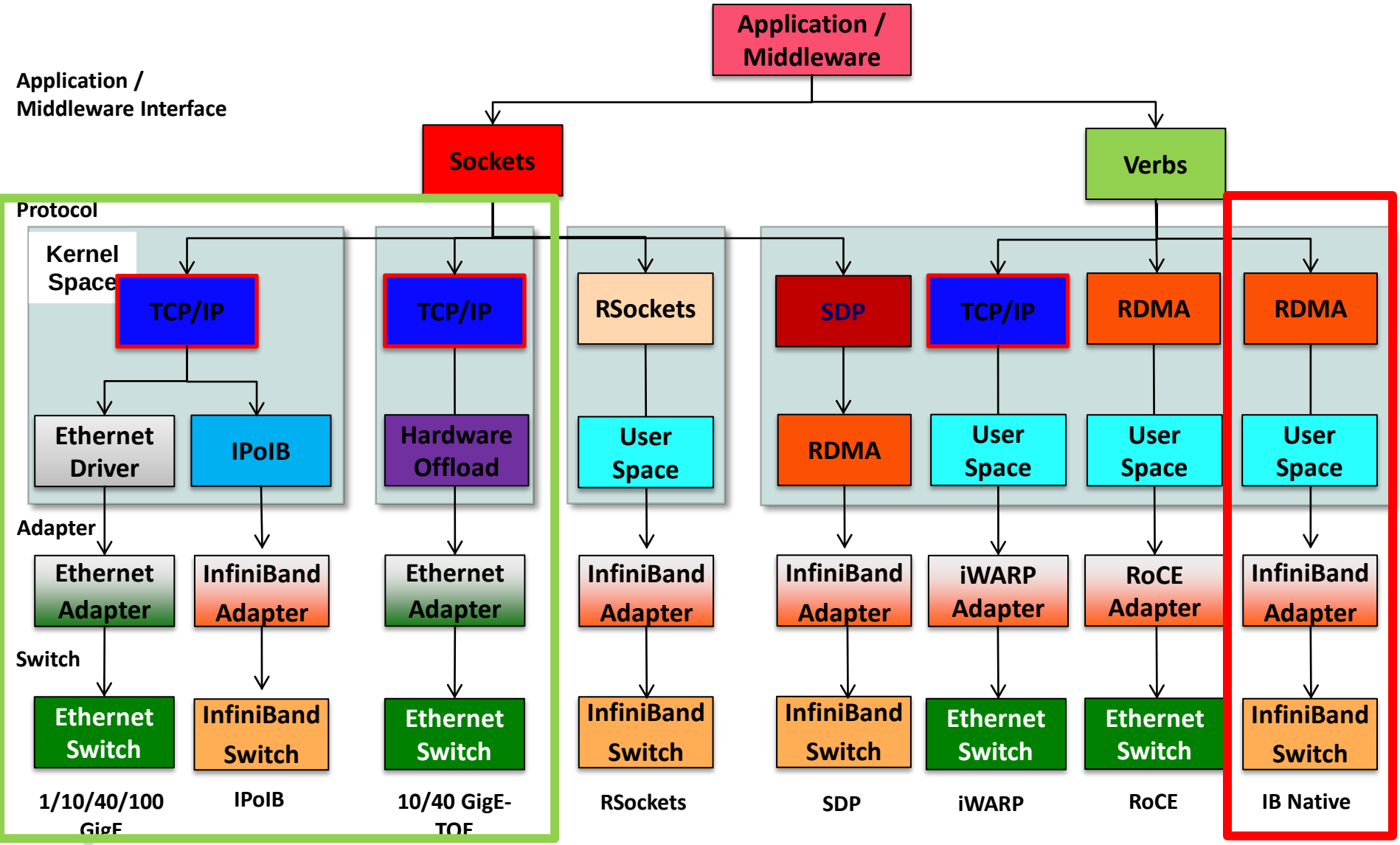
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)



Overview of High Performance Interconnects

- High-Performance Computing (HPC) has adopted advanced interconnects and protocols
  - InfiniBand
  - 10 Gigabit Ethernet/iWARP
  - RDMA over Converged Enhanced Ethernet (RoCE)
- Very Good Performance
  - Low latency (few micro seconds)
  - High Bandwidth (100 Gb/s with dual FDR InfiniBand)
  - Low CPU overhead (5-10%)
- OpenFabrics software stack with IB, iWARP and RoCE interfaces are driving HPC systems
- Many such systems in Top500 list

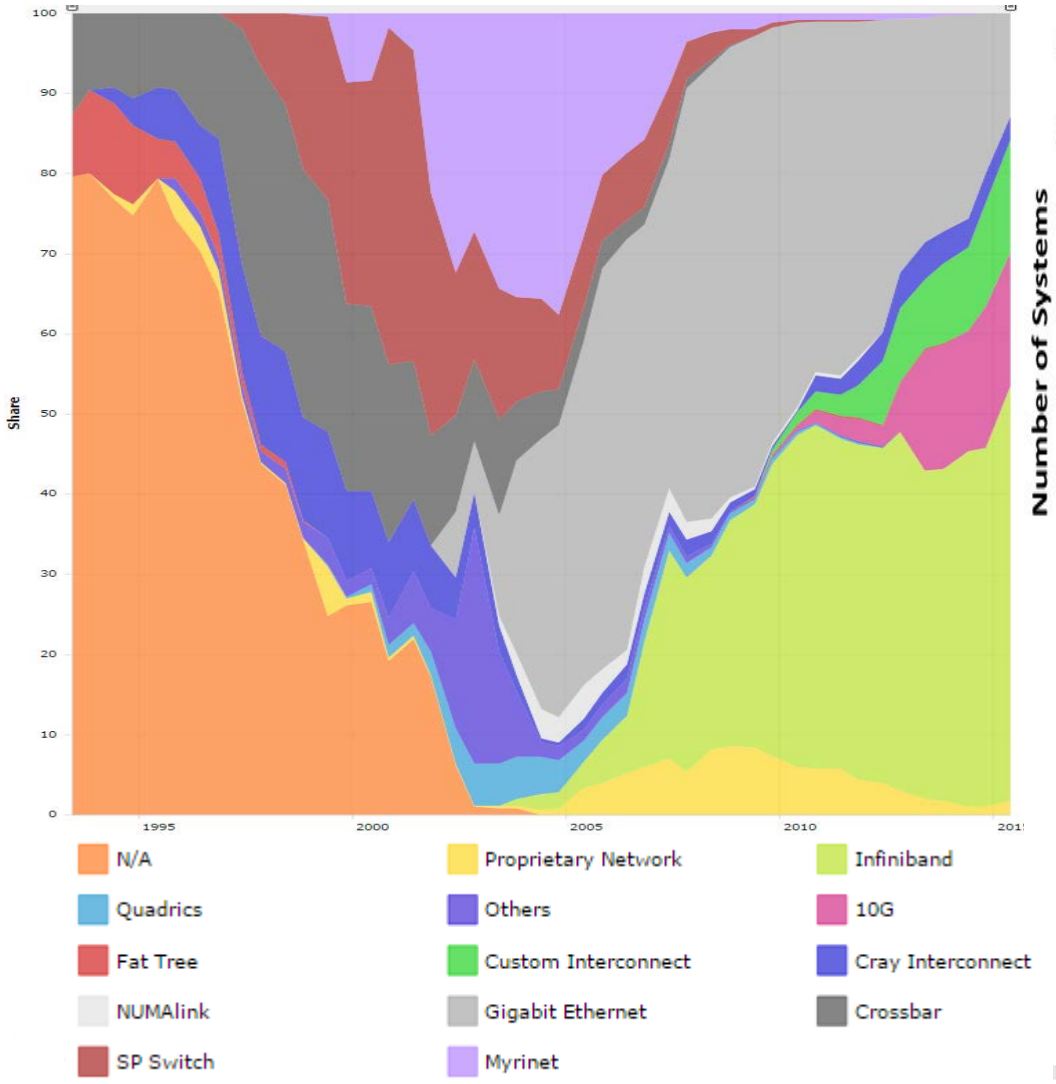
All interconnects and protocols in OpenFabrics Stack

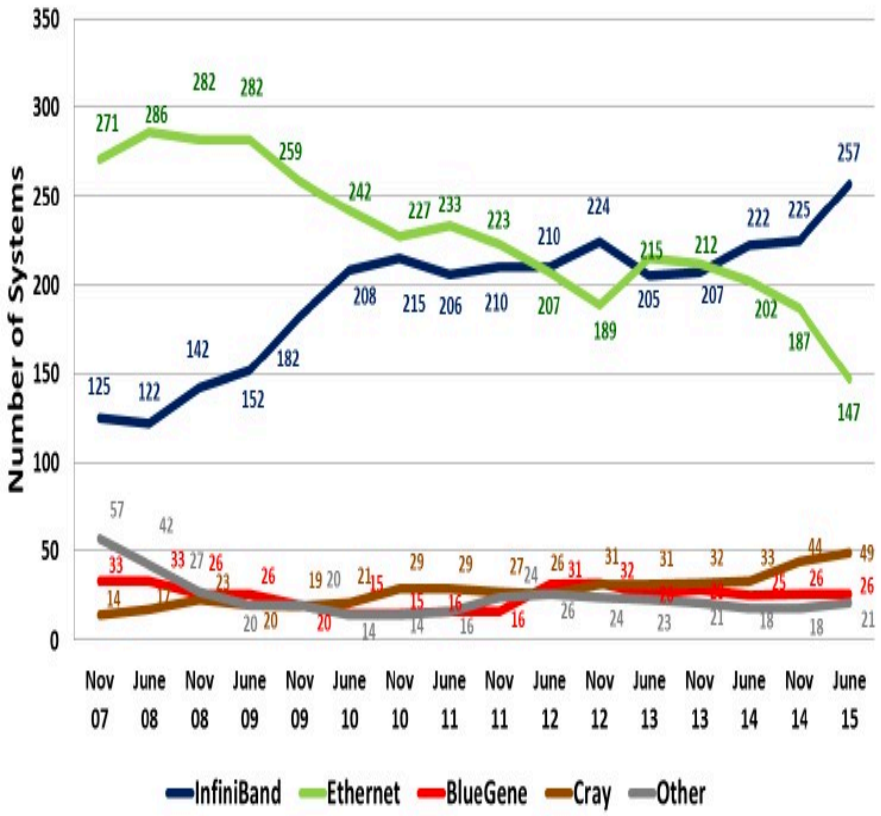


Trends of Networking Technologies in TOP500 Systems

Percentage share of InfiniBand is steadily increasing

Interconnect Family – Systems Share





Courtesy:
http://top500.org

http://www.theplatform.net/2015/07/20/ethernet-will-have-to-work-harder-to-win-hpc/

Large-scale InfiniBand Installations

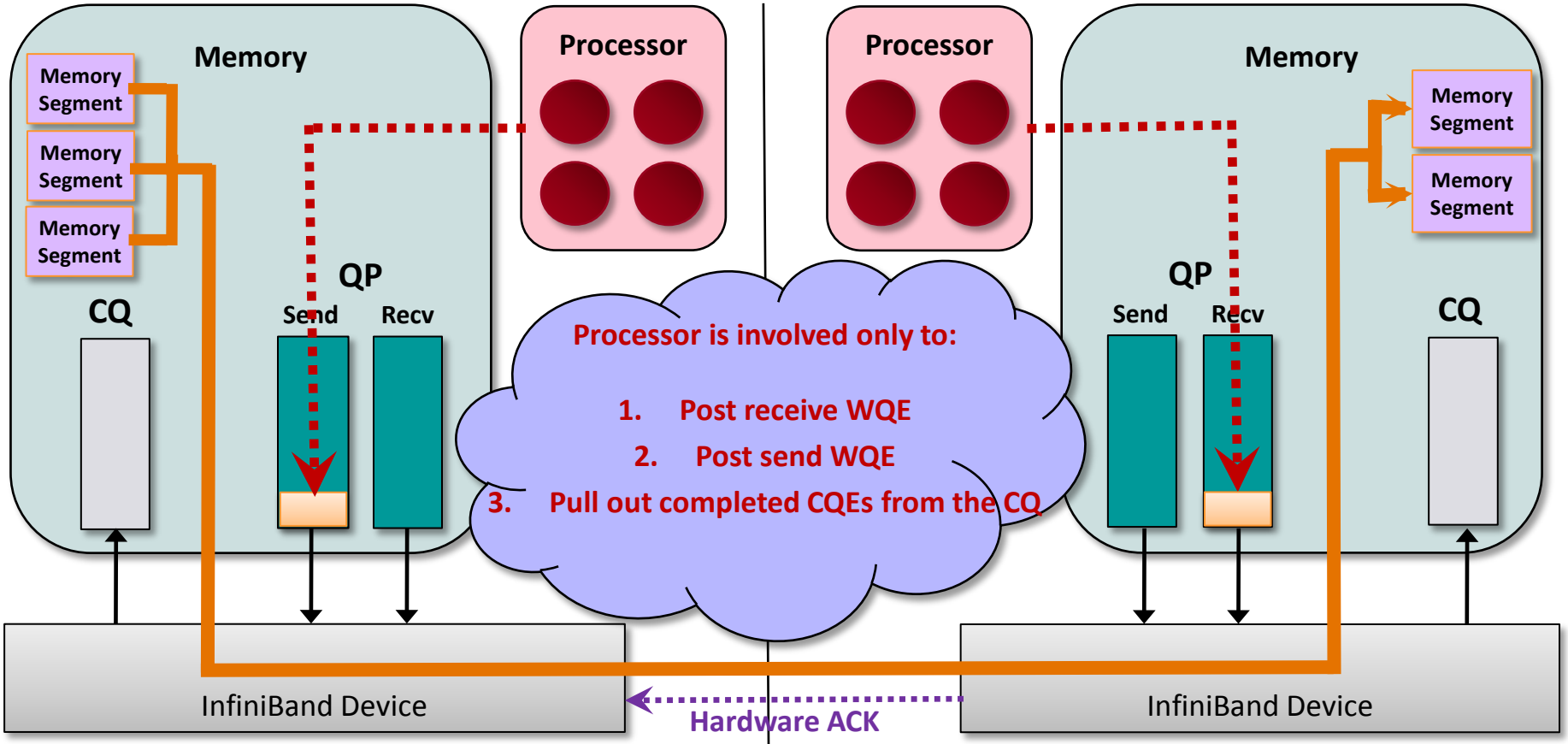
- 259 IB Clusters (51%) in the June 2015 Top500 list (http://www.top500.org)
- Installations in the Top 50 (24 systems):

519,640 cores (Stampede) at TACC (8th)	76,032 cores (Tsubame 2.5) at Japan/GSIC (22nd)
185,344 cores (Pleiades) at NASA/Ames (11th)	194,616 cores (Cascade) at PNNL (25th)
72,800 cores Cray CS-Storm in US (13th)	76,032 cores (Makman-2) at Saudi Aramco (28th)
72,800 cores Cray CS-Storm in US (14th)	110,400 cores (Pangea) in France (29th)
265,440 cores SGI ICE at Tulip Trading Australia (15th)	37,120 cores (Lomonosov-2) at Russia/MSU (31st)
124,200 cores (Topaz) SGI ICE at ERDC DSRC in US (16th)	57,600 cores (SwiftLucy) in US (33rd)
72,000 cores (HPC2) in Italy (17th)	50,544 cores (Occigen) at France/GENCI-CINES (36th)
115,668 cores (Thunder) at AFRL/USA (19th)	76,896 cores (Salomon) SGI ICE in Czech Republic (40th)
147,456 cores (SuperMUC) in Germany (20th)	73,584 cores (Spirit) at AFRL/USA (42nd)
86,016 cores (SuperMUC Phase 2) in Germany (21st)	**and many more!**

Open Standard InfiniBand Networking Technology

- Introduced in Oct 2000
- High Performance Data Transfer
  - Interprocessor communication and I/O
  - Low latency (<1.0 microsec), High bandwidth (up to 12.5 GigaBytes/sec -> 100Gbps), and low CPU utilization (5-10%)
- Flexibility for LAN and WAN communication
- Multiple Transport Services
  - Reliable Connection (RC), Unreliable Connection (UC), Reliable Datagram (RD), Unreliable Datagram (UD), and Raw Datagram
  - Provides flexibility to develop upper layers
- Multiple Operations
  - Send/Recv
  - RDMA Read/Write
  - Atomic Operations (very unique)
    - high performance and scalable implementations of distributed locks, semaphores, collective communication operations
- Leading to big changes in designing HPC clusters, file systems, cloud computing systems, grid computing systems, ….

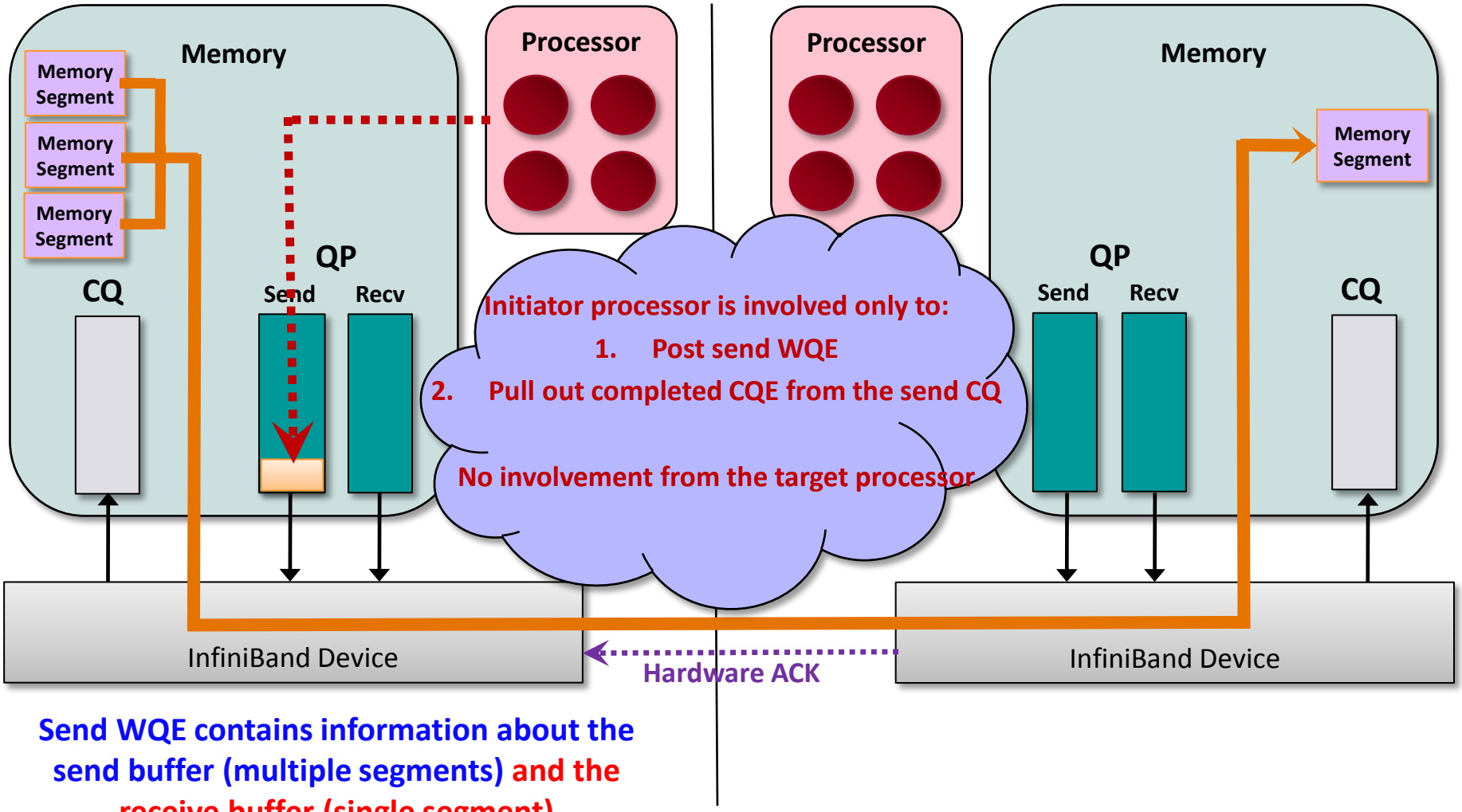
Communication in the Channel Semantics (Send/Receive Model)



Send WQE contains information about the send buffer (multiple non-contiguous segments)

Receive WQE contains information on the receive buffer (multiple non-contiguous segments); Incoming messages have to be matched to a receive WQE to know where to place

Communication in the Memory Semantics (RDMA Model)



Send WQE contains information about the send buffer (multiple segments) and the receive buffer (single segment)

Feature Comparison: IB, IWARP/HSE and RoCE

Features	IB	iWARP/HSE	RoCE
Hardware Acceleration	Yes	Yes	Yes
RDMA	Yes	Yes	Yes
Congestion Control	Yes	Optional	Yes
Multipathing	Yes	Yes	Yes
Atomic Operations	Yes	No	Yes
Multicast	Optional	No	Optional
Data Placement	Ordered	Out-of-order	Ordered
Prioritization	Optional	Optional	Yes
Fixed BW QoS (ETS)	No	Optional	Yes
Ethernet Compatibility	No	Yes	Yes
TCP/IP Compatibility	Yes (using IPoIB)	Yes	Yes (using IPoIB)

Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

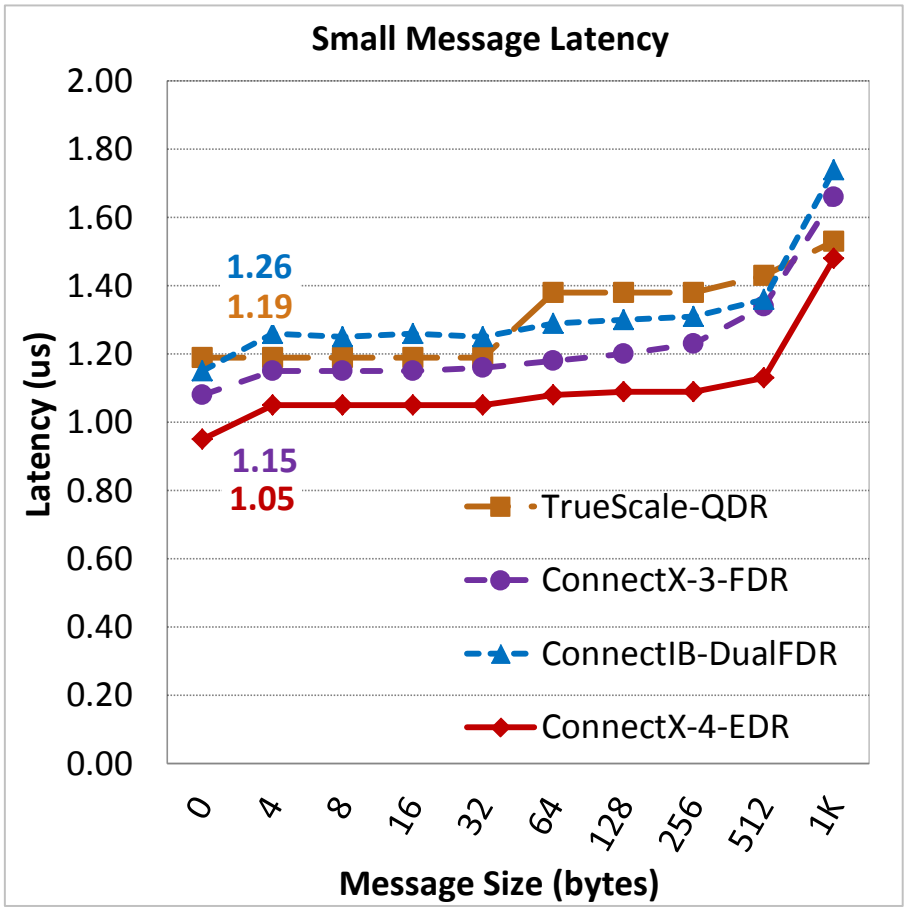
Wide Adaptation of RDMA Technology

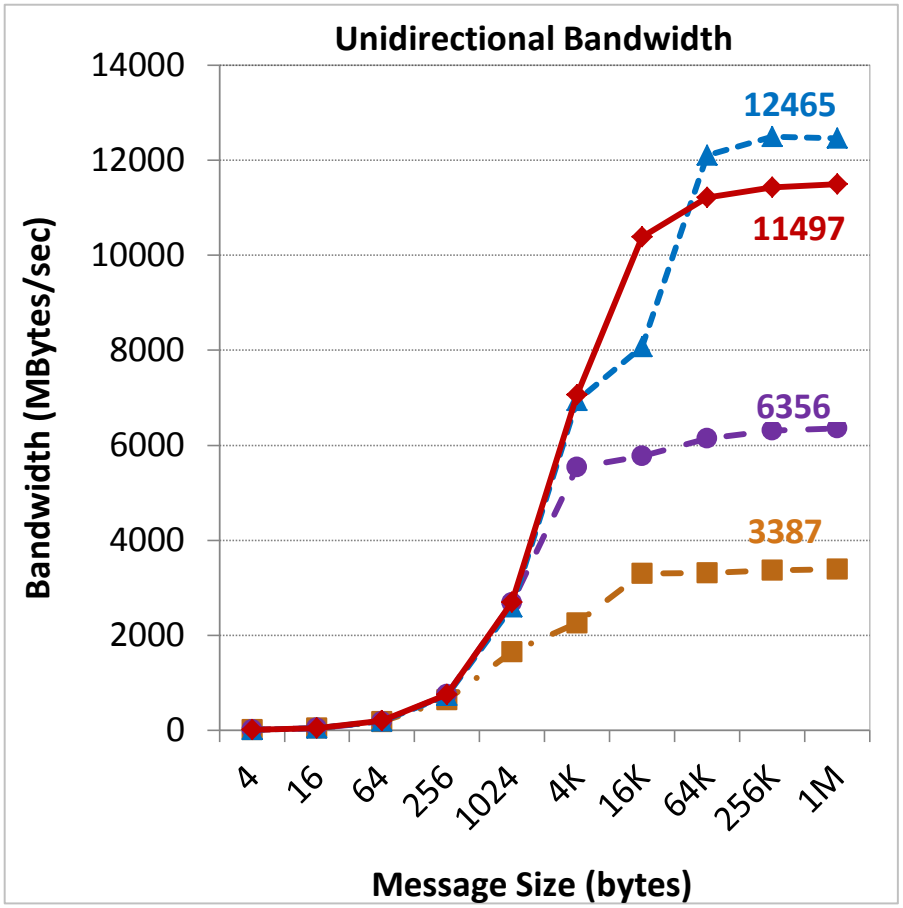
- Message Passing Interface (MPI) for HPC
- Parallel File Systems
  - Lustre
  - GPFS
- Delivering excellent performance (latency, bandwidth and CPU Utilization)
- Delivering excellent scalability

MVAPICH2 Software

- High Performance open-source MPI Library for InfiniBand, 10-40Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - **Used by more than 2,450 organizations in 76 countries**
  - **More than 282,000 downloads from the OSU site directly**
  - Empowering many TOP500 clusters (June '15 ranking)
    - 8th ranked 519,640-core cluster (Stampede) at TACC
    - 11th ranked 185,344-core cluster (Pleiades) at NASA
    - 22nd ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu
- Empowering Top500 systems for over a decade
  - System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  - Stampede at TACC (8th in Jun'15, 462,462 cores, 5.168 Plops)

Latency & Bandwidth: MPI over IB with MVAPICH2





TrueScale-QDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

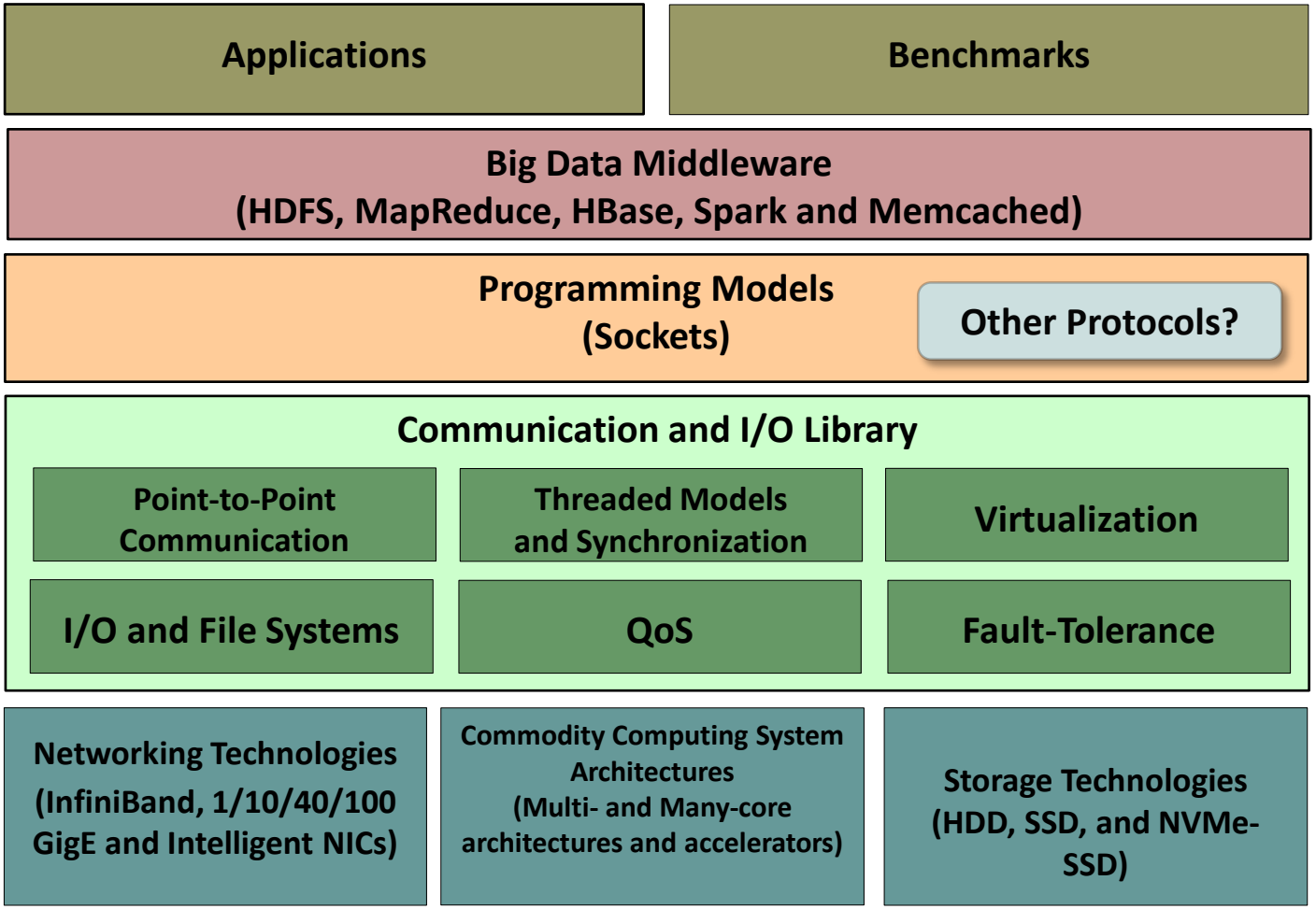
ConnectIB-Dual FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch

ConnectX-4-EDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 Back-to-back

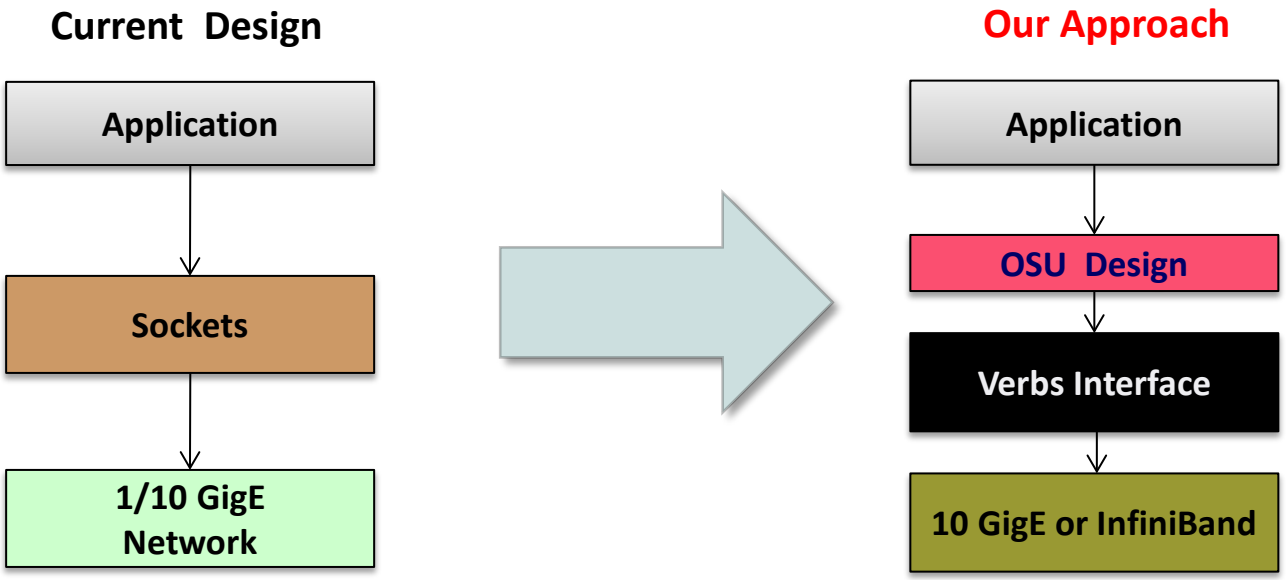
Can High-Performance Interconnects Benefit Big Data Computing?

- Most of the current Big Data systems use Ethernet Infrastructure with Sockets
- Concerns for performance and scalability
- Usage of High-Performance Networks is beginning to draw interest
  - Oracle, IBM, Google, Intel are working along these directions
- What are the challenges?
- Where do the bottlenecks lie?
- Can these bottlenecks be alleviated with new designs (similar to the designs adopted for MPI)?
- Can HPC Clusters with High-Performance networks be used for Big Data applications using Hadoop and Memcached?

Designing Communication and I/O Libraries for Big Data Systems: Challenges



Can Big Data Processing Systems be Designed with High-Performance Networks and Protocols?



- Sockets not designed for high-performance
  - Stream semantics often mismatch for upper layers
  - Zero-copy not available for non-blocking sockets

Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

Overview of Benchmarks using Hadoop and Memcached

- Hadoop Benchmarks
  - TestDFSIO, Enhanced TestDFSIO, RandomWriter, Sort, TeraGen, TeraSort
  - PUMA
- NoSQL Benchmarks
  - YCSB
- Spark Benchmarks
  - GroupBy, PageRank, K-means
- Memcached Benchmark
  - Olio
- OSU HiBD Micro-Benchmark (OHB) Suite
  - HDFS, MapReduce, RPC, Memcached
- BigDataBench

Hadoop Benchmarks

- **TestDFSIO**
  - HDFS benchmark; measures the I/O performance of HDFS
  - Includes Read and Write tests
  - Input: No. of files (to be read or written to), size of each file
  - Output: Average throughput per map, Average I/O rate per map, Job execution time
- **Enhanced TestDFSIO in Intel HiBench**
  - Attempts to report an aggregate performance curve, rather than just a summary rate at the end
    - Computes the aggregated throughput by sampling the number of bytes read/written at fixed time intervals in each map task
    - In the reduce stage, the samples of each map task are re-sampled at a fixed plot rate to compute the aggregated read/write throughput by all the map tasks
  - Input: No. of files (to be read or written to), size of each file
  - Output: Aggregated throughput, Job execution time

Hadoop Benchmarks

- **RandomWriter, Sort**
  - MapReduce benchmark; RandomWriter generates the data for Sort
  - The Sort benchmark uses the MapReduce framework to sort the input directory into the output directory. The input/output format is *SequenceFile* where the key/value format is *BytesWritable*

- **TeraGen, TeraSort**
  - MapReduce benchmark; TeraGen generates the input data for TeraSort
  - TeraSort is implemented as a MapReduce sort job with a custom partitioner (TotalOrderPartitioner) that uses a sorted list of sampled keys that define the key range for each reduce

PUMA

- PUrdue MApReduce benchmarks suite (a total of 13 benchmarks)
- Represents a broad range of MapReduce applications exhibiting application characteristics with high/low computation and high/low shuffle volumes
- https://sites.google.com/site/farazahmad/pumabenchmarks

Benchmark	Description
Adjacency List	Generates adjacency and reverse adjacency lists of nodes of a graph for use by PageRank-like algorithms
Self Join	Generates association among k+1 fields given the set of k-field associations
Word Count	Counts the occurrences of each word in a large collection of documents
Inverted Index	Generates word-to-document indexing from a list of documents
Sequence Count	Generates a count of all unique sets of three consecutive words per document in the input data

Yahoo! Cloud Serving Benchmark – YCSB

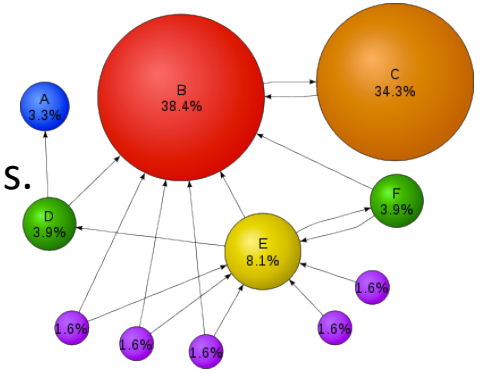
- Benchmarking systems including different No-SQL databases, like HBase, Cassandra, MongoDB, etc.
- Consists of six different workloads for cloud systems

Workload	Function
A	50% read -50% write
B	95% read – 5% write
C	100% read
D	New records are inserted and read
E	Short ranges of records are queried, instead of individual
F	Read a record, modify it, and write back the changes

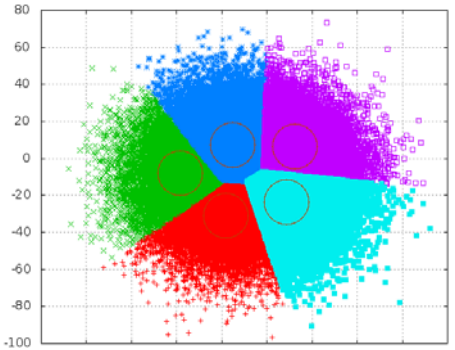
https://github.com/brianfrankcooper/YCSB/wiki

Spark Benchmarks

- **GroupBy**
  - A commonly used micro-benchmark that uses the groupByKey operation, which groups the values for each key in the RDD into a single sequence and results in data shuffle
- **PageRank**
  - Counts the number and quality of links to a page to determine a rough estimate of how important the website is. A typical search engine benchmark



- **K-means**
  - Partitions the input objects to k clusters by calculating the nearest mean cluster of each object belongs to. A typical social network benchmark



- **Sort/Grep/WordCount**
  - Spark-based implementations for micro-benchmarks

OSU HiBD Micro-Benchmark (OHB) Suite - HDFS

- Evaluate the performance of standalone HDFS
- Five different benchmarks
  - Sequential Write Latency (**SWL**)
  - Sequential or Random Read Latency (**SRL** or **RRL**)
  - Sequential Write Throughput (**SWT**)
  - Sequential Read Throughput (**SRT**)
  - Sequential Read-Write Throughput (**SRWT**)

N. S. Islam, X. Lu, M. W. Rahman, J. Jose, and D. K. Panda, A Micro-benchmark Suite for Evaluating HDFS Operations on Modern Clusters, Int'l Workshop on Big Data Benchmarking (WBDB '12), December 2012.

Benchmark	File Name	File Size	HDFS Parameter	Readers	Writers	Random/ Sequential Read	Seek Interval
SWL	√	√	√				
SRL/RRL	√	√	√			√	√ (RRL)
SWT		√	√		√		
SRT		√	√	√			
SRWT		√	√	√	√		

OSU HiBD Micro-Benchmark (OHB) Suite - MapReduce

- Evaluate the performance of stand-alone MapReduce
- Does not require or involve HDFS or any other 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.
- Three different micro-benchmarks based on intermediate shuffle data patterns
  - **MR-AVG micro-benchmark:** intermediate data is evenly distributed among reduce tasks.
  - **MR-RAND micro-benchmark:** intermediate data is pseudo-randomly distributed among reduce tasks.
  - **MR-SKEW micro-benchmark:** intermediate data is unevenly distributed among reduce tasks.

D. Shankar, X. Lu, M. W. Rahman, N. Islam, and D. K. Panda, A Micro-Benchmark Suite for Evaluating Hadoop MapReduce on High-Performance Networks, BPOE-5 (2014).

OSU HiBD Micro-Benchmark (OHB) Suite - RPC

- Two different micro-benchmarks to evaluate the performance of standalone Hadoop RPC
  - Latency: Single Server, Single Client
  - Throughput: Single Server, Multiple Clients
- A simple script framework for job launching and resource monitoring
- Calculates statistics like Min, Max, Average
- Network configuration, Tunable parameters, DataType, CPU Utilization

Component	Network Address	Port	Data Type	Min Msg Size	Max Msg Size	No. of Iterations	Handlers	Verbose
lat_client	√	√	√	√	√	√		√
lat_server	√	√					√	√

Component	Network Address	Port	Data Type	Min Msg Size	Max Msg Size	No. of Iterations	No. of Clients	Handlers	Verbose
thr_client	√	√	√	√	√	√			√
thr_server	√	√			√		√	√	√

X. Lu, M. W. Rahman, N. Islam, and D. K. Panda, A Micro-Benchmark Suite for Evaluating Hadoop RPC on High-Performance Networks, Int'l Workshop on Big Data Benchmarking (WBDB '13), July 2013.

OSU HiBD Micro-Benchmark (OHB) Suite - Memcached

- Evaluates the performance of stand-alone Memcached
- Three different micro-benchmarks
  - **SET Micro-benchmark**: Micro-benchmark for memcached set operations
  - **GET Micro-benchmark**: Micro-benchmark for memcached get operations
  - **MIX Micro-benchmark**: Micro-benchmark for a mix of memcached set/get operations (Read:Write ratio is 90:10)
- Calculates average latency of Memcached operations
- Can measure throughput in Transactions Per Second

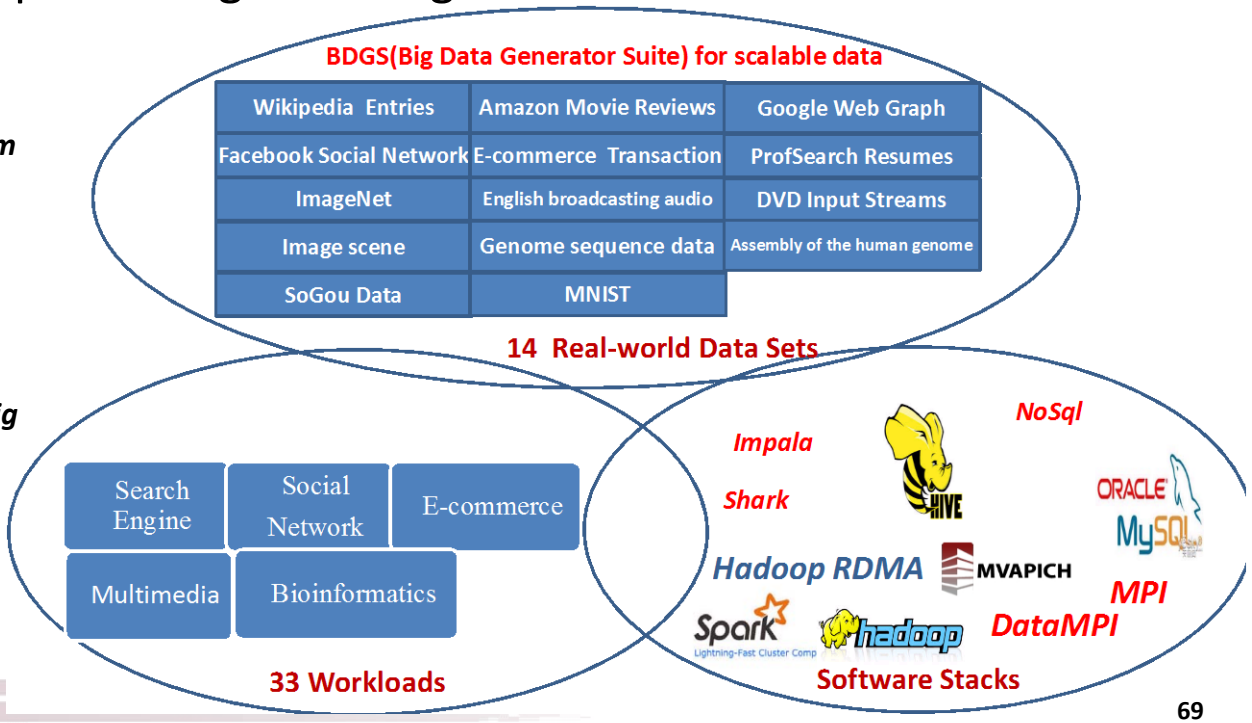
BigDataBench

- An open source project on big data benchmarking
  - http://prof.ict.ac.cn/BigDataBench
- Measuring big data architecture and systems quantitatively
- 14 real-world data sets and 33 big data workloads
- Provide a tool to generate scalable data from small or medium-scale real-world data while preserving their original characteristics

BigDataBench: a Big Data Benchmark Suite from Internet Services. HPCA 2014.

BigOP: Generating Comprehensive Big Data Workloads as a Benchmarking Framework. DASFAA 2014

BDGS: A Scalable Big Data Generator Suite in Big Data Benchmarking. WBDB 2014.



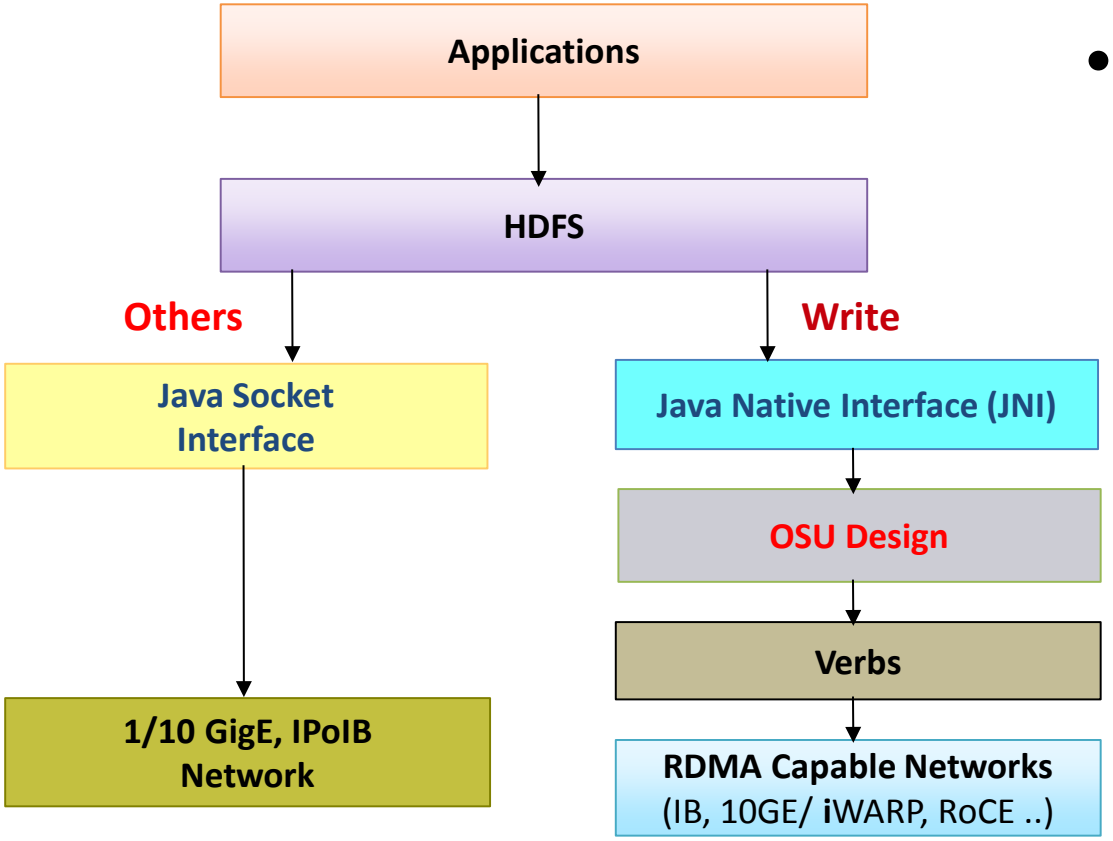
Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

Acceleration Case Studies and In-Depth Performance Evaluation

- RDMA-based Designs and Performance Evaluation
  - HDFS
  - MapReduce
  - RPC
  - Spark
  - HBase
  - Memcached

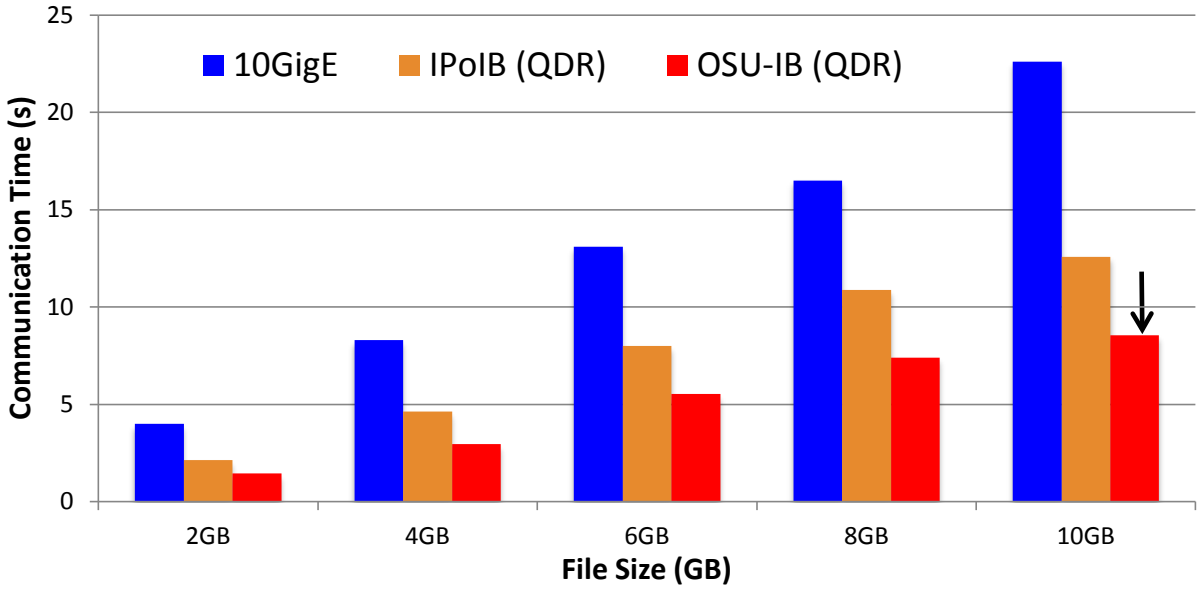
Design Overview of HDFS with RDMA



- Design Features
  - RDMA-based HDFS write
  - RDMA-based HDFS replication
  - Parallel replication support
  - On-demand connection setup
  - InfiniBand/RoCE support

- Enables high performance RDMA communication, while supporting traditional socket interface
- JNI Layer bridges Java based HDFS with communication library written in native code

Communication Times in HDFS

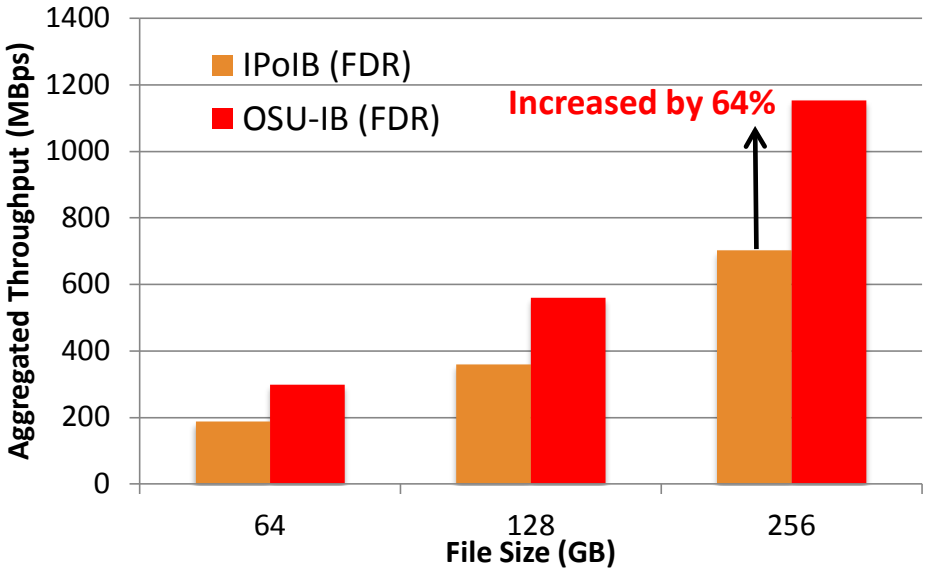


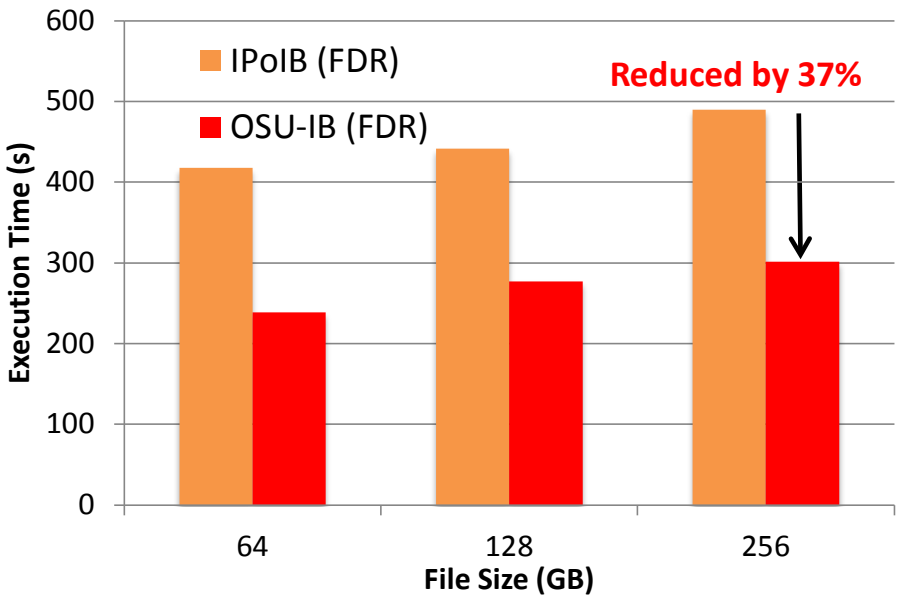
- Cluster with HDD DataNodes
  - **30%** improvement in communication time over IPoIB (QDR)
  - **56%** improvement in communication time over 10GigE
- Similar improvements are obtained for SSD DataNodes

N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy and D. K. Panda , High Performance RDMA-Based Design of HDFS over InfiniBand , Supercomputing (SC), Nov 2012

N. Islam, X. Lu, W. Rahman, and D. K. Panda, SOR-HDFS: A SEDA-based Approach to Maximize Overlapping in RDMA-Enhanced HDFS, HPDC '14, June 2014

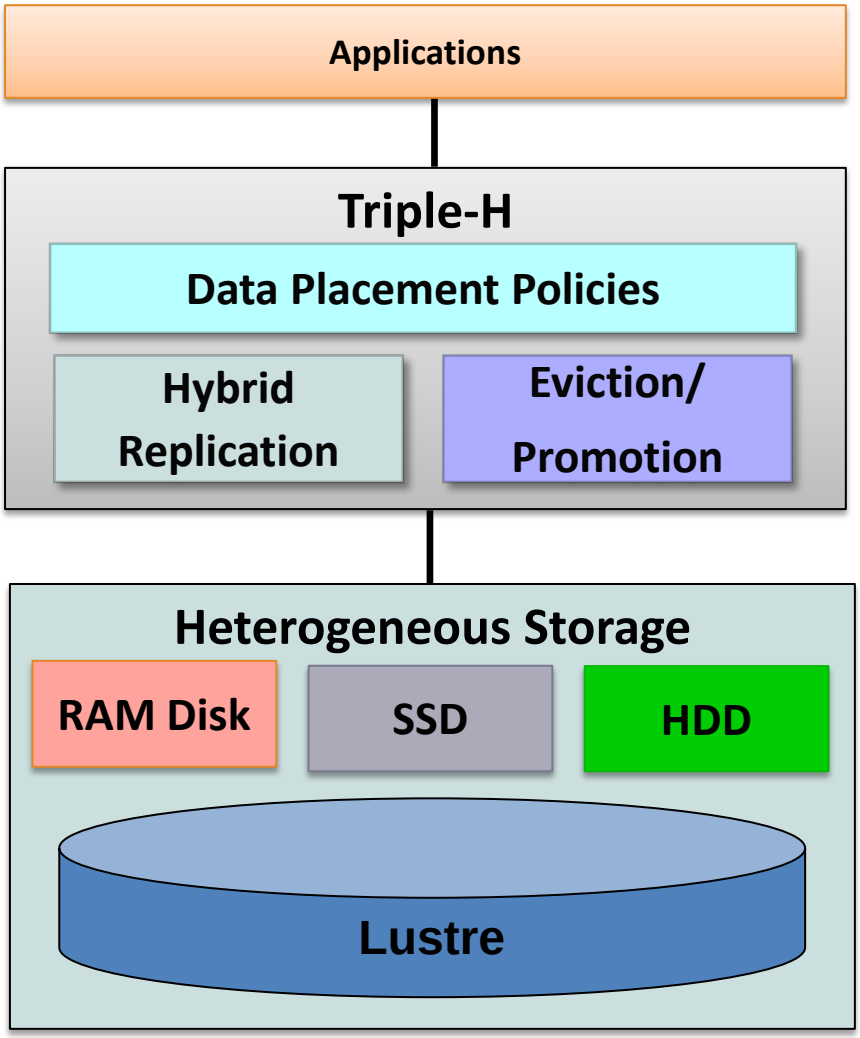
Evaluations using Enhanced DFSIO of Intel HiBench on TACC-Stampede





- Cluster with 64 DataNodes, single HDD per node
  - **64%** improvement in throughput over IPoIB (FDR) for 256GB file size
  - **37%** improvement in latency over IPoIB (FDR) for 256GB file size

Enhanced HDFS with In-memory and Heterogeneous Storage



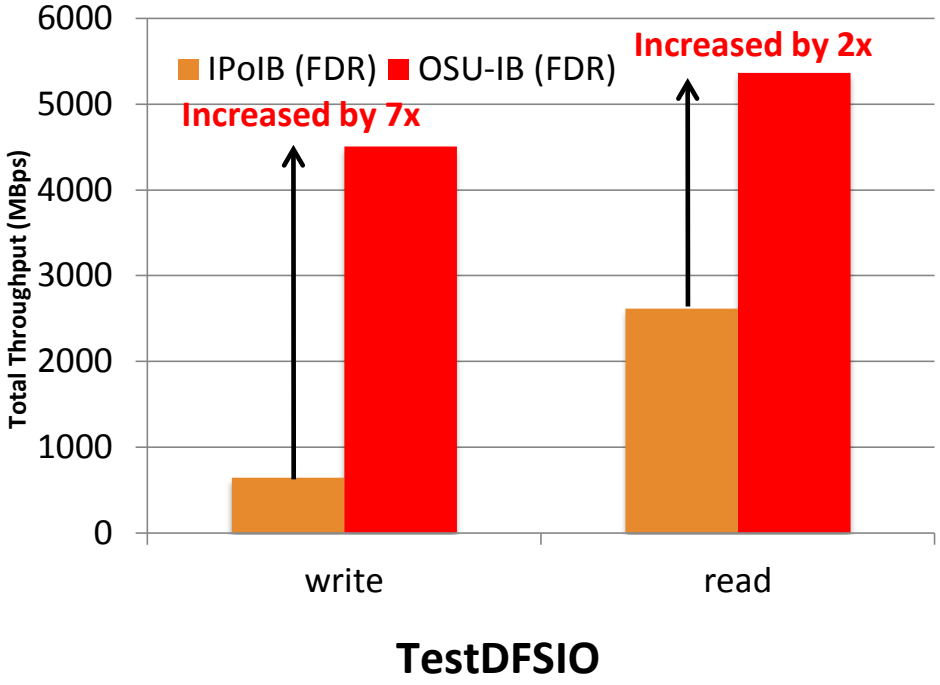
- Design Features
  - Three modes
    - Default (HHH)
    - In-Memory (HHH-M)
    - Lustre-Integrated (HHH-L)
  - Policies to efficiently utilize the heterogeneous storage devices
    - RAM, SSD, HDD, Lustre
  - Eviction/Promotion based on data usage pattern
  - Hybrid Replication
  - Lustre-Integrated mode:
    - Lustre-based fault-tolerance

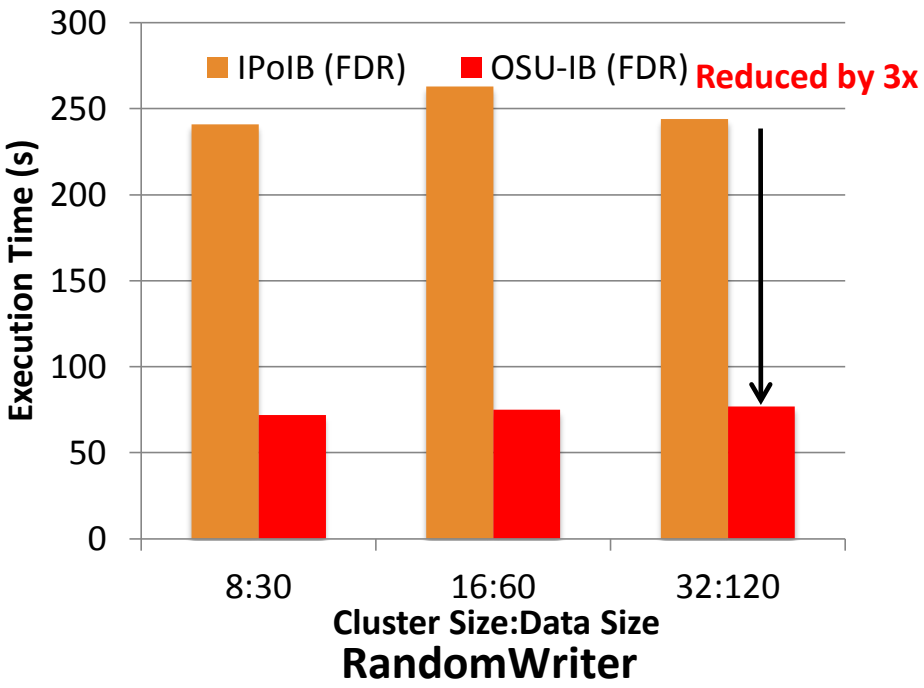
N. Islam, X. Lu, M. W. Rahman, D. Shankar, and D. K. Panda, Triple-H: A Hybrid Approach to Accelerate HDFS on HPC Clusters with Heterogeneous Storage Architecture, CCGrid '15, May 2015

Enhanced HDFS with In-memory and Heterogeneous Storage – Three Modes

- **HHH (default)**: Heterogeneous storage devices with hybrid replication schemes
  - I/O operations over RAM disk, SSD, and HDD
  - Hybrid replication (in-memory and persistent storage)
  - Better fault-tolerance as well as performance
- **HHH-M**: High-performance in-memory I/O operations
  - Memory replication (in-memory only with lazy persistence)
  - As much performance benefit as possible
- **HHH-L**: Lustre integrated
  - Take advantage of the Lustre available in HPC clusters
  - Lustre-based fault-tolerance (No HDFS replication)
  - Reduced local storage space usage

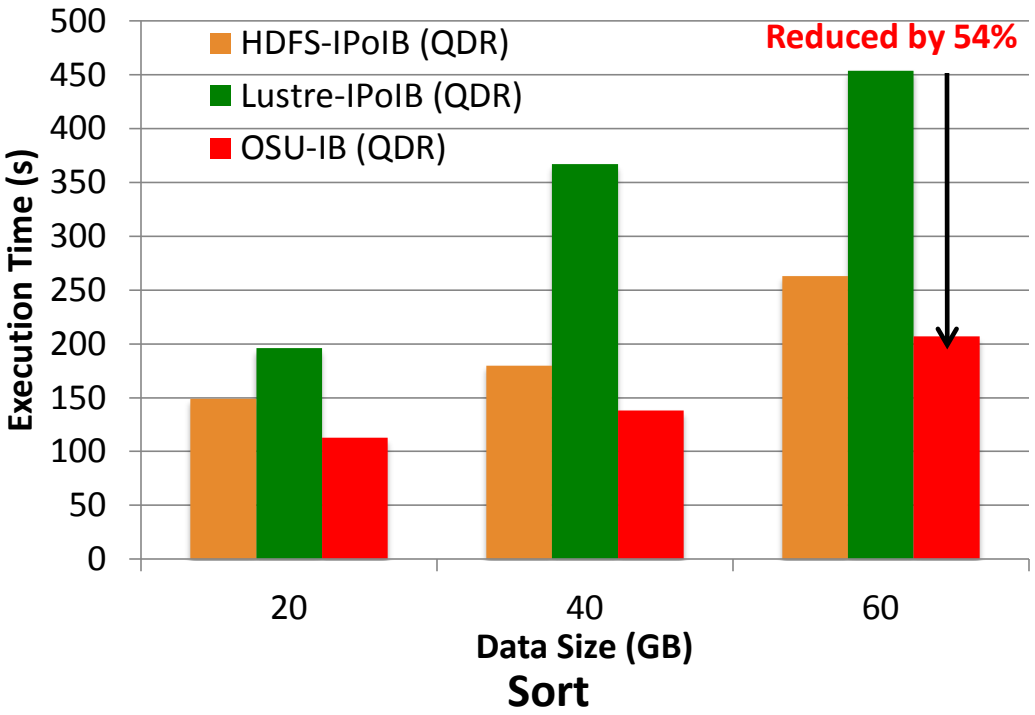
Performance Improvement on TACC Stampede (HHH)





- For 160GB TestDFSIO in 32 nodes
  - Write Throughput: 7x improvement over IPoIB (FDR)
  - Read Throughput: 2x improvement over IPoIB (FDR)
- For 120GB RandomWriter in 32 nodes
  - 3x improvement over IPoIB (QDR)

Performance Improvement on SDSC Gordon (HHH-L)

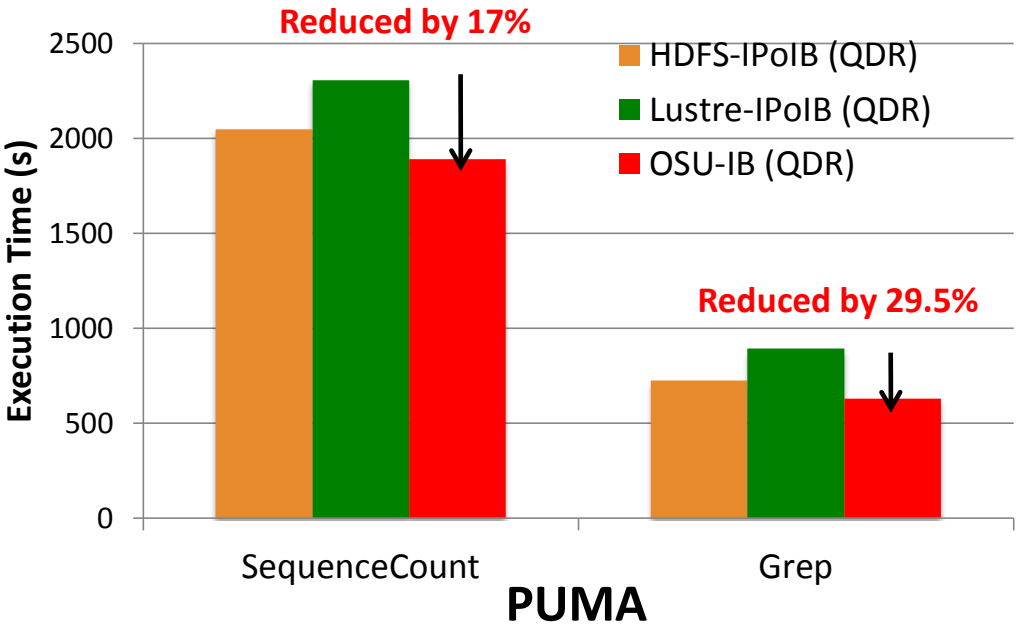


Storage Used (GB)	
HDFS-IPoIB (QDR)	360
Lustre-IPoIB (QDR)	120
OSU-IB (QDR)	240

Storage space for 60GB Sort

- For 60GB Sort in 8 nodes
  - 24% improvement over default HDFS
  - 54% improvement over Lustre
  - 33% storage space saving compared to default HDFS

Evaluation with PUMA and CloudBurst (HHH-L/HHH)

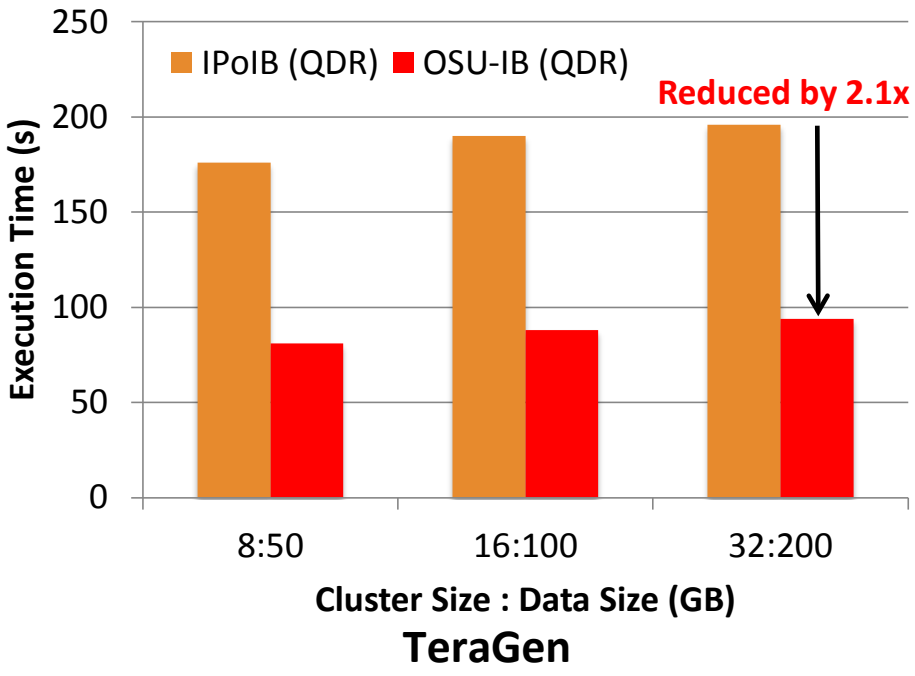


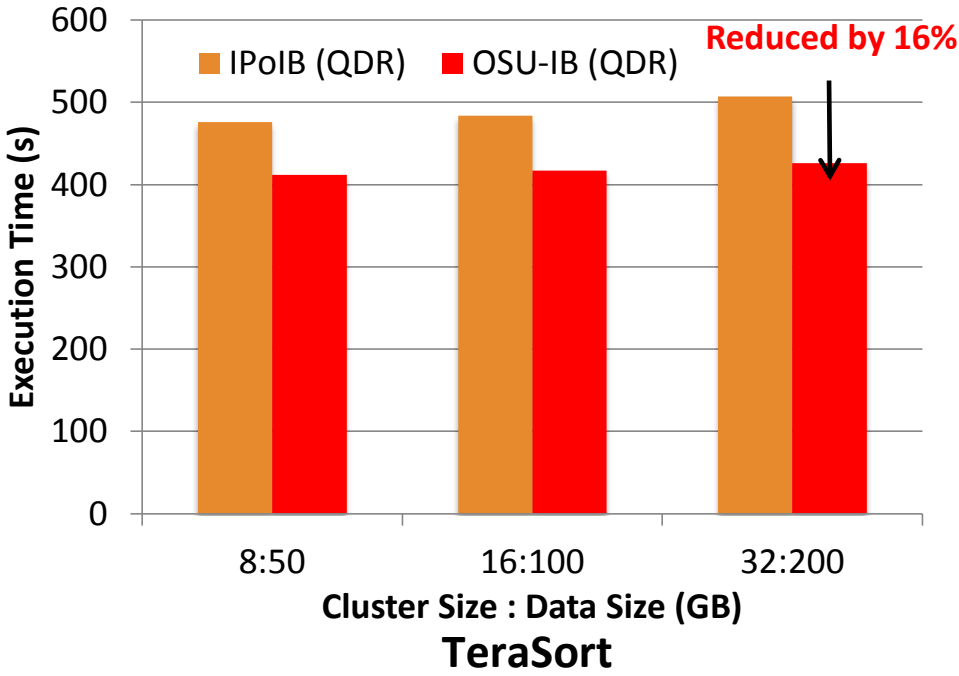
HDFS-IPoIB (FDR)	OSU-IB (FDR)
60.24 s	48.3 s

CloudBurst

- PUMA on OSU RI
  - SequenceCount with **HHH-L**: **17%** benefit over Lustre, **8%** over HDFS
  - Grep with **HHH**: **29.5%** benefit over Lustre, **13.2%** over HDFS
- CloudBurst on TACC Stampede
  - With **HHH**: **19%** improvement over HDFS

Evaluation with Spark on SDSC Gordon (HHH)



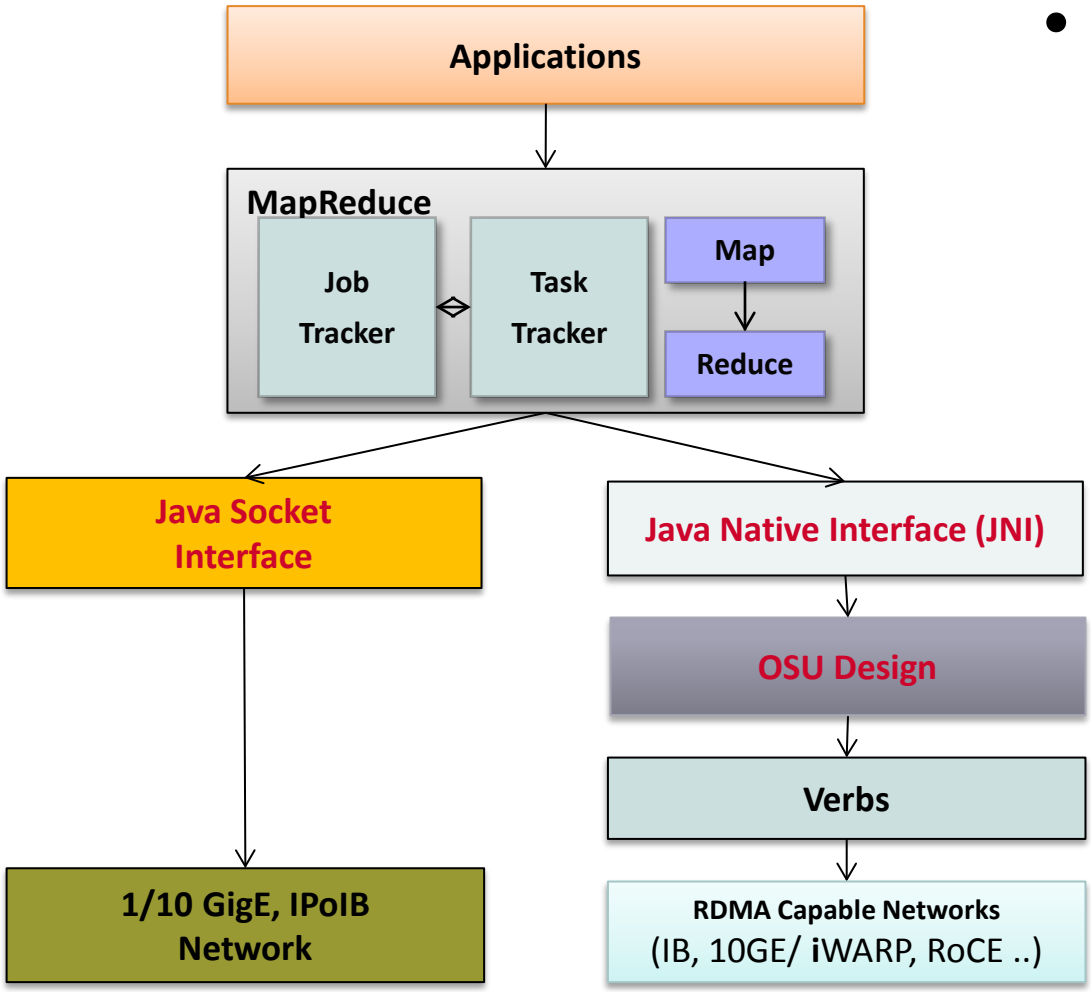


- For 200GB TeraGen on 32 nodes
  - Spark-TeraGen: HHH has **2.1x** improvement over HDFS-IPoIB (QDR)
  - Spark-TeraSort: HHH has **16%** improvement over HDFS-IPoIB (QDR)

Acceleration Case Studies and In-Depth Performance Evaluation

- RDMA-based Designs and Performance Evaluation
  - HDFS
  - MapReduce
  - RPC
  - Spark
  - HBase
  - Memcached

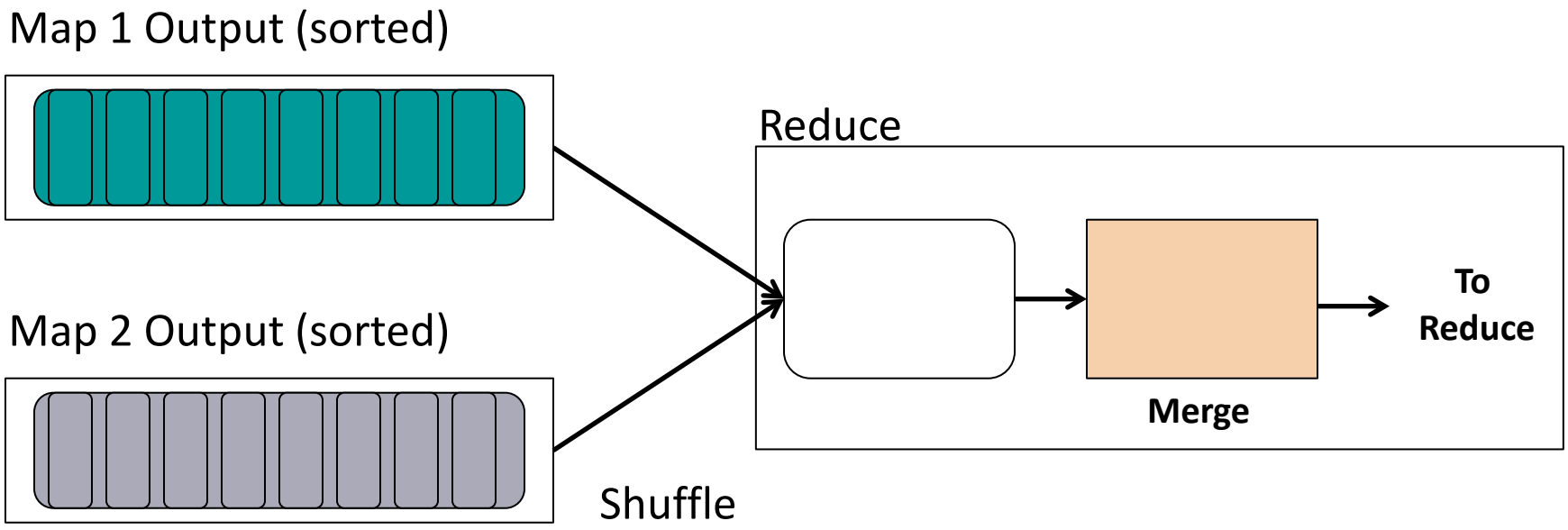
Design Overview of MapReduce with RDMA



- 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

- Enables high performance RDMA communication, while supporting traditional socket interface
- JNI Layer bridges Java based MapReduce with communication library written in native code

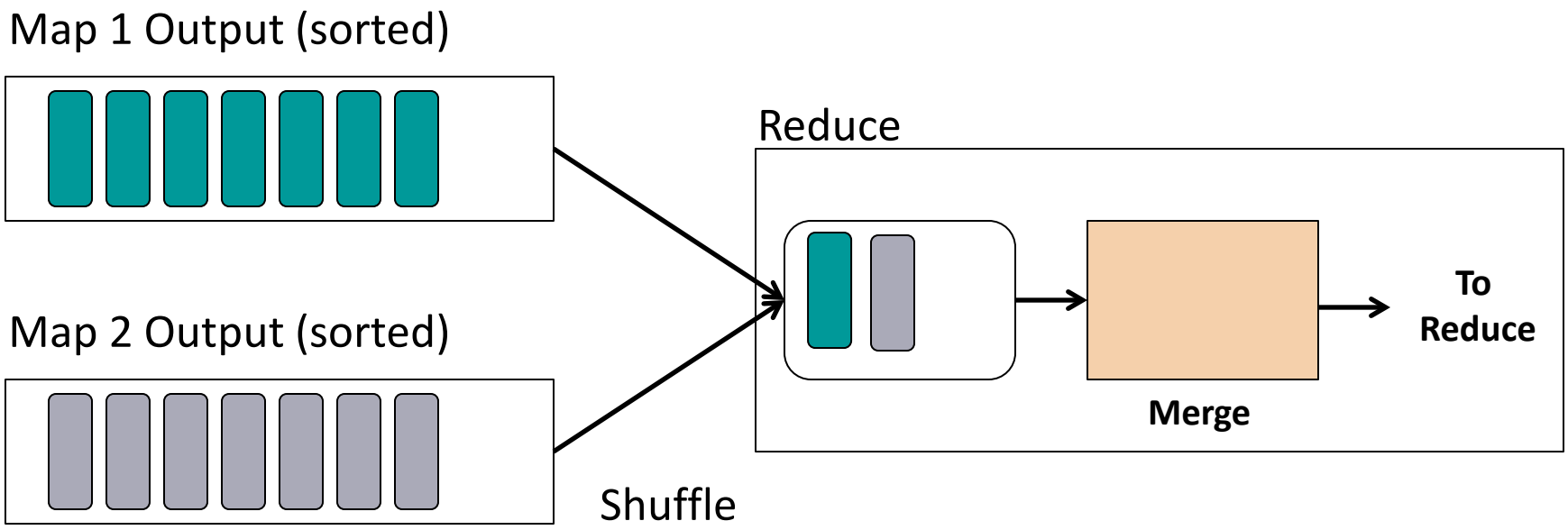
In-Memory Merge



- Sorted map output is divided into small pieces based on shuffle packet size and size of the key-value pairs
- As small portion of data is shuffled, merge can take place in memory

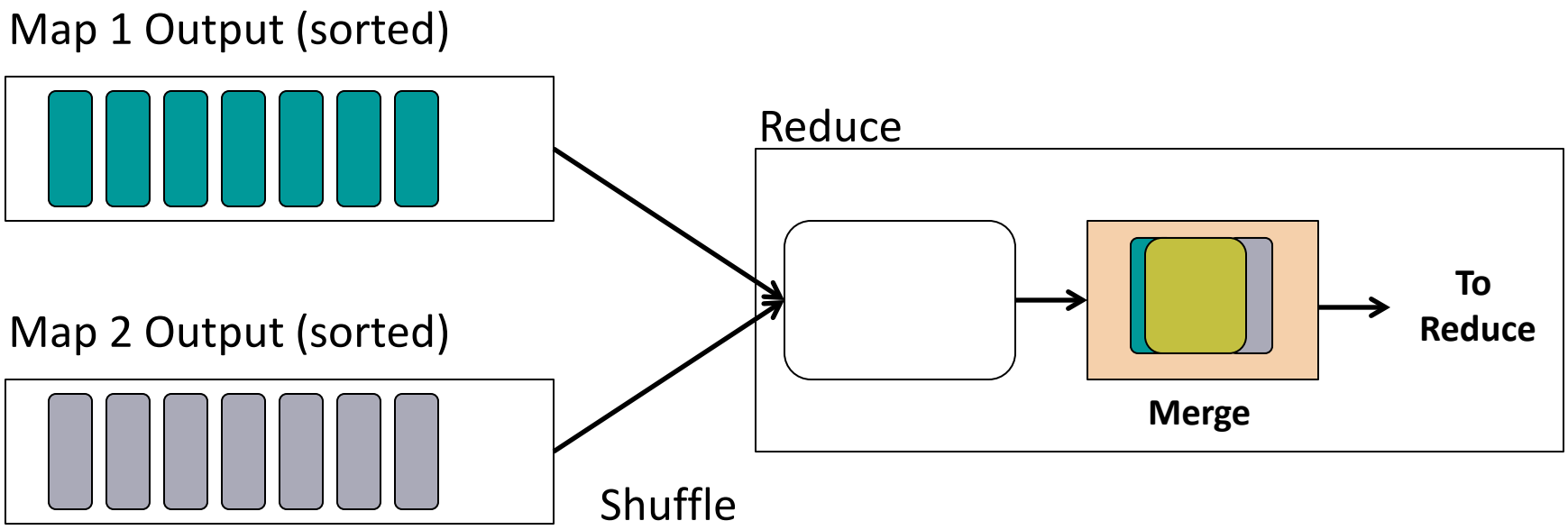
M. W. Rahman, N. S. Islam, X. Lu, J. Jose, H. Subramon, H. Wang, and D. K. Panda, High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand, HPDIC Workshop, held in conjunction with IPDPS, May 2013.

In-Memory Merge



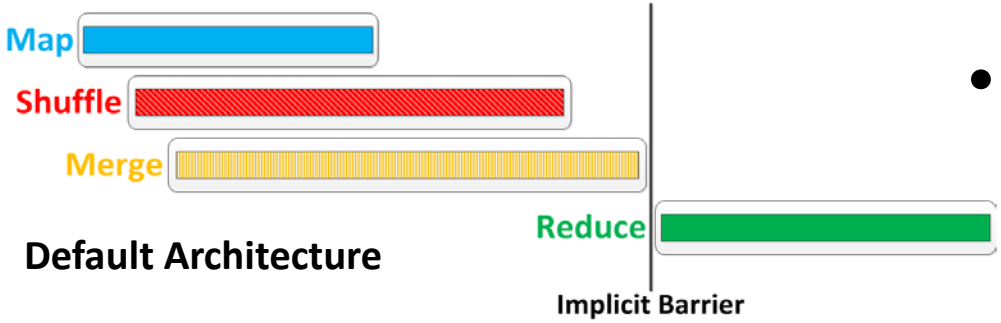
- Sorted map output is divided into small pieces based on shuffle packet size and size of the key-value pairs
- As small portion of data is shuffled, merge can take place in memory

In-Memory Merge

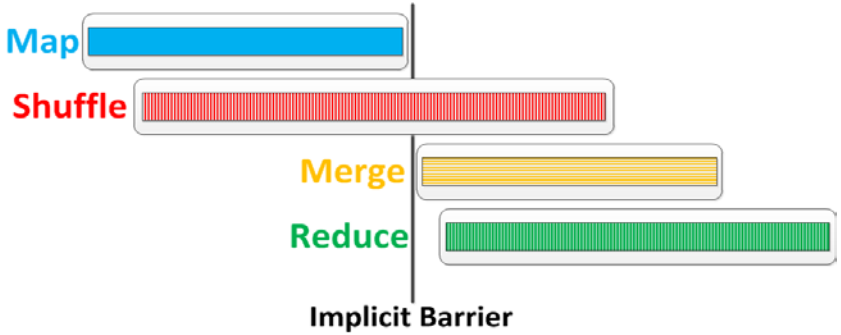


- Sorted map output is divided into small pieces based on shuffle packet size and size of the key-value pairs
- As small portion of data is shuffled, merge can take place in memory

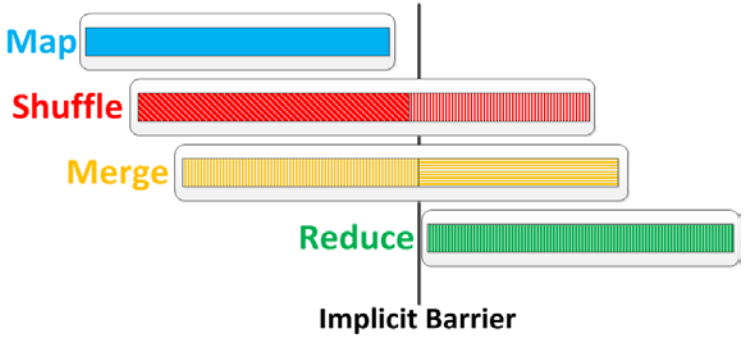
Advanced Overlapping among different phases



Default Architecture



Enhanced Overlapping

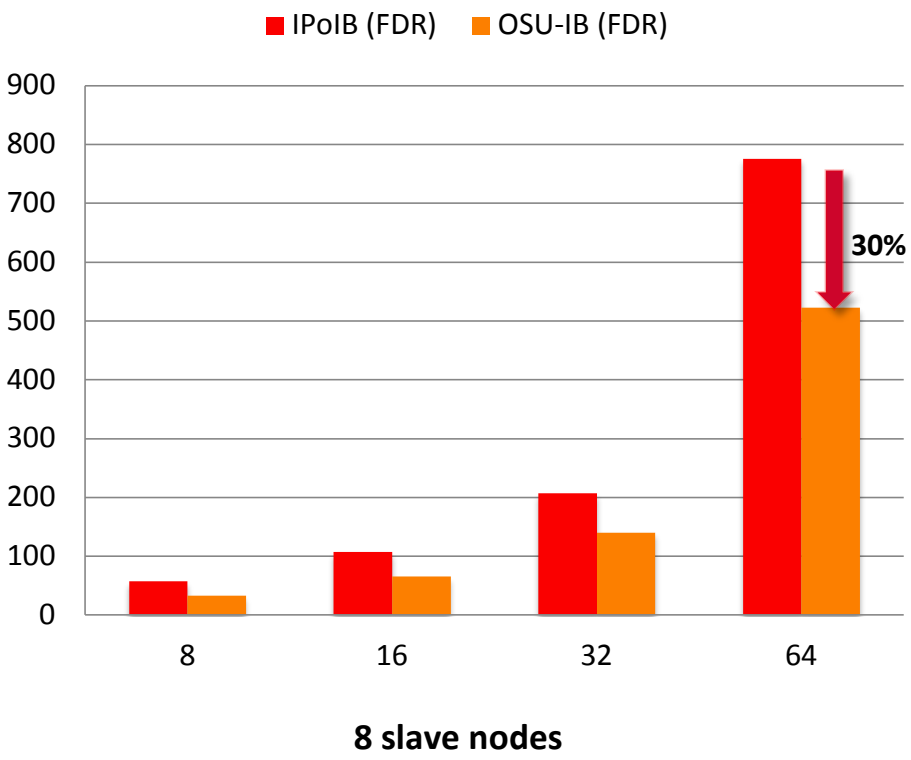


Advanced Overlapping

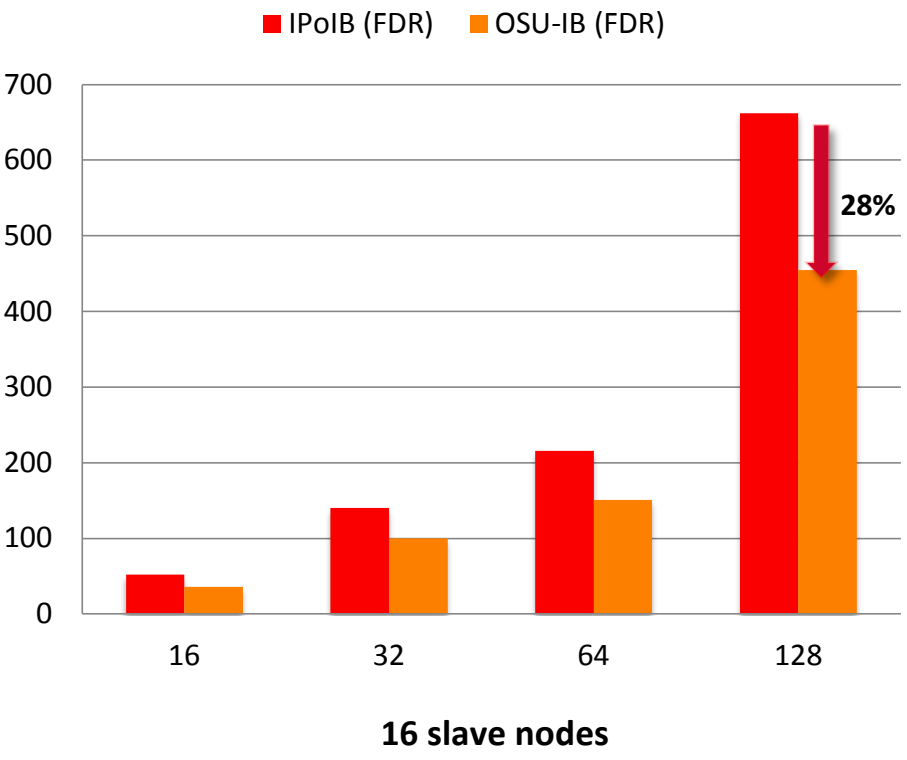
- A hybrid approach to achieve maximum possible overlapping in MapReduce across all phases compared to other approaches
  - Efficient Shuffle Algorithms
  - Dynamic and Efficient Switching
  - On-demand Shuffle Adjustment

M. W. Rahman, X. Lu, N. S. Islam, and D. K. Panda, HOMR: A Hybrid Approach to Exploit Maximum Overlapping in MapReduce over High Performance Interconnects, ICS, June 2014.

Evaluations using OHB MapReduce Micro-benchmark



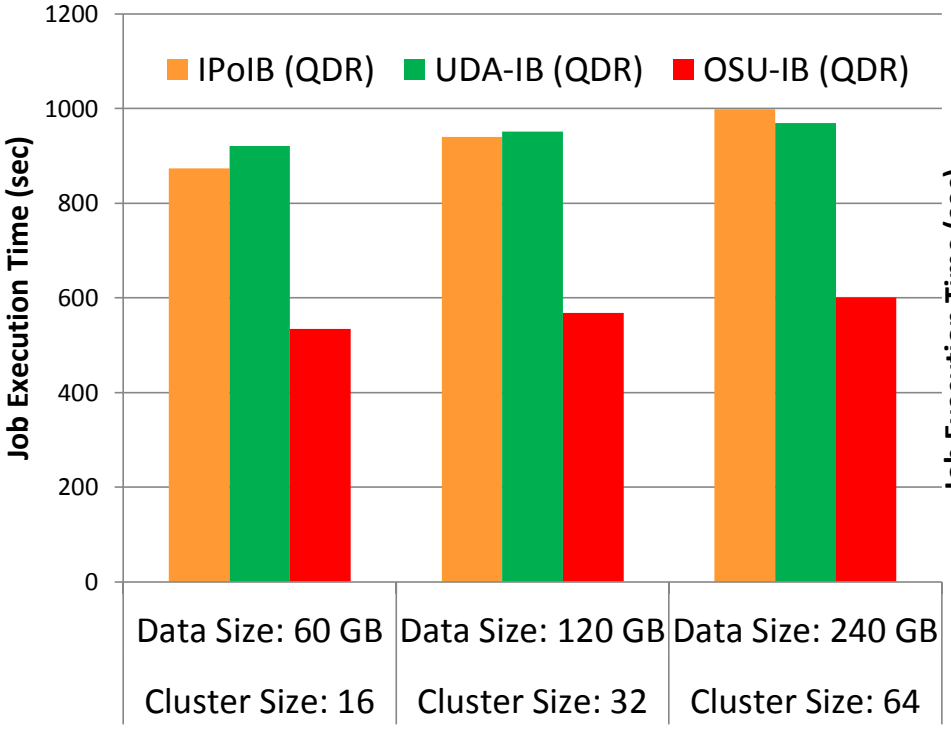
8 slave nodes



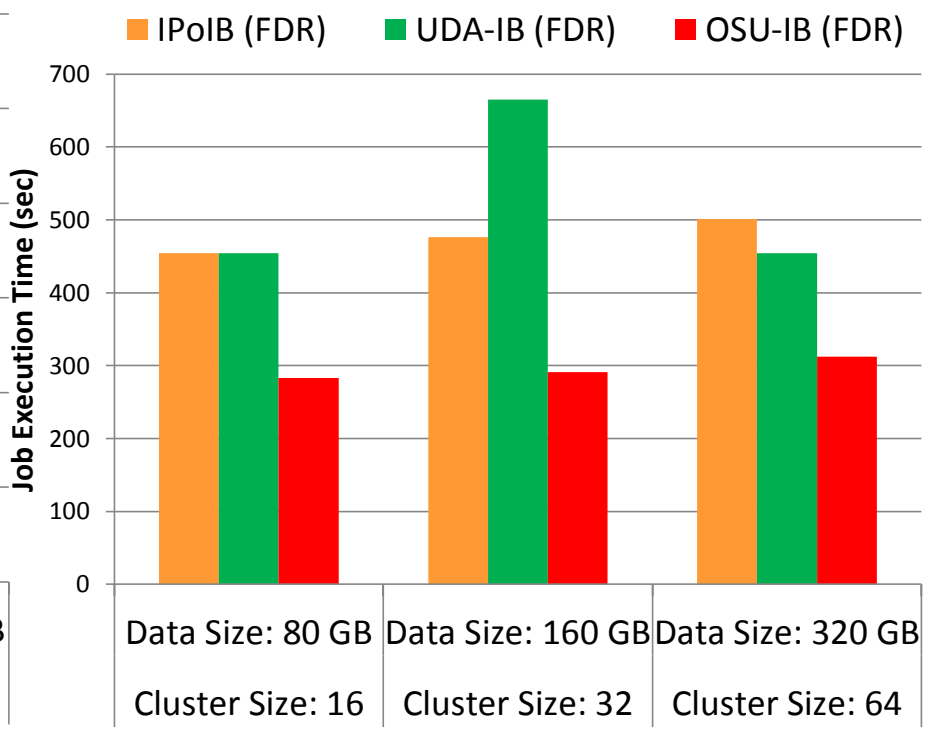
16 slave nodes

- Stand-alone MapReduce micro-benchmark (MR-AVG)
- 1 KB key/value pair size
- For 8 slave nodes, RDMA has up to 30% over IPoIB (56Gbps)
- For 16 slave nodes, RDMA has up to 28% over IPoIB (56Gbps)

Performance Evaluation of Sort and TeraSort



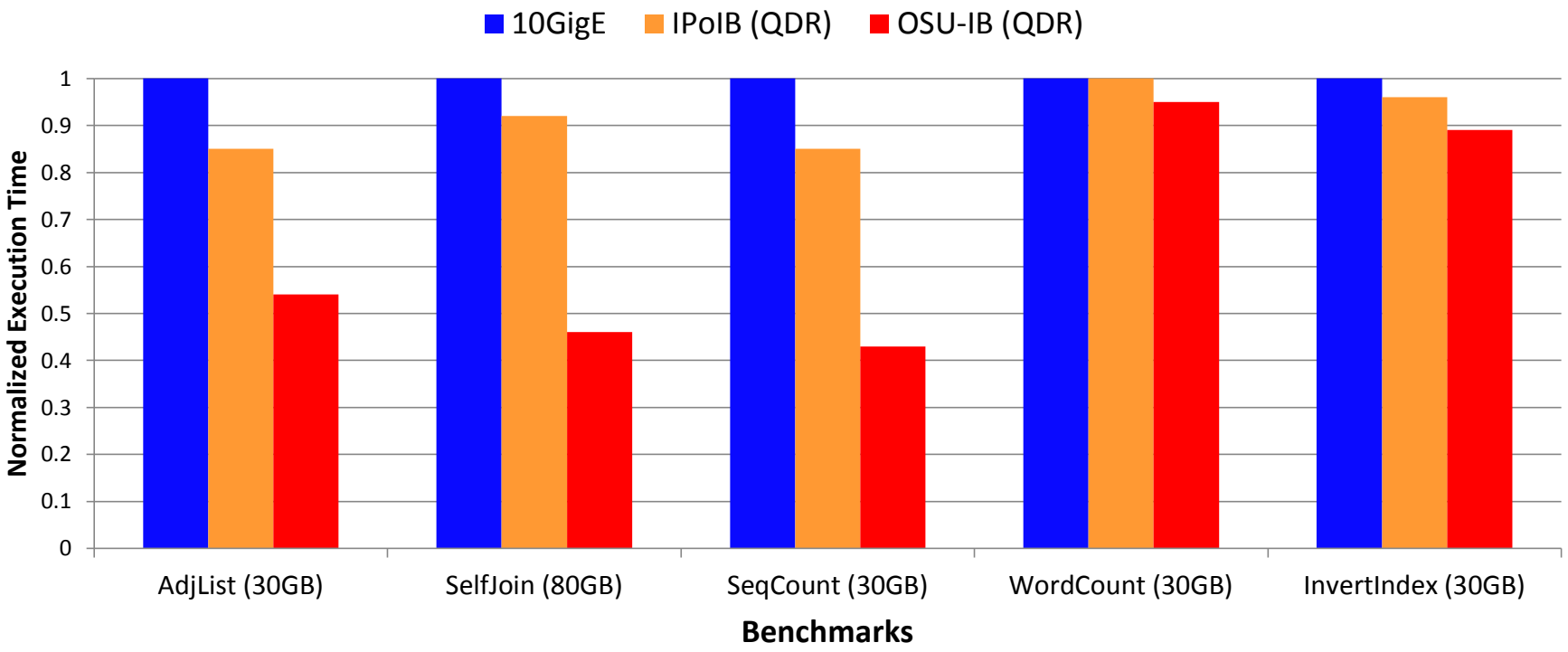
Sort in OSU Cluster



TeraSort in TACC Stampede

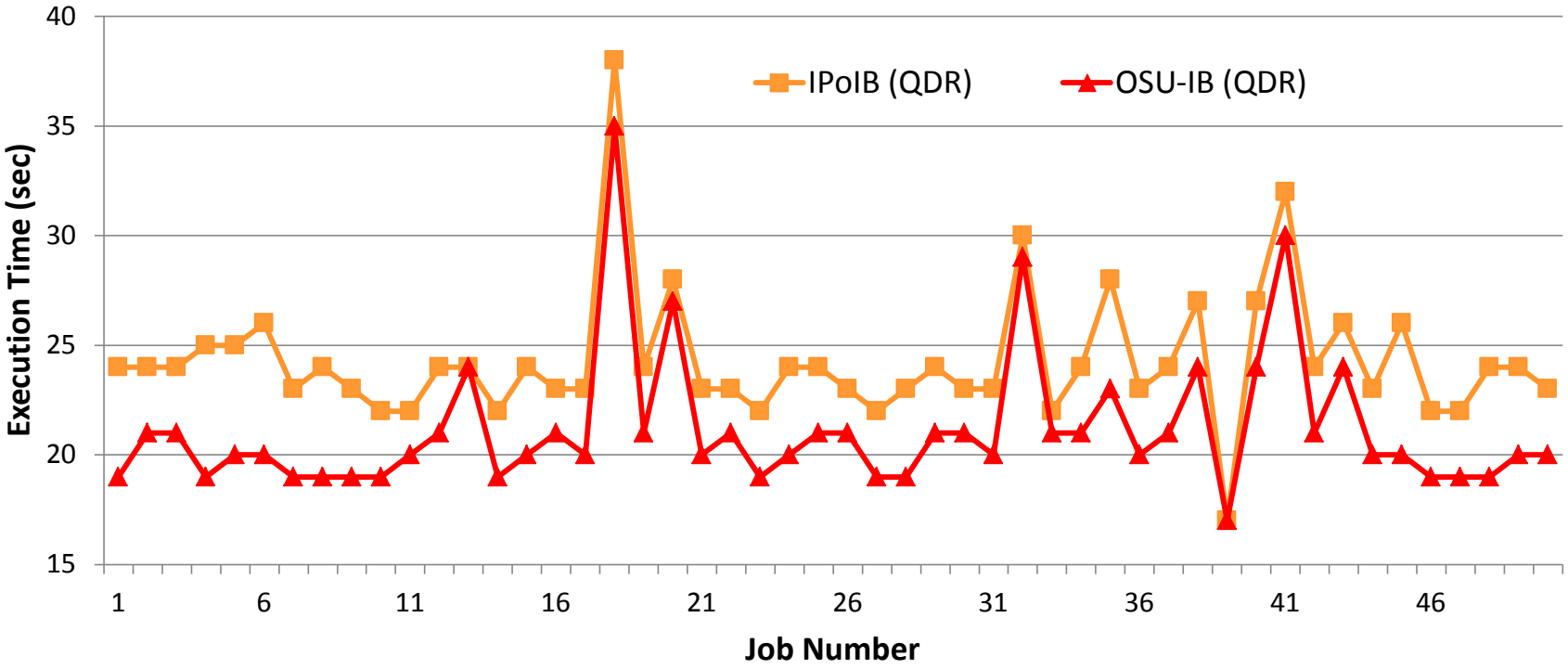
- For 240GB Sort in 64 nodes (512 cores)
  - 40% improvement over IPoIB (QDR) with HDD used for HDFS
- For 320GB TeraSort in 64 nodes (1K cores)
  - 38% improvement over IPoIB (FDR) with HDD used for HDFS

Evaluations using PUMA Workload



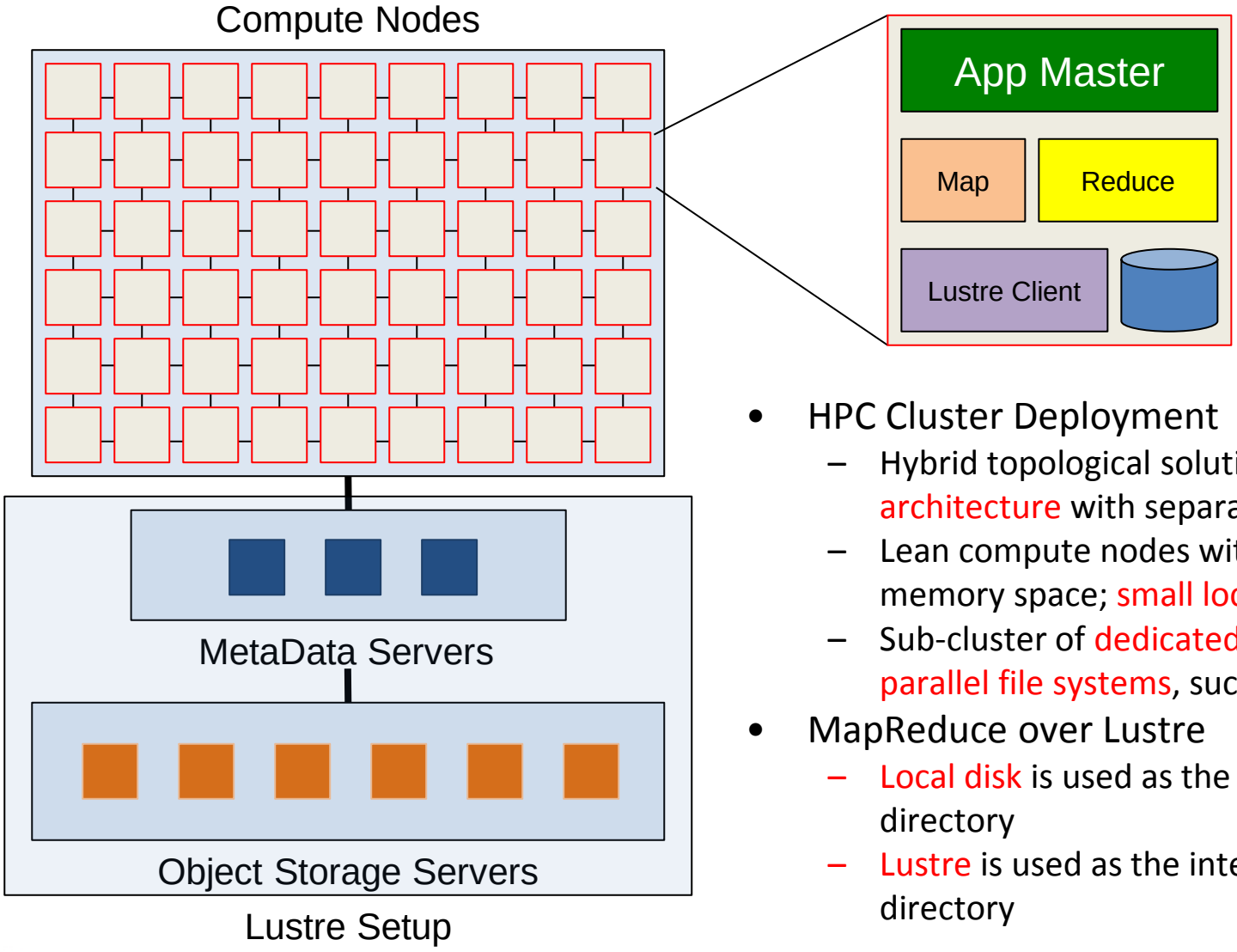
- 50% improvement in Self Join over IPoIB (QDR) for 80 GB data size
- 49% improvement in Sequence Count over IPoIB (QDR) for 30 GB data size

Evaluations using SWIM



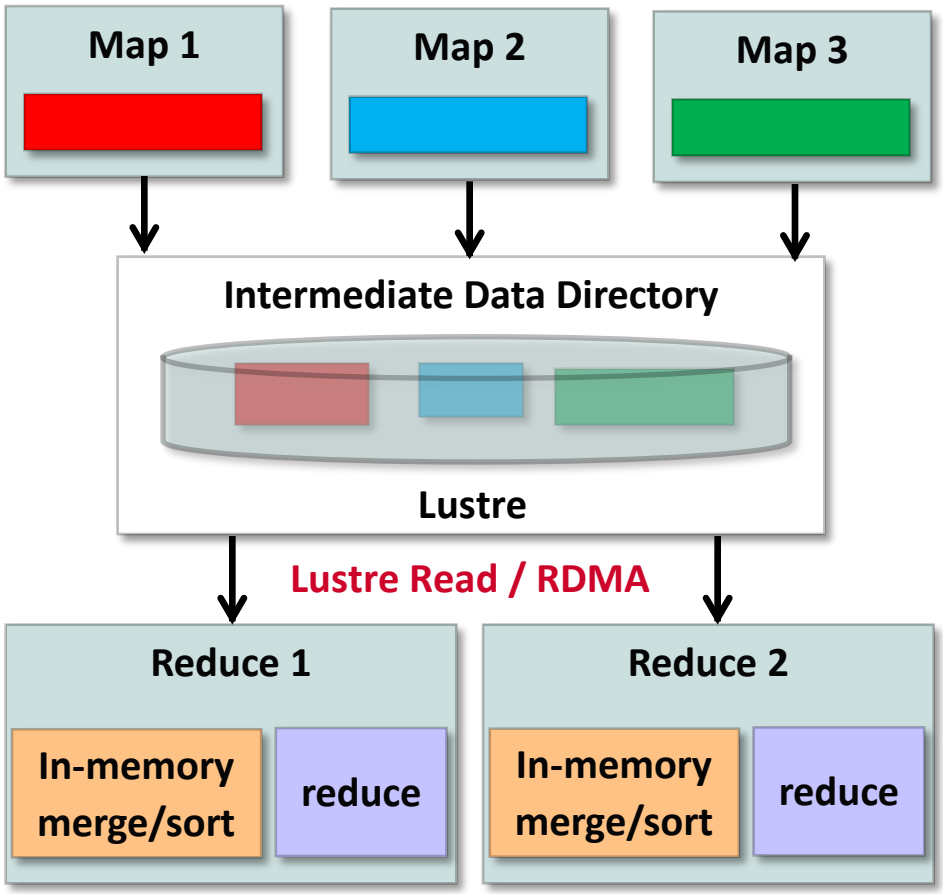
- 50 small MapReduce jobs executed in a cluster size of 4
- Maximum performance benefit 24% over IPoIB (QDR)
- Average performance benefit 13% over IPoIB (QDR)

Optimize Hadoop YARN MapReduce over Parallel File Systems



- HPC Cluster Deployment
  - Hybrid topological solution of Beowulf architecture with separate I/O nodes
  - Lean compute nodes with light OS; more memory space; small local storage
  - Sub-cluster of dedicated I/O nodes with parallel file systems, such as Lustre
- MapReduce over Lustre
  - Local disk is used as the intermediate data directory
  - Lustre is used as the intermediate data directory

Design Overview of Shuffle Strategies for MapReduce over Lustre

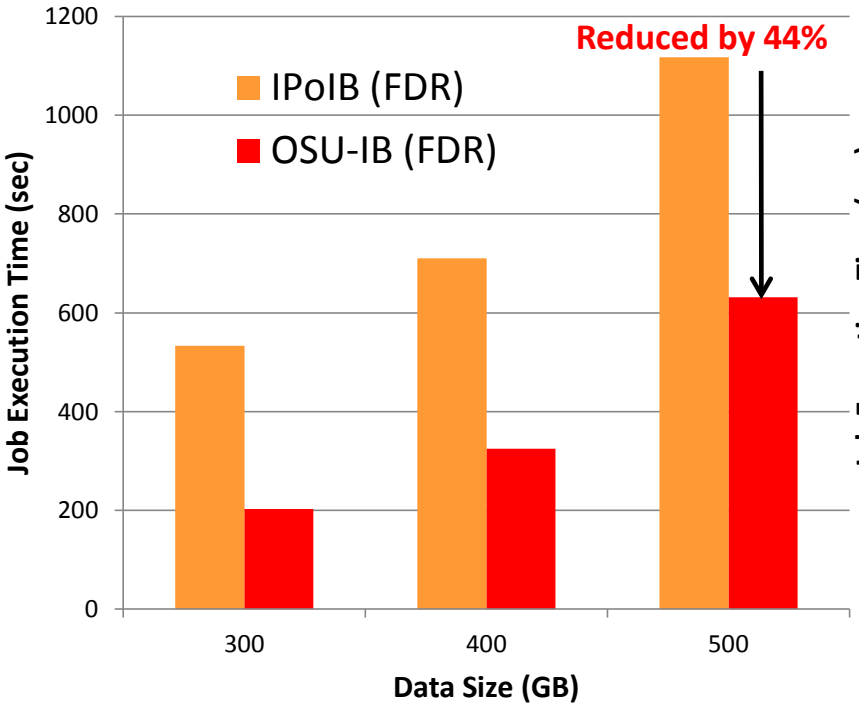


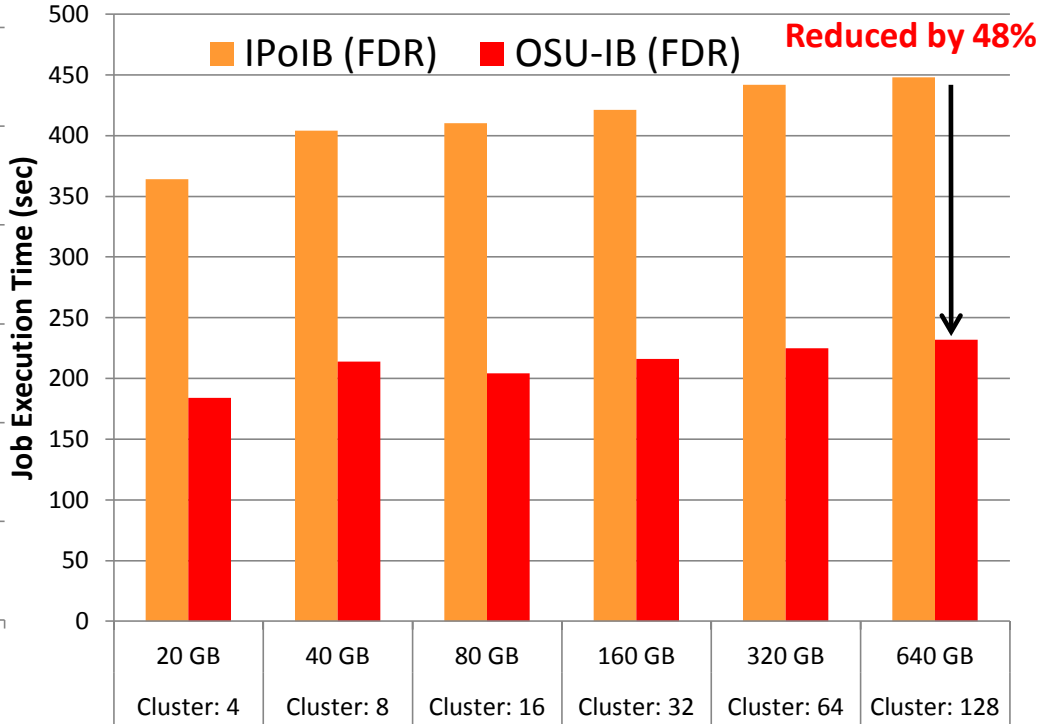
- Design Features
  - Two shuffle approaches
    - Lustre read based shuffle
    - RDMA based shuffle
  - Hybrid shuffle algorithm to take benefit from both shuffle approaches
  - Dynamically adapts to the better shuffle approach for each shuffle request based on profiling values for each Lustre read operation
  - In-memory merge and overlapping of different phases are kept similar to RDMA-enhanced MapReduce design

M. W. Rahman, X. Lu, N. S. Islam, R. Rajachandrasekar, and D. K. Panda, High Performance Design of YARN MapReduce on Modern HPC Clusters with Lustre and RDMA, IPDPS, May 2015.

Performance Improvement of MapReduce over Lustre on TACC-Stampede

- Local disk is used as the intermediate data directory



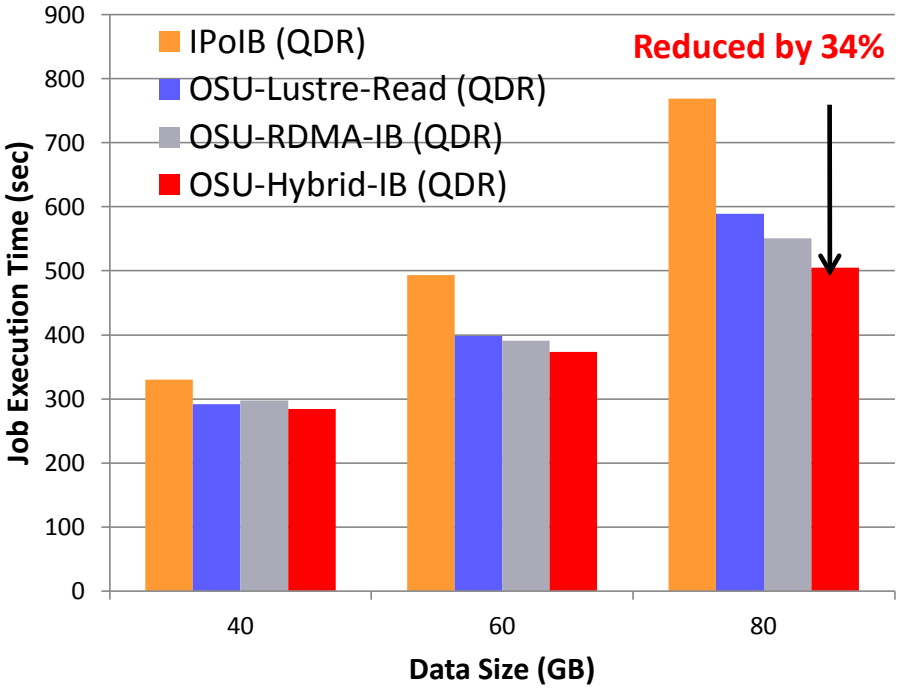


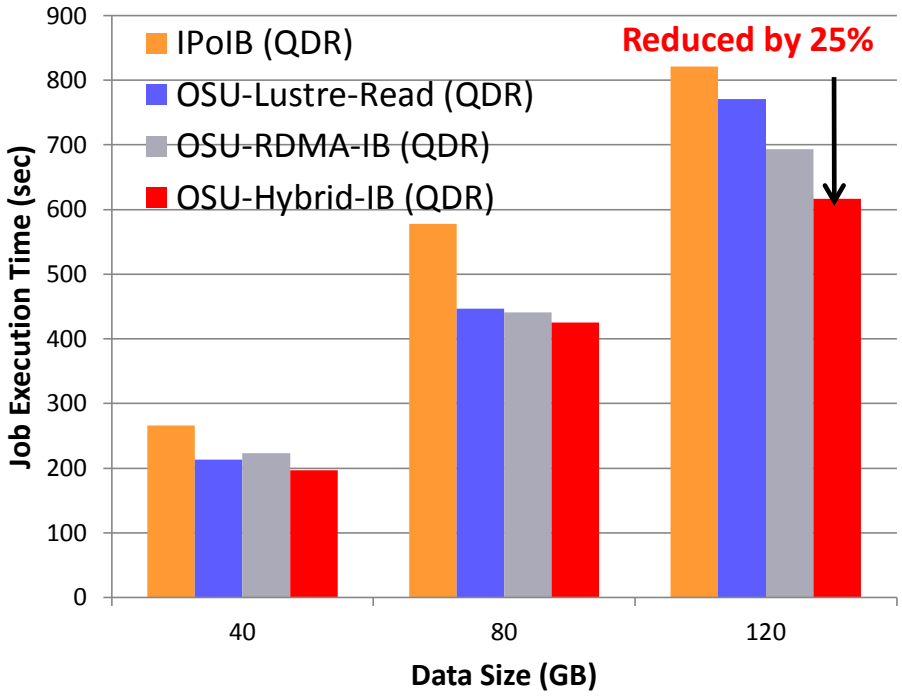
- For 500GB Sort in 64 nodes
  - 44% improvement over IPoIB (FDR)
- For 640GB Sort in 128 nodes
  - 48% improvement over IPoIB (FDR)

M. W. Rahman, X. Lu, N. S. Islam, R. Rajachandrasekar, and D. K. Panda, MapReduce over Lustre: Can RDMA-based Approach Benefit?, Euro-Par, August 2014.

Case Study - Performance Improvement of MapReduce over Lustre on SDSC-Gordon

- Lustre is used as the intermediate data directory



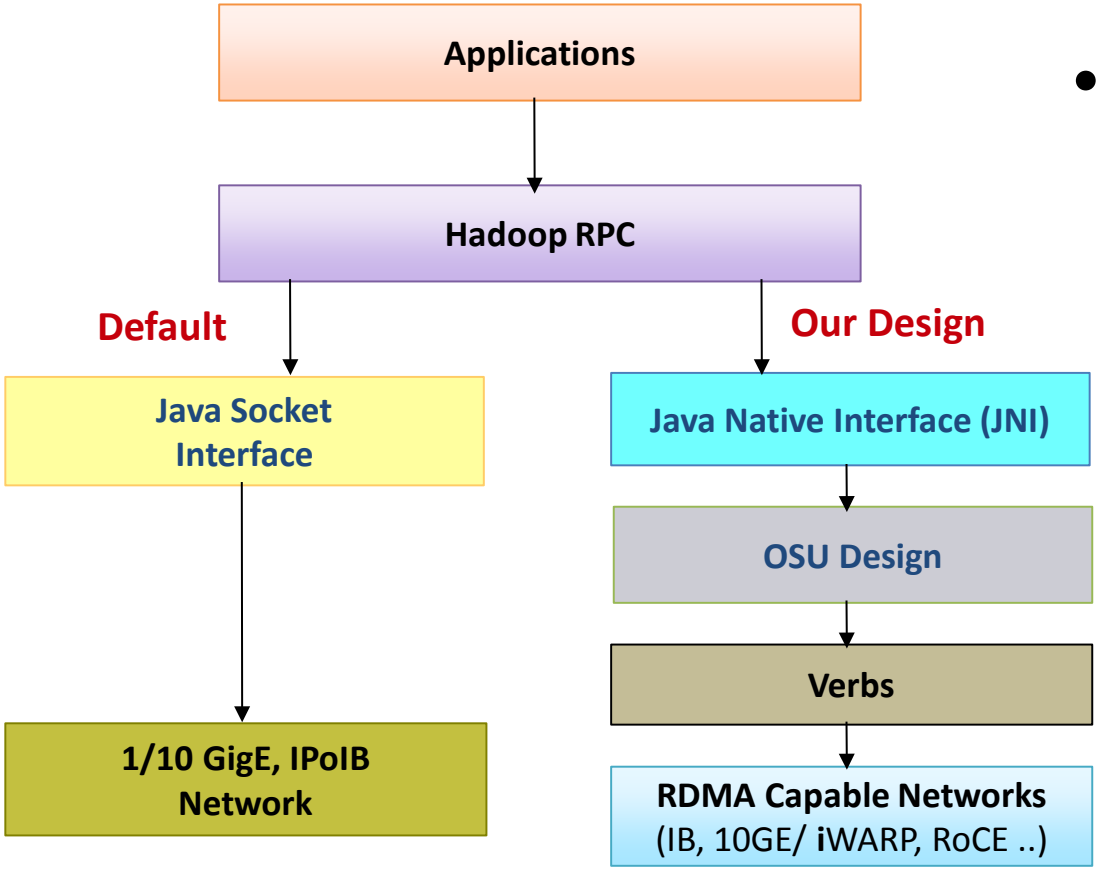


- For 80GB Sort in 8 nodes
  - 34% improvement over IPoIB (QDR)
- For 120GB TeraSort in 16 nodes
  - 25% improvement over IPoIB (QDR)

Acceleration Case Studies and In-Depth Performance Evaluation

- RDMA-based Designs and Performance Evaluation
  - HDFS
  - MapReduce
  - RPC
  - Spark
  - HBase
  - Memcached

Design Overview of Hadoop RPC with RDMA

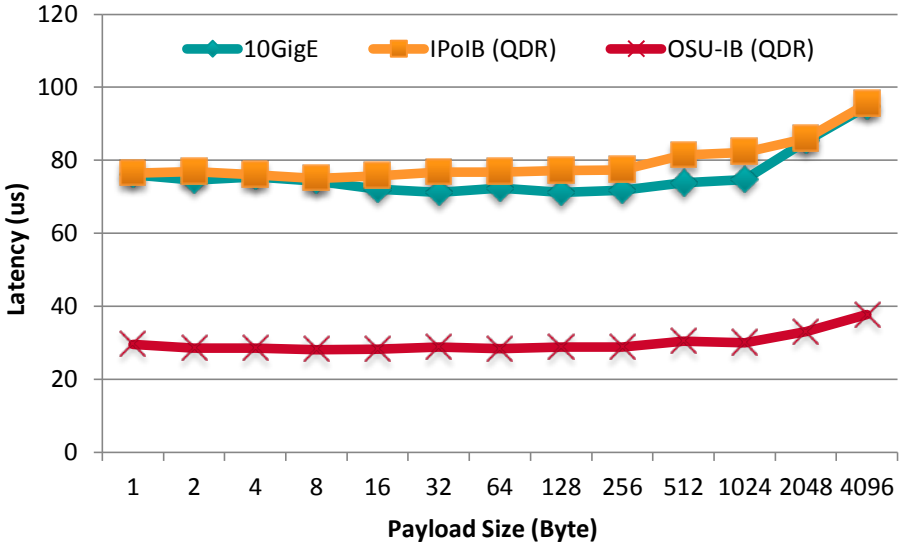


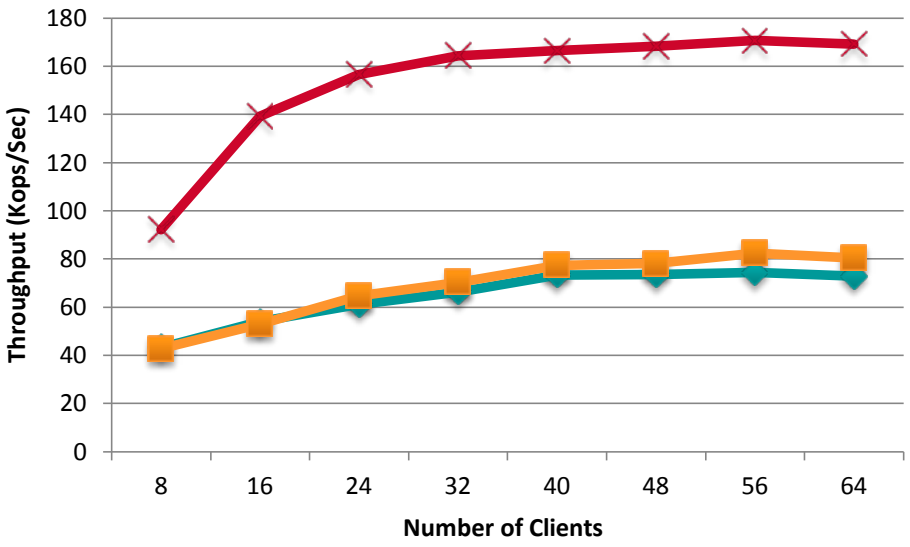
- Design Features
  - JVM-bypassed buffer management
  - RDMA or send/recv based adaptive communication
  - Intelligent buffer allocation and adjustment for serialization
  - On-demand connection setup
  - InfiniBand/RoCE support

- Enables high performance RDMA communication, while supporting traditional socket interface
- JNI Layer bridges Java based RPC with communication library written in native code

X. Lu, N. Islam, M. W. Rahman, J. Jose, H. Subramoni, H. Wang, and D. K. Panda, High-Performance Design of Hadoop RPC with RDMA over InfiniBand, Int'l Conference on Parallel Processing (ICPP '13), October 2013.

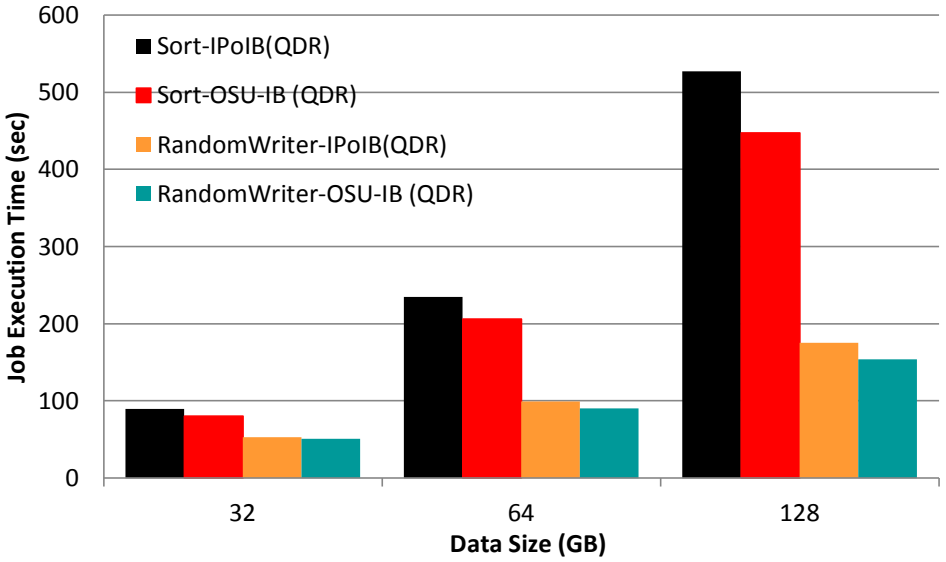
Hadoop RPC with RDMA - Gain in Latency and Throughput



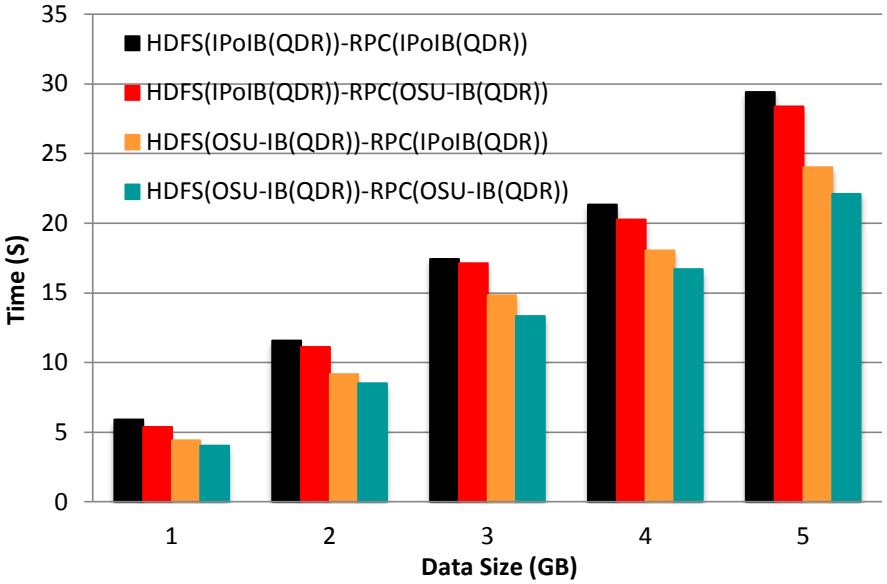


- Hadoop RPC with RDMA PingPong Latency
  - 1 byte: 30 us; 4 KB: 38 us
  - 59%-62% and 60%-63% improvements compared with the performance on 10 GigE and IPoIB (32Gbps), respectively
- Hadoop RPC with RDMA Throughput
  - 512 bytes & 56 clients: 170.63 Kops/sec
  - 129% and 107% improvements compared with the peak performance on 10 GigE and IPoIB (32Gbps), respectively

Hadoop RPC over IB - Gain in MapReduce and HDFS



MapReduce RandomWriter & Sort



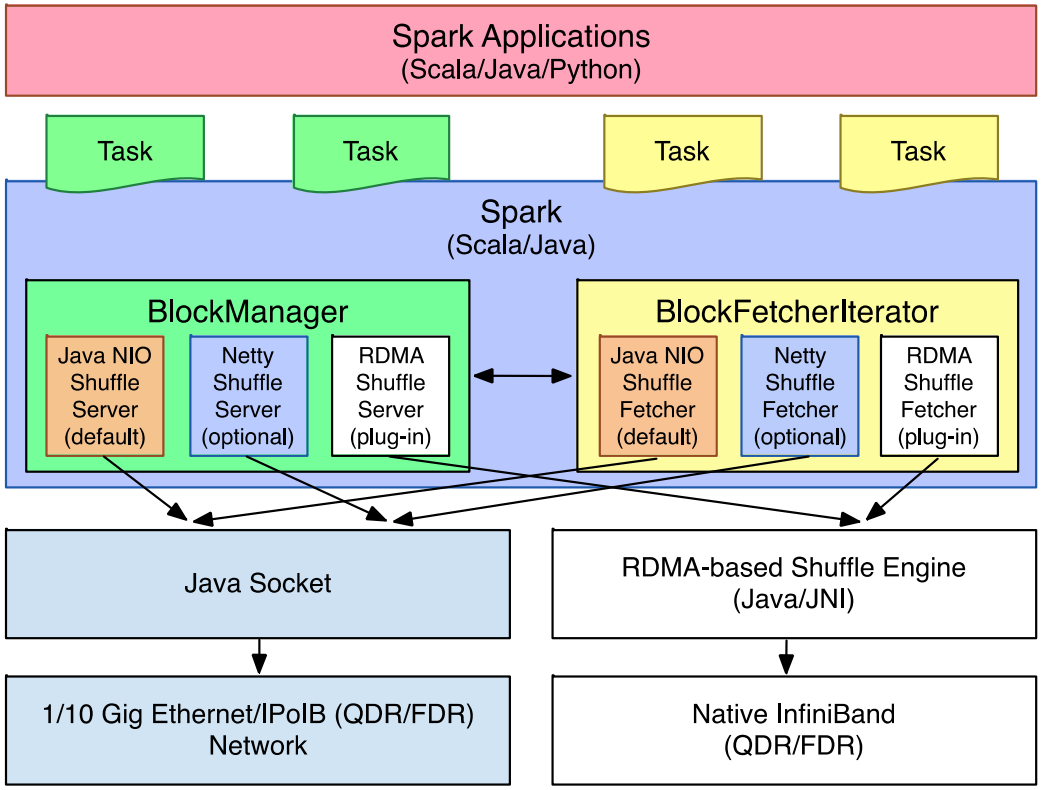
HDFS Write Micro-Benchmark

- The performance gains in MapReduce benchmarks
  - RandomWriter: up to 12% improvement over IPoIB
  - Sort: up to 15.2% improvement over IPoIB
- The latency of HDFS Write is reduced by 10% compared to Hadoop RPC running on IPoIB and OSU-IB HDFS only

Acceleration Case Studies and In-Depth Performance Evaluation

- RDMA-based Designs and Performance Evaluation
  - HDFS
  - MapReduce
  - RPC
  - Spark
  - HBase
  - Memcached

Design Overview of Spark with RDMA

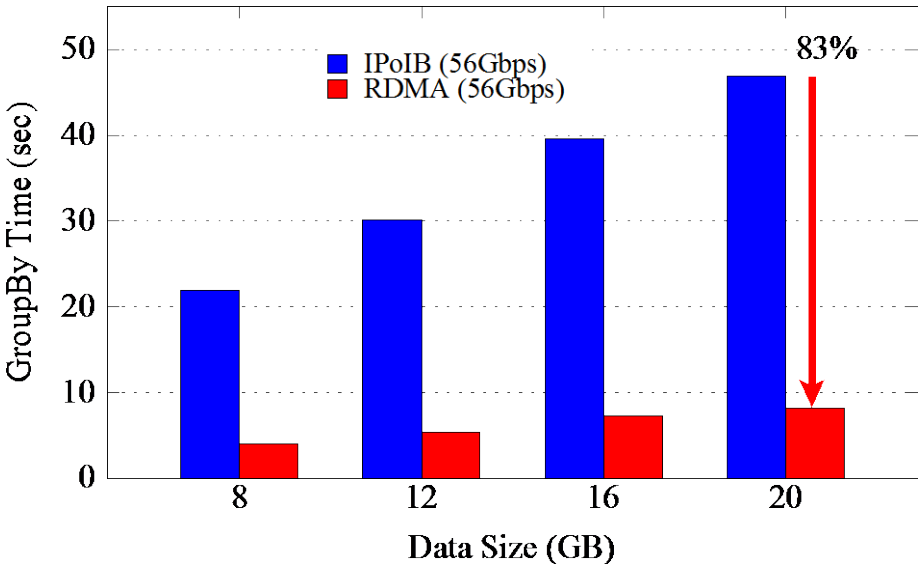


- Design Features
  - RDMA based shuffle
  - SEDA-based plugins
  - Dynamic connection management and sharing
  - Non-blocking and out-of-order data transfer
  - Off-JVM-heap buffer management
  - InfiniBand/RoCE support

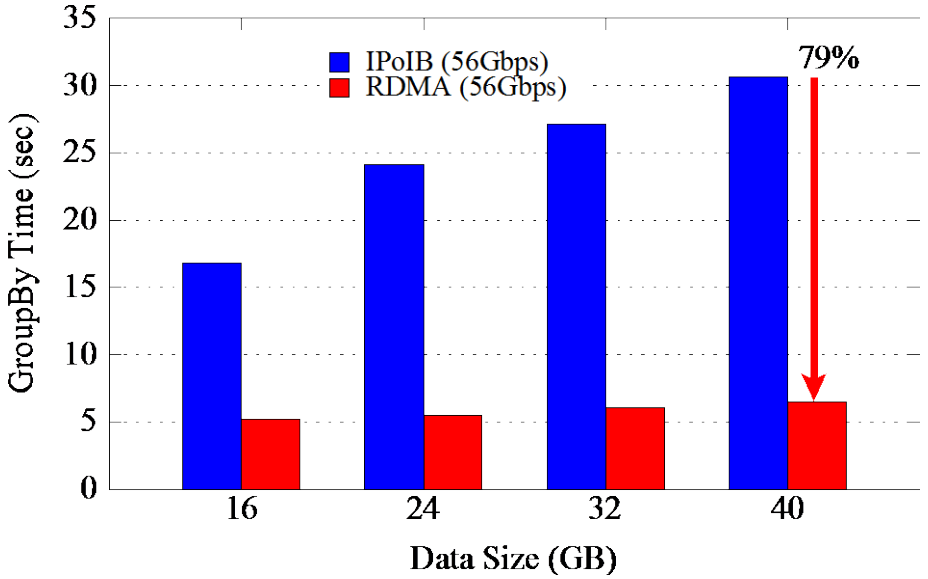
- Enables high performance RDMA communication, while supporting traditional socket interface
- JNI Layer bridges Scala based Spark with communication library written in native code

X. Lu, M. W. Rahman, N. Islam, D. Shankar, and D. K. Panda, Accelerating Spark with RDMA for Big Data Processing: Early Experiences, Int'l Symposium on High Performance Interconnects (HotI'14), August 2014

Performance Evaluation on TACC Stampede - GroupByTest



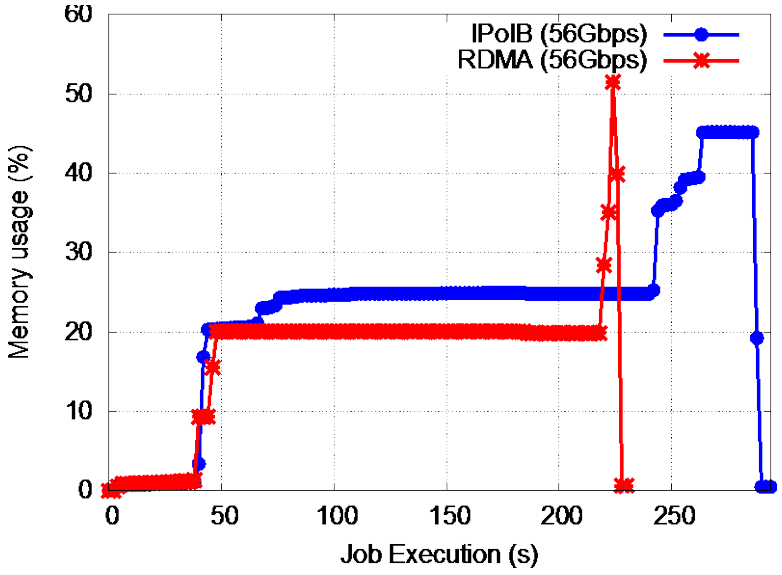
8 Worker Nodes, 128 Cores, (128M 128R)



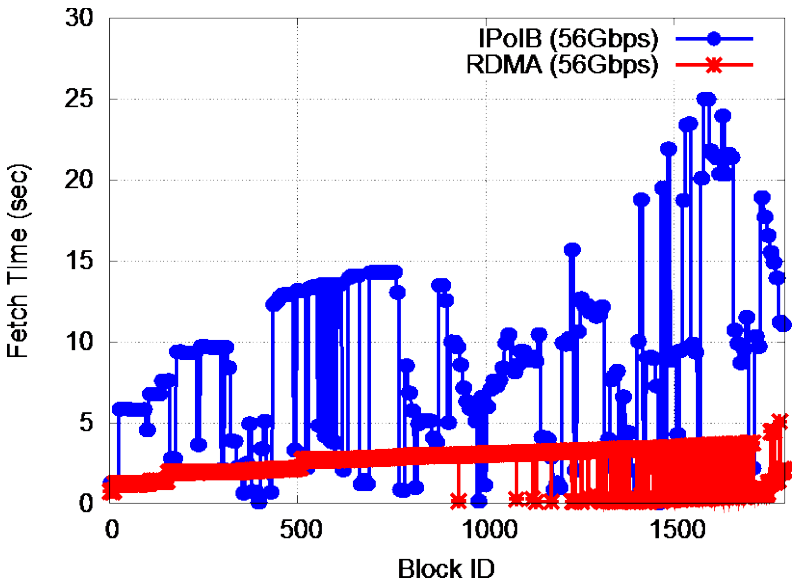
16 Worker Nodes, 256 Cores, (256M 256R)

- Intel SandyBridge + FDR
- Cluster with 8 HDD Nodes, single disk per node, 128 concurrent tasks
  - up to 83% over IPoIB (56Gbps)
- Cluster with 16 HDD Nodes, single disk per node, 256 concurrent tasks
  - up to 79% over IPoIB (56Gbps)

Performance Analysis on TACC Stampede



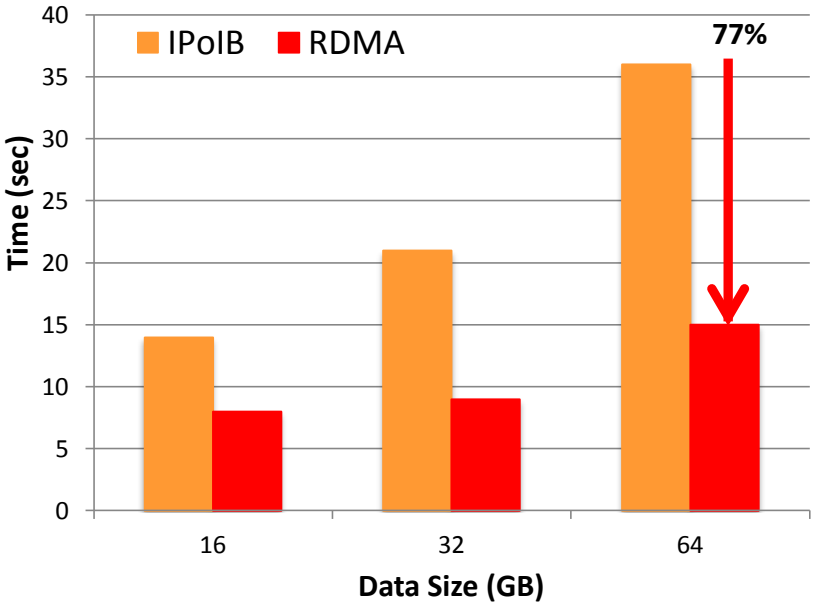
Memory Footprint



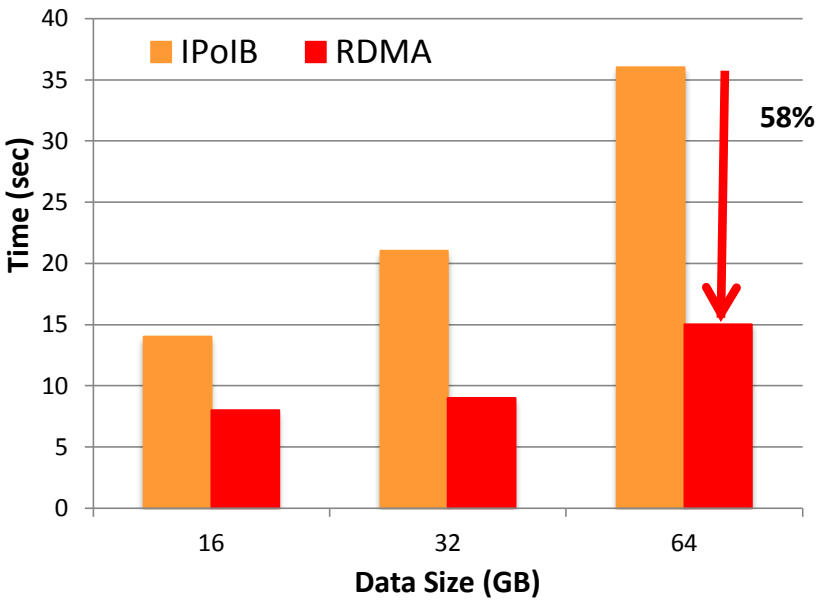
Block Fetch Time

- Overall lower memory footprint
- Much faster block fetch time with RDMA
- Total Job Execution Time
  - IPoIB: 294 secs; RDMA: 230 secs
  - up to 22% over IPoIB (56Gbps)

Performance Evaluation on TACC Stampede - SortByTest



16 Worker Nodes, SortByTest Shuffle Time



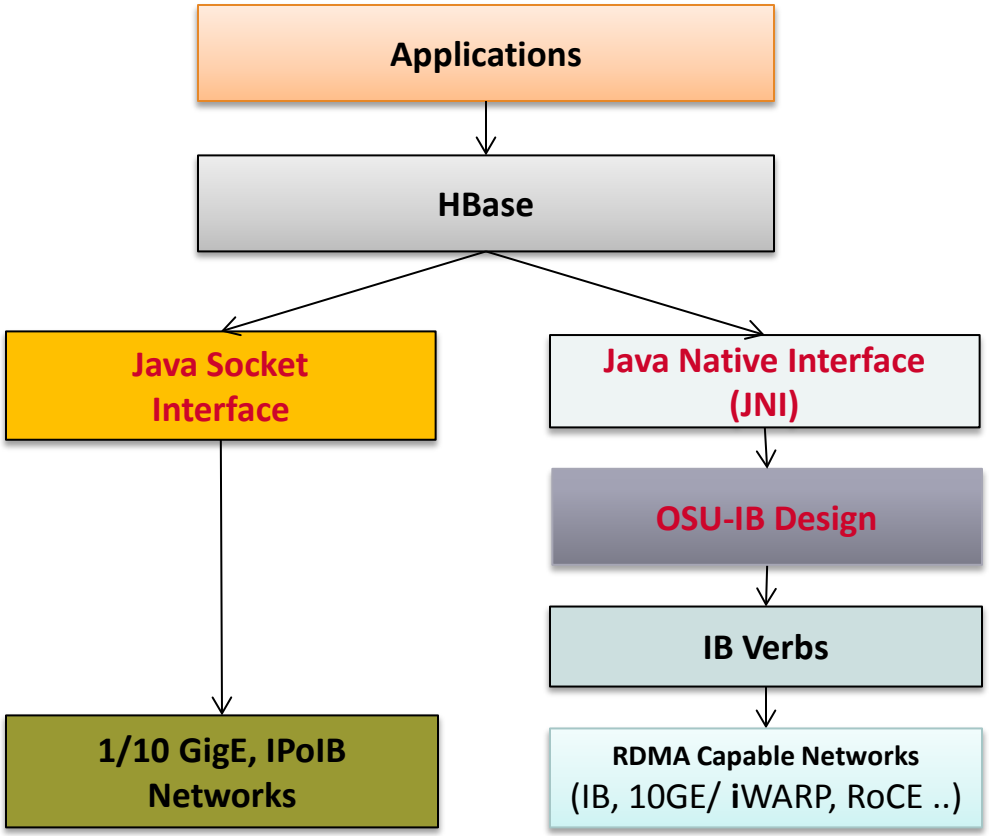
16 Worker Nodes, SortByTest Total Time

- Intel SandyBridge + FDR, 16 Worker Nodes, 256 Cores, (256M 256R)
- RDMA-based design for Spark 1.4.0
- RDMA vs. IPoIB with 256 concurrent tasks, single disk per node and RamDisk. For SortByKey Test:
  - Shuffle time reduced by up to 77% over IPoIB (56Gbps)
  - Total time reduced by up to 58% over IPoIB (56Gbps)

Acceleration Case Studies and In-Depth Performance Evaluation

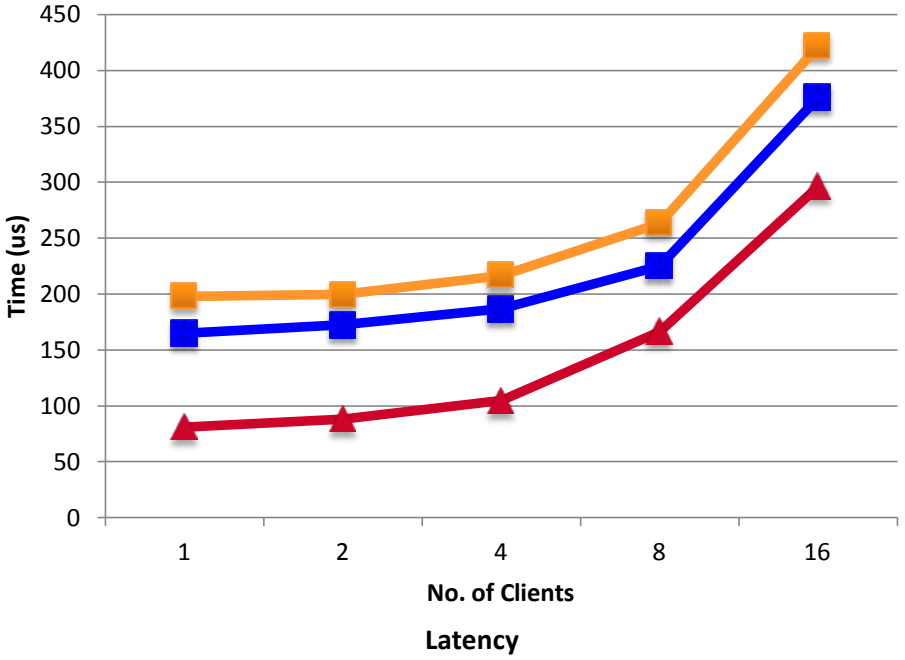
- RDMA-based Designs and Performance Evaluation
  - HDFS
  - MapReduce
  - RPC
  - Spark
  - HBase
  - Memcached

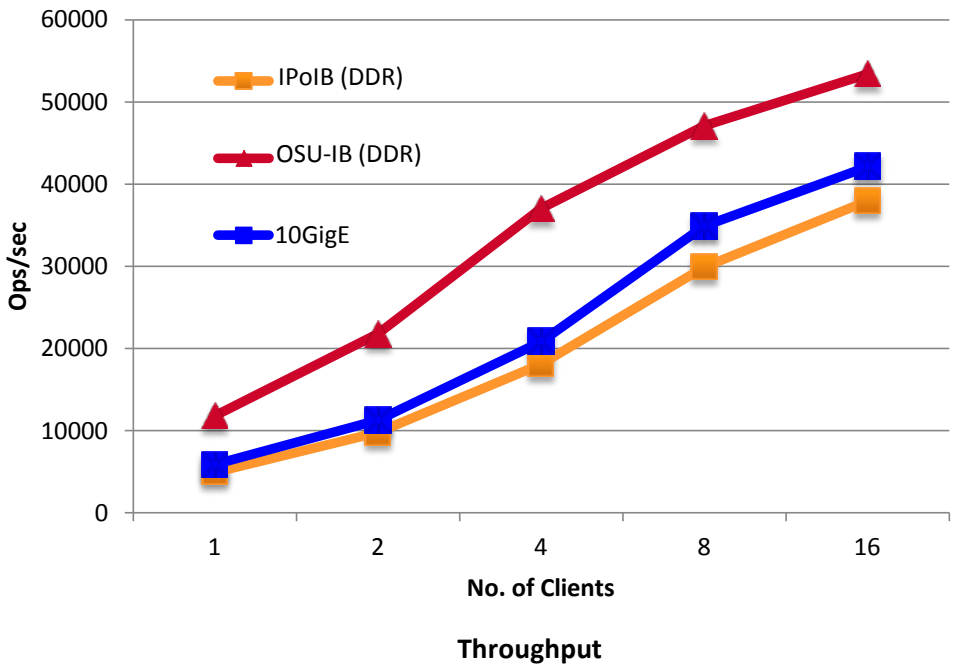
HBase-RDMA Design Overview



- JNI Layer bridges Java based HBase with communication library written in native code
- Enables high performance RDMA communication, while supporting traditional socket interface

HBase Micro-benchmark (Single-Server-Multi-Client) Results

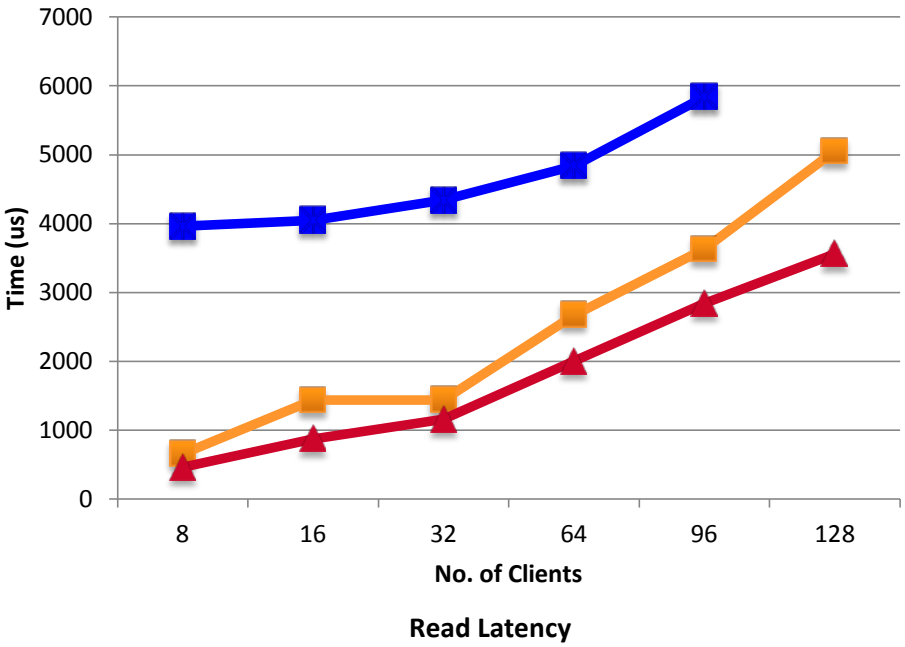




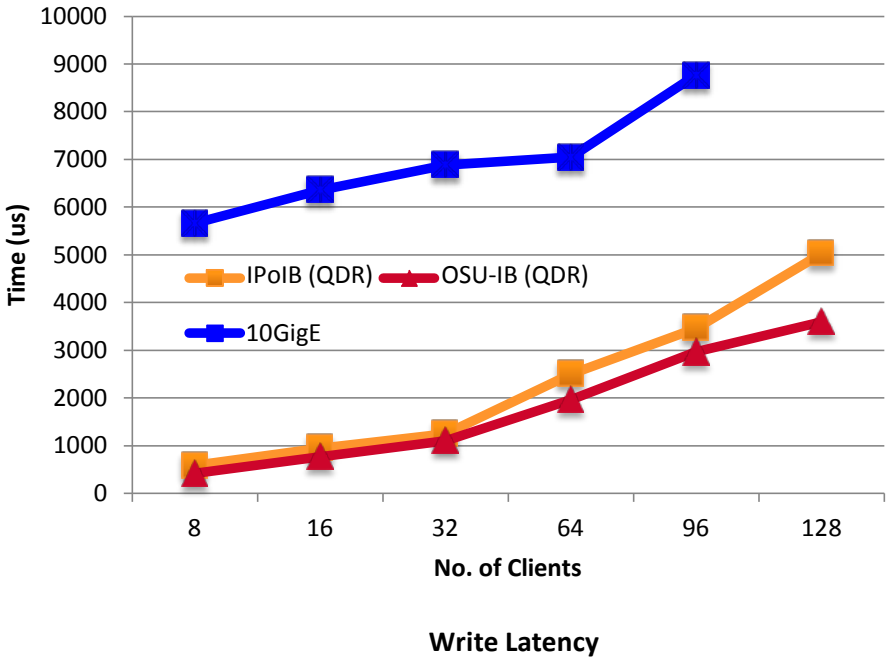
- HBase Get latency
  - 4 clients: 104.5 us; 16 clients: 296.1 us
- HBase Get throughput
  - 4 clients: 37.01 Kops/sec; 16 clients: 53.4 Kops/sec
- 27% improvement in throughput for 16 clients over 10GE

J. Huang, X. Ouyang, J. Jose, M. W. Rahman, H. Wang, M. Luo, H. Subramoni, Chet Murthy, and D. K. Panda, High-Performance Design of HBase with RDMA over InfiniBand, IPDPS'12

HBase – YCSB Read-Write Workload



Read Latency



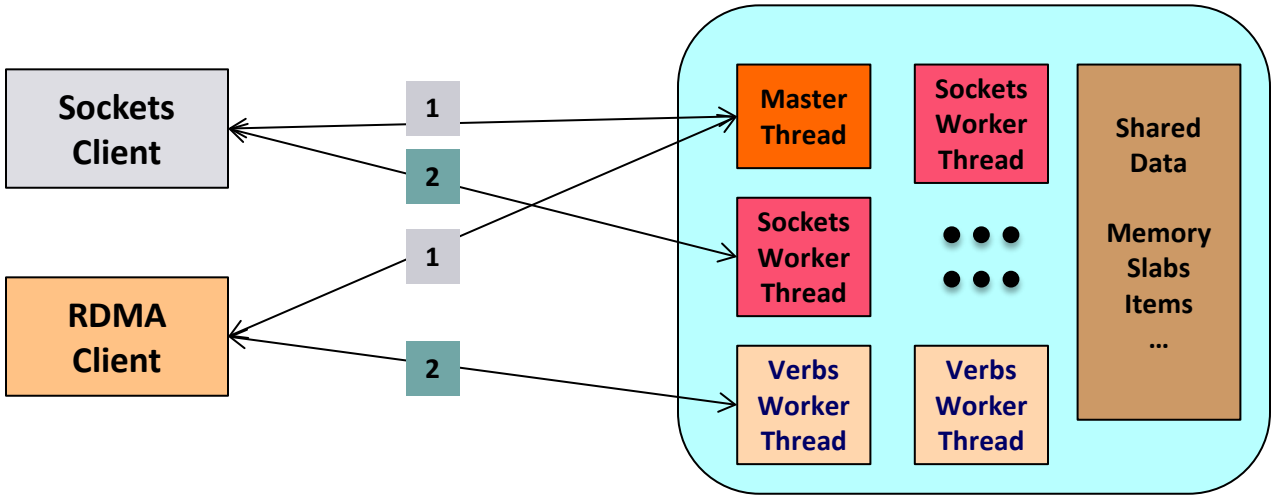
Write Latency

- HBase Get latency (Yahoo! Cloud Service Benchmark)
  - 64 clients: 2.0 ms; 128 Clients: 3.5 ms
  - 42% improvement over IPoIB for 128 clients
- HBase Put latency
  - 64 clients: 1.9 ms; 128 Clients: 3.5 ms
  - 40% improvement over IPoIB for 128 clients

Acceleration Case Studies and In-Depth Performance Evaluation

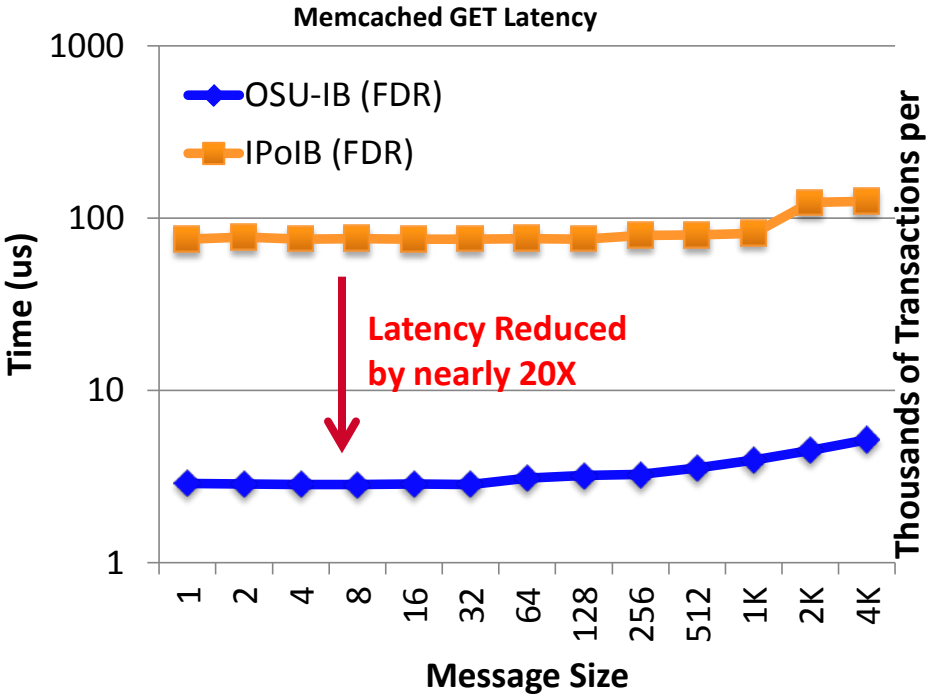
- RDMA-based Designs and Performance Evaluation
  - HDFS
  - MapReduce
  - RPC
  - Spark
  - HBase
  - Memcached

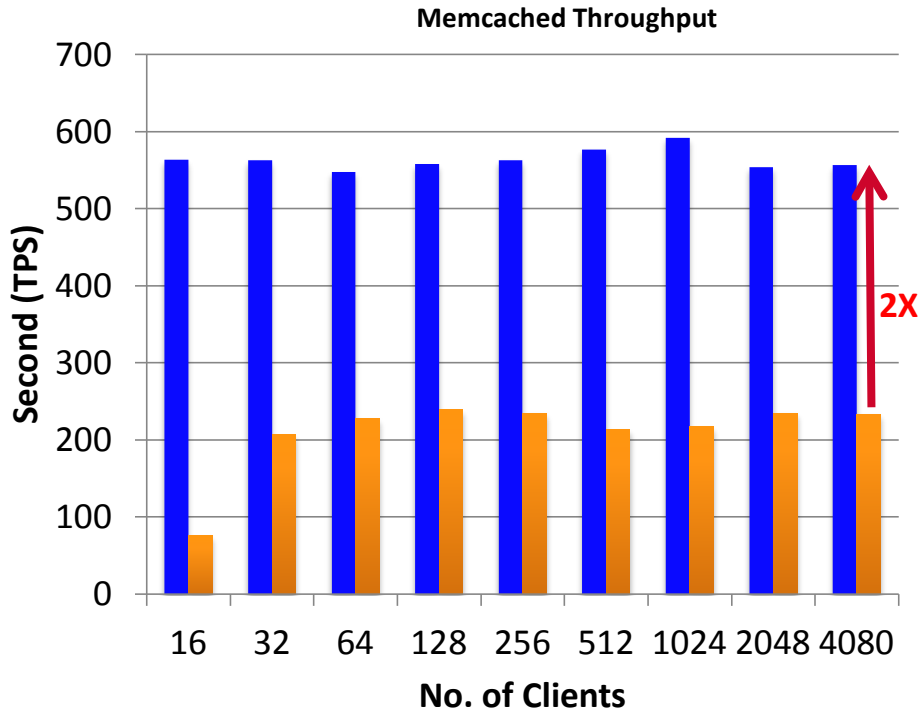
Memcached-RDMA Design



- Server and client perform a negotiation protocol
  - Master thread assigns clients to appropriate worker thread
- Once a client is assigned a verbs worker thread, it can communicate directly and is "bound" to that thread
- All other Memcached data structures are shared among RDMA and Sockets worker threads
- Memcached Server can serve both socket and verbs clients simultaneously
- Memcached applications need not be modified; uses verbs interface if available

Memcached Performance (FDR Interconnect)

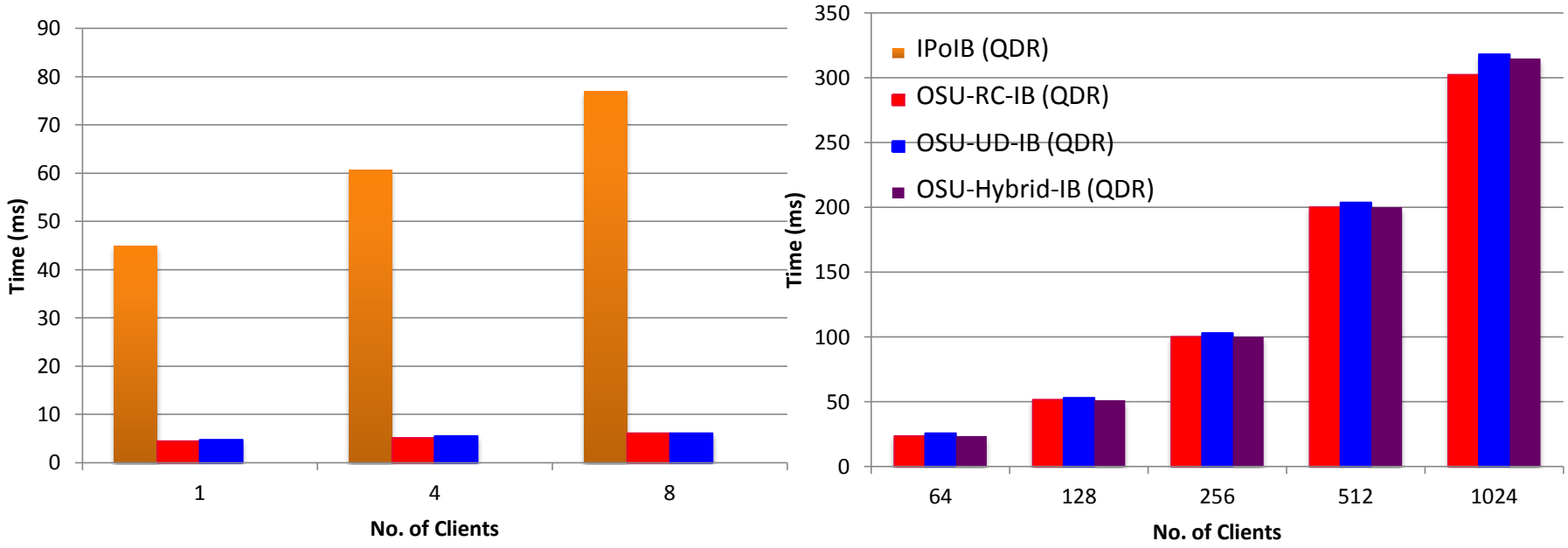




Experiments on TACC Stampede (Intel SandyBridge Cluster, IB: FDR)

- Memcached Get latency
  - 4 bytes OSU-IB: 2.84 us; IPoIB: 75.53 us
  - 2K bytes OSU-IB: 4.49 us; IPoIB: 123.42 us
- Memcached Throughput (4bytes)
  - 4080 clients OSU-IB: 556 Kops/sec, IPoIB: 233 Kops/s
  - Nearly 2X improvement in throughput

Application Level Evaluation – Real Application Workloads

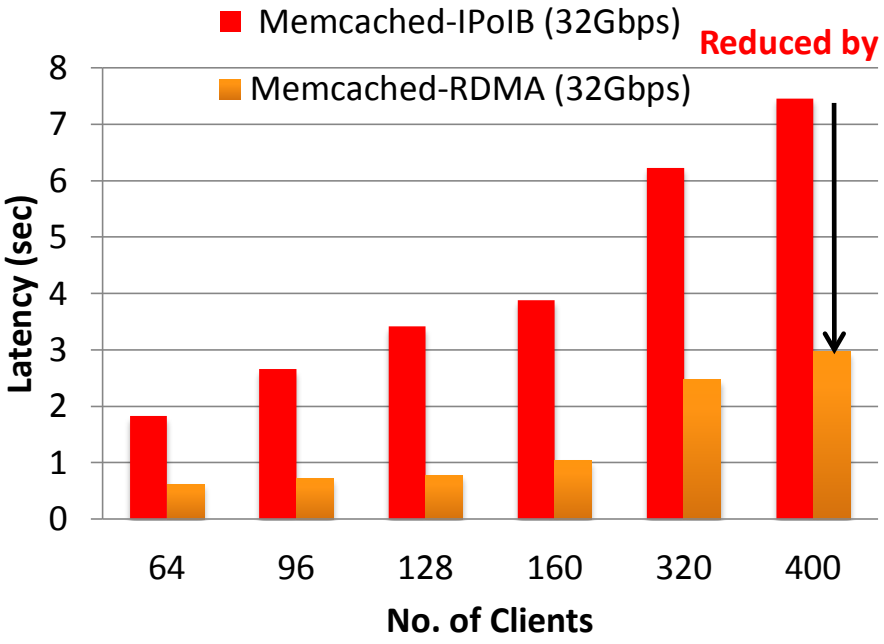


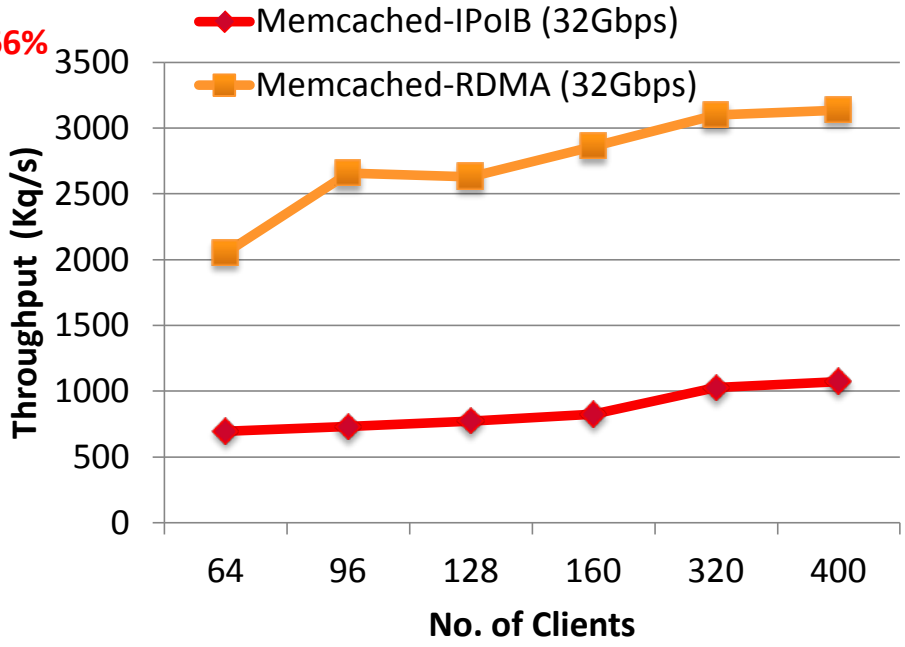
- Real Application Workload
  - RC – 302 ms, UD – 318 ms, Hybrid – 314 ms for 1024 clients
- 12X times better than IPoIB for 8 clients
- Hybrid design achieves comparable performance to that of pure RC design

J. Jose, H. Subramoni, M. Luo, M. Zhang, J. Huang, M. W. Rahman, N. Islam, X. Ouyang, H. Wang, S. Sur and D. K. Panda, Memcached Design on High Performance RDMA Capable Interconnects, ICPP'11

J. Jose, H. Subramoni, K. Kandalla, M. W. Rahman, H. Wang, S. Narravula, and D. K. Panda, Scalable Memcached design for InfiniBand Clusters using Hybrid Transport, CCGrid'12

Micro-benchmark Evaluation for OLDP workloads

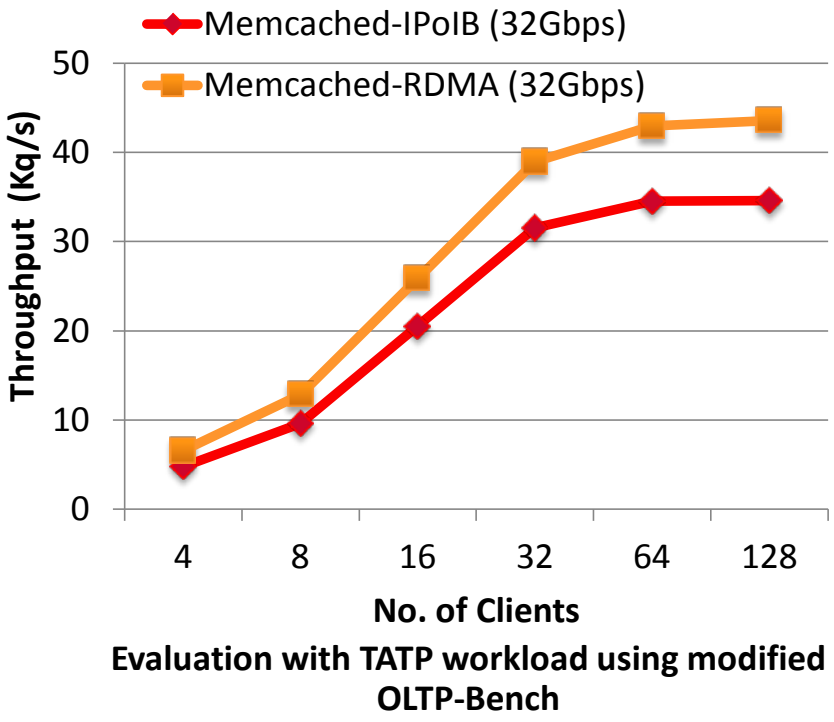




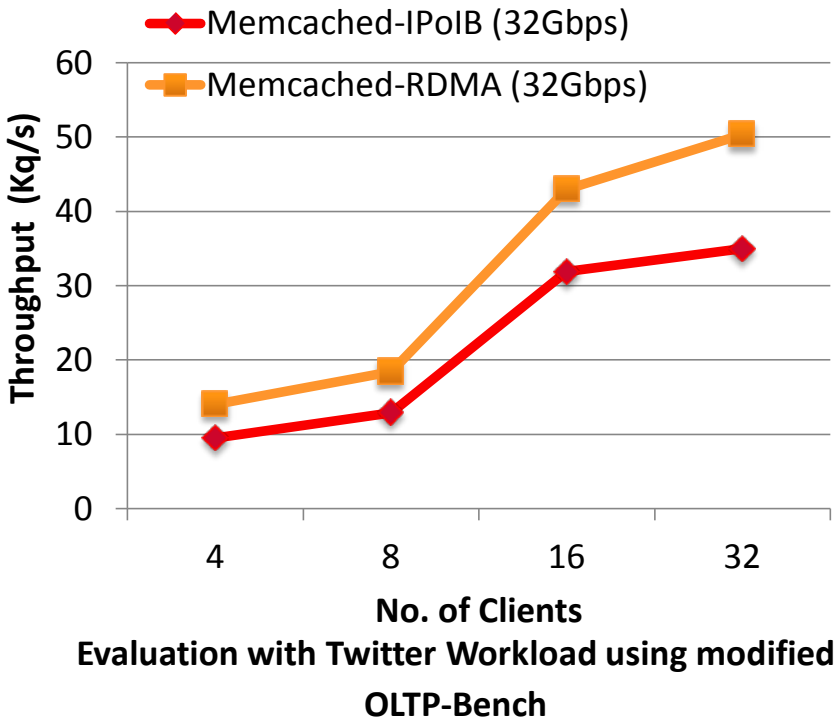
- Illustration with Read-Cache-Read access pattern using modified mysqlslap load testing tool
- Memcached-RDMA can
  - improve query latency by up to 66% over IPoIB (32Gbps)
  - throughput by up to 69% over IPoIB (32Gbps)

D. Shankar, X. Lu, J. Jose, M. W. Rahman, N. Islam, and D. K. Panda, Can RDMA Benefit On-Line Data Processing Workloads with Memcached and MySQL, ISPASS'15

Evaluation with Transactional and Web-oriented Workloads



Evaluation with TATP workload using modified OLTP-Bench



Evaluation with Twitter Workload using modified OLTP-Bench

Transactional workloads. Example: TATP

- Up to 29% improvement in overall throughput as compared to default Memcached running over IPoIB

Web-Oriented workloads. Example: Twitter workload

- Up to 42% improvement in overall throughput compared to default Memcached running over IPoIB

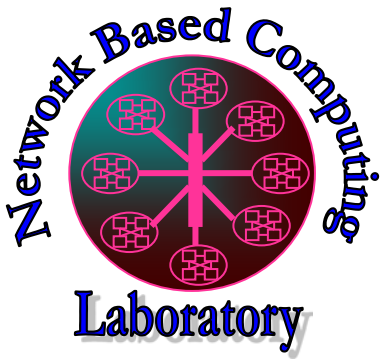
Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

The High-Performance Big Data (HiBD) Project

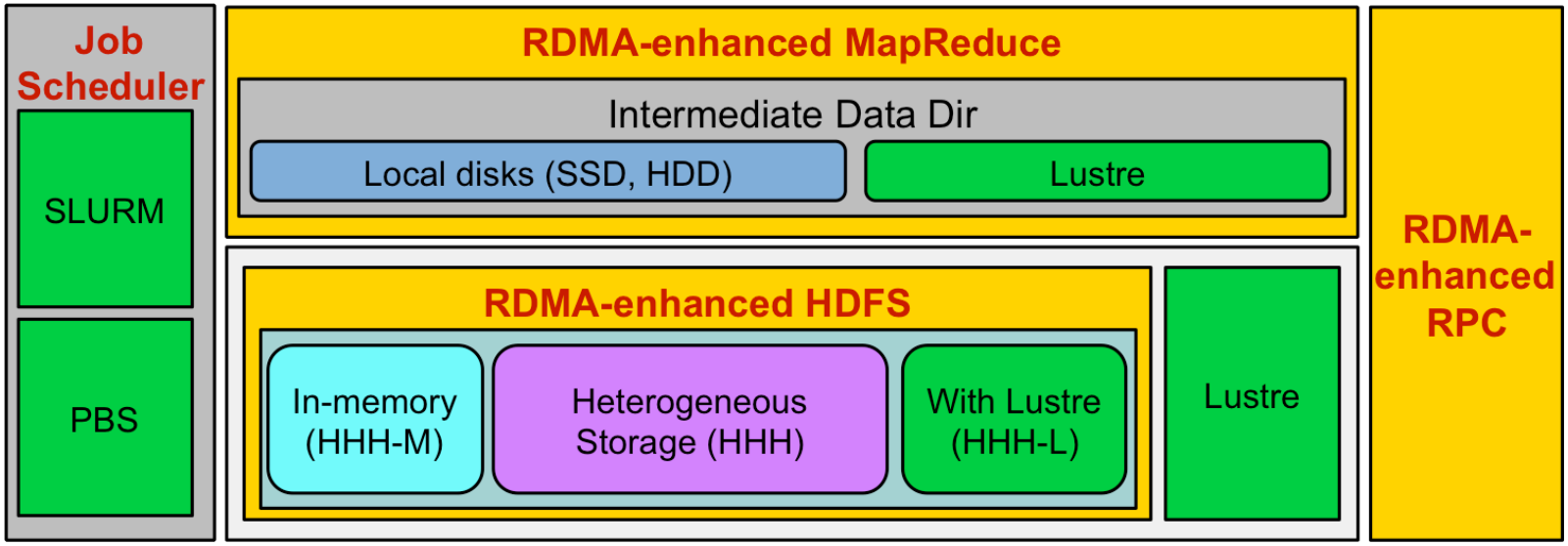
- RDMA for Apache Hadoop 2.x (RDMA-Hadoop-2.x)
  - Plugins for Apache and HDP Hadoop distributions
- RDMA for Apache Hadoop 1.x (RDMA-Hadoop)
- RDMA for Memcached (RDMA-Memcached)
- OSU HiBD-Benchmarks (OHB)
  - HDFS and Memcached Micro-benchmarks
- **http://hibd.cse.ohio-state.edu**
- Users Base: 125 organizations from 20 countries
- More than 13,000 downloads from the project site
- RDMA for Apache HBase, Spark and CDH

High-Performance Big Data



THE OHIO STATE UNIVERSITY

Different Modes of RDMA for Apache Hadoop 2.x



- **HHH**: Heterogeneous storage devices with hybrid replication schemes are supported in this mode of operation to have better fault-tolerance as well as performance. This mode is enabled by **default** in the package.
- **HHH-M**: A high-performance in-memory based setup has been introduced in this package that can be utilized to perform all I/O operations in-memory and obtain as much performance benefit as possible.
- **HHH-L**: With parallel file systems integrated, HHH-L mode can take advantage of the Lustre available in the cluster.
- **MapReduce over Lustre, with/without local disks**: Besides, HDFS based solutions, this package also provides support to run MapReduce jobs on top of Lustre alone. Here, two different modes are introduced: with local disks and without local disks.
- **Running with Slurm and PBS**: Supports deploying RDMA for Apache Hadoop 2.x with Slurm and PBS in different running modes (HHH, HHH-M, HHH-L, and MapReduce over Lustre).

RDMA for Apache Hadoop 2.x Distribution

- High-Performance Design of Hadoop over RDMA-enabled Interconnects
  - High performance RDMA-enhanced design with native InfiniBand and RoCE support at the verbs-level for HDFS, MapReduce, and RPC components
  - Enhanced HDFS with in-memory and heterogeneous storage
  - High performance design of MapReduce over Lustre
  - Plugin-based architecture supporting RDMA-based designs for Apache Hadoop and HDP
  - Easily configurable for different running modes (HHH, HHH-M, HHH-L, and MapReduce over Lustre) and different protocols (native InfiniBand, RoCE, and IPoIB)
- Current release: 0.9.7
  - Based on Apache Hadoop 2.6.0
  - Compliant with Apache Hadoop 2.6.0 and HDP 2.2.0.0 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 and Lustre
  - http://hibd.cse.ohio-state.edu

RDMA for Memcached Distribution

- High-Performance Design of Memcached over RDMA-enabled Interconnects
  - High performance RDMA-enhanced design with native InfiniBand and RoCE support at the verbs-level for Memcached and libMemcached components
  - High performance design of SSD-Assisted Hybrid Memory
  - Easily configurable for native InfiniBand, RoCE and the traditional sockets-based support (Ethernet and InfiniBand with IPoIB)
- Current release: 0.9.3
  - Based on Memcached 1.4.22 and libMemcached 1.0.18
  - Compliant with libMemcached APIs and applications
  - Tested with
    - Mellanox InfiniBand adapters (DDR, QDR and FDR)
    - RoCE support with Mellanox adapters
    - Various multi-core platforms
    - SSD
  - **http://hibd.cse.ohio-state.edu**

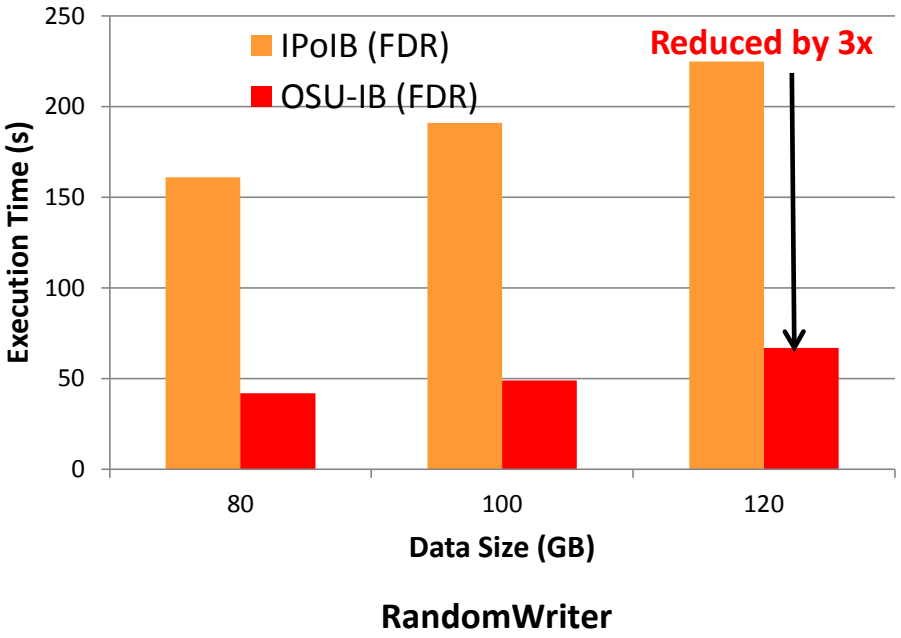
OSU HiBD Micro-Benchmark (OHB) Suite – HDFS & Memcached

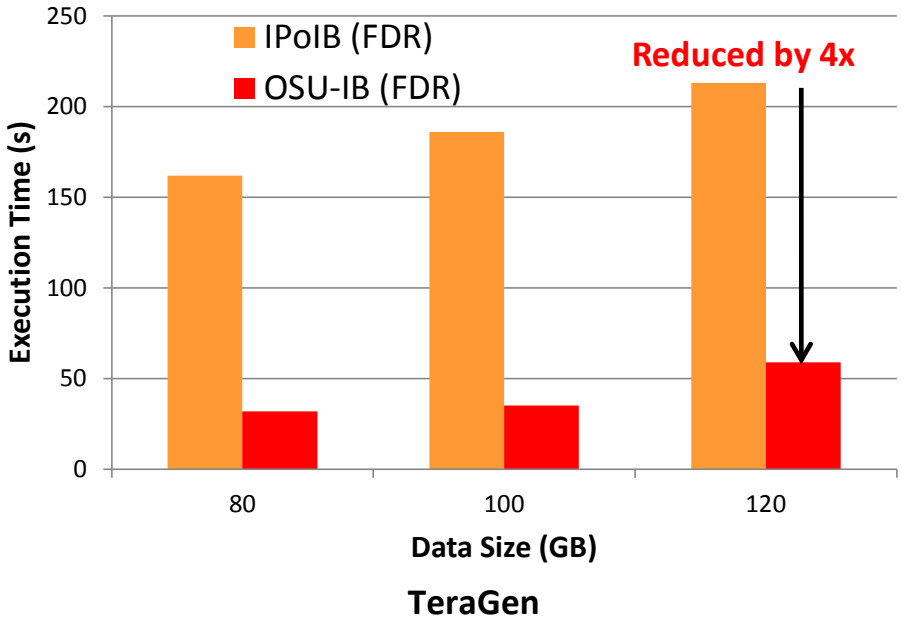
- Micro-benchmarks for Hadoop Distributed File System (HDFS)
  - Sequential Write Latency (**SWL**) Benchmark, Sequential Read Latency (**SRL**) Benchmark, Random Read Latency (**RRL**) Benchmark, Sequential Write Throughput (**SWT**) Benchmark, Sequential Read Throughput (**SRT**) Benchmark
  - Support benchmarking of
    - Apache Hadoop 1.x and 2.x HDFS, Hortonworks Data Platform (HDP) HDFS, Cloudera Distribution of Hadoop (CDH) HDFS
- Micro-benchmarks for Memcached
  - **Get** Benchmark, **Set** Benchmark, and **Mixed** Get/Set Benchmark
- Current release: 0.8
- **http://hibd.cse.ohio-state.edu**

Performance Numbers of RDMA for Apache Hadoop 2.x 0.9.7

- New Features
  - Enhanced HDFS with In-memory and Heterogeneous Storage
    - Standalone, Lustre-integrated
  - Lustre-based designs for MapReduce
- Distributions
  - Apache Hadoop
  - HDP
- Testbeds
  - TACC-Stampede (HDD, FDR)
  - SDSC-Gordon (SSD, QDR)
- Benchmarks
  - TestDFSIO
  - Sort & TeraSort
  - RandomWriter & TeraGen

Performance Benefits – RandomWriter & TeraGen in TACC-Stampede

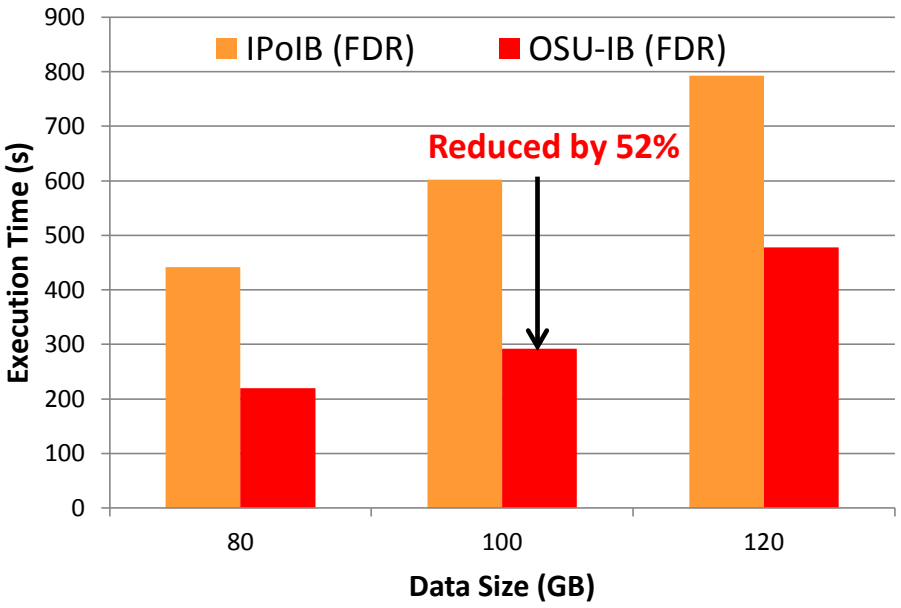




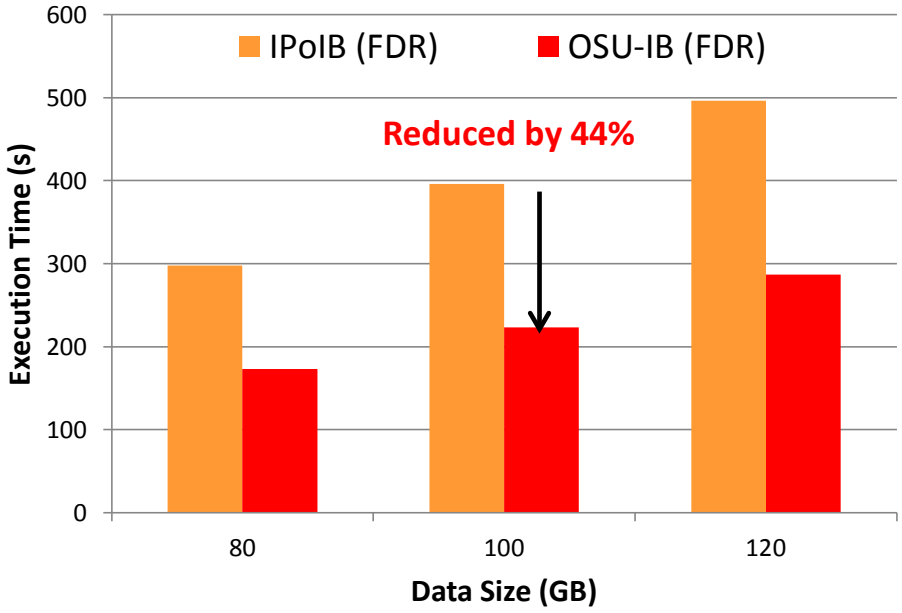
Cluster with 32 Nodes with a total of 128 maps

- RandomWriter
  - **3-4x** improvement over IPoIB for 80-120 GB file size
- TeraGen
  - **4-5x** improvement over IPoIB for 80-120 GB file size

Performance Benefits – Sort & TeraSort in TACC-Stampede



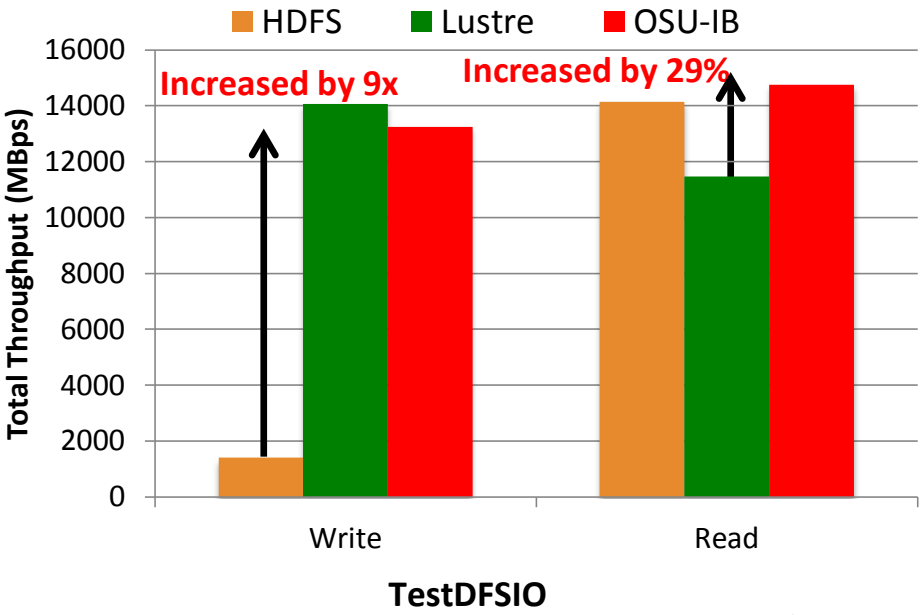
Cluster with 32 Nodes with a total of 128 maps and 57 reduces

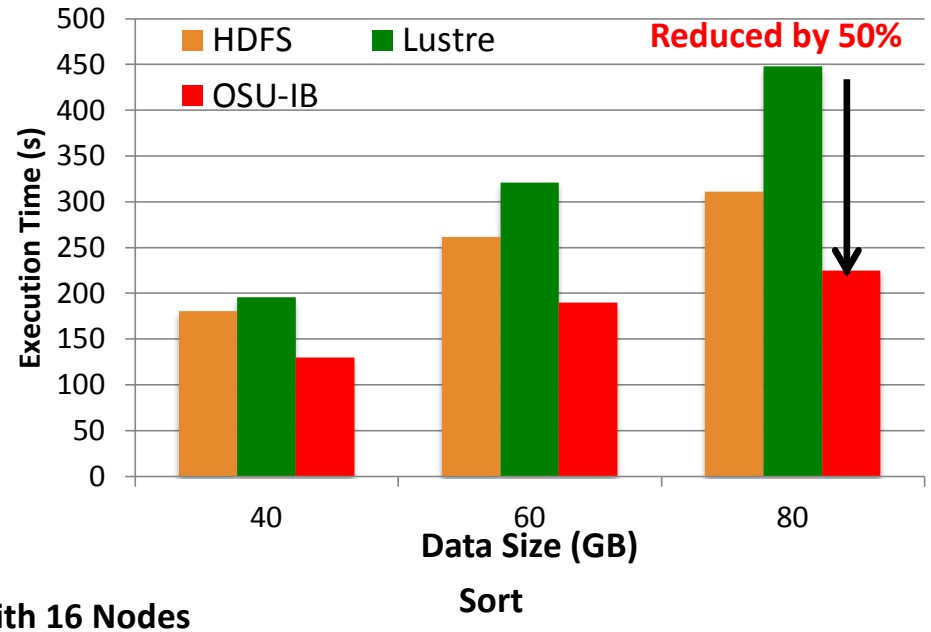


Cluster with 32 Nodes with a total of 128 maps and 64 reduces

- Sort with single HDD per node
  - **40-52%** improvement over IPoIB for 80-120 GB data
- TeraSort with single HDD per node
  - **42-44%** improvement over IPoIB for 80-120 GB data

Performance Benefits – TestDFSIO and Sort in SDSC-Gordon (Lustre-Integrated Mode)

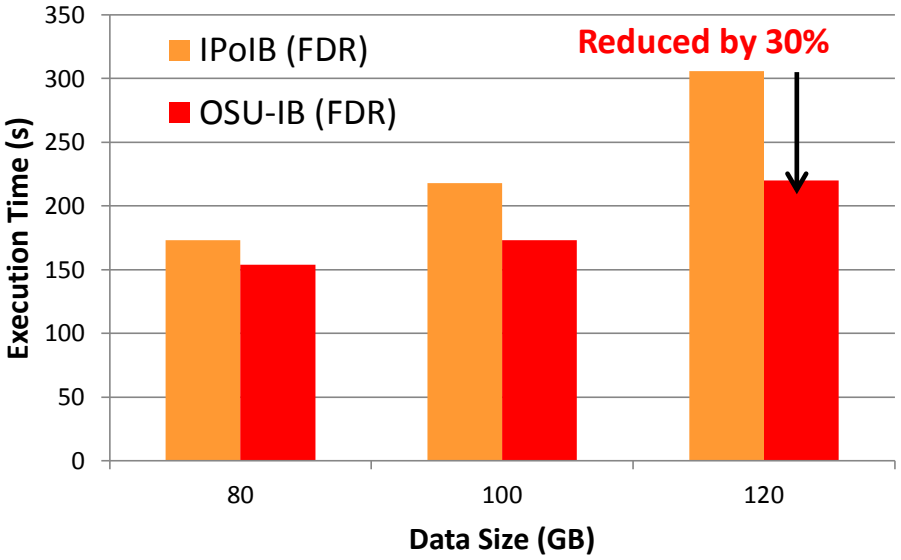




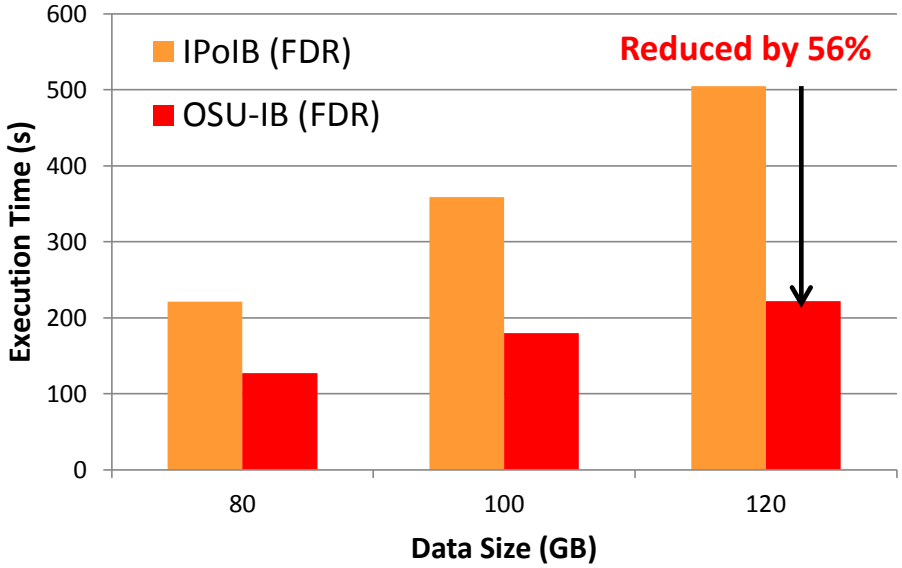
Cluster with 16 Nodes

- TestDFSIO for 80GB data size
  - Write: **9x** improvement over HDFS
  - Read: **29%** improvement over Lustre
- Sort
  - up to **28%** improvement over HDFS
  - up to **50%** improvement over Lustre

Performance Benefits for MR over Lustre – TeraSort & Sort in TACC-Stampede



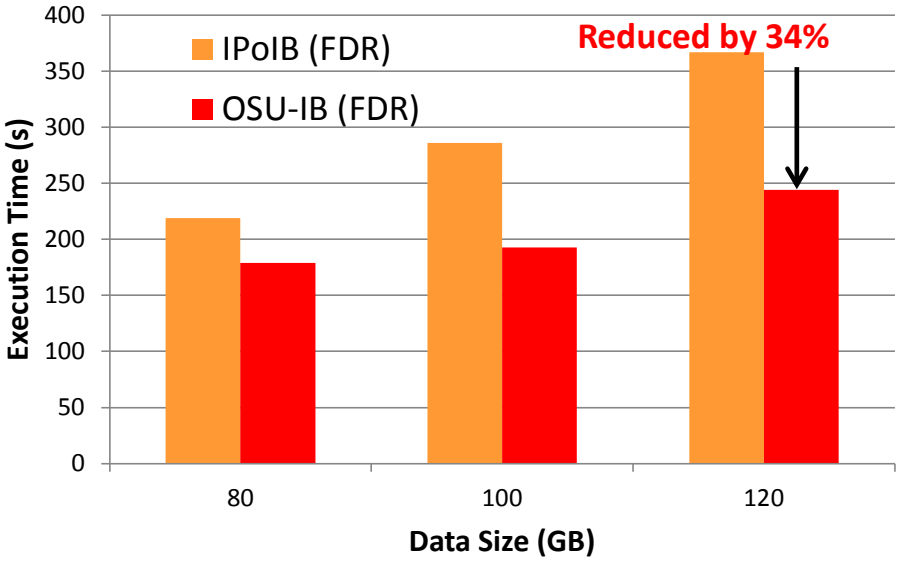
Cluster with 32 Nodes with a total of 128 maps and 64 reduces



Cluster with 32 Nodes with a total of 128 maps and 57 reduces

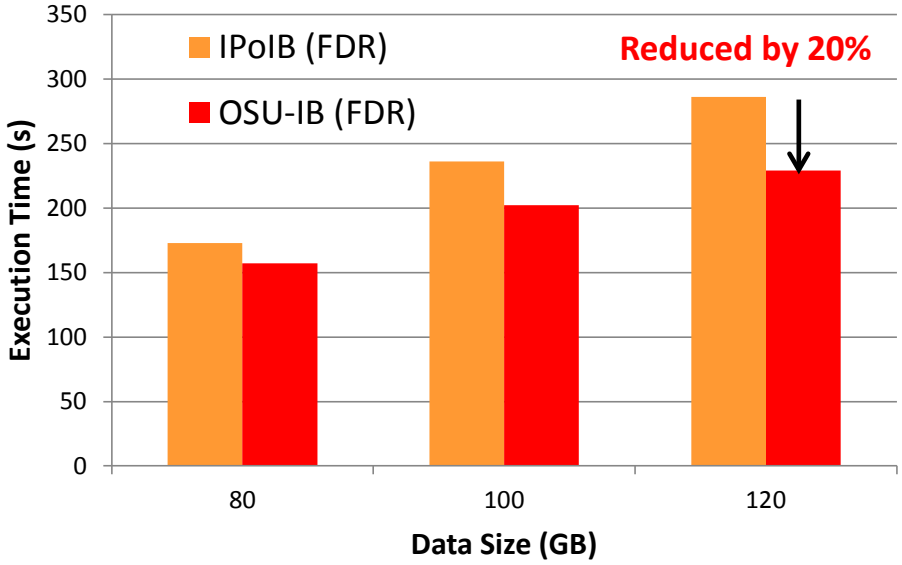
- TeraSort with local disks for intermediate data
  - **11-30%** improvement over IPoIB for 80-120 GB data
- Sort with local disks for intermediate data
  - **43-56%** improvement over IPoIB for 80-120 GB data

Performance Benefits for MR over Lustre – TeraSort & Sort in TACC-Stampede



Cluster with 32 Nodes with a total of 128 maps and 64 reduces

- TeraSort with Lustre for intermediate data
  - **19-34%** improvement over IPoIB for 80-120 GB data



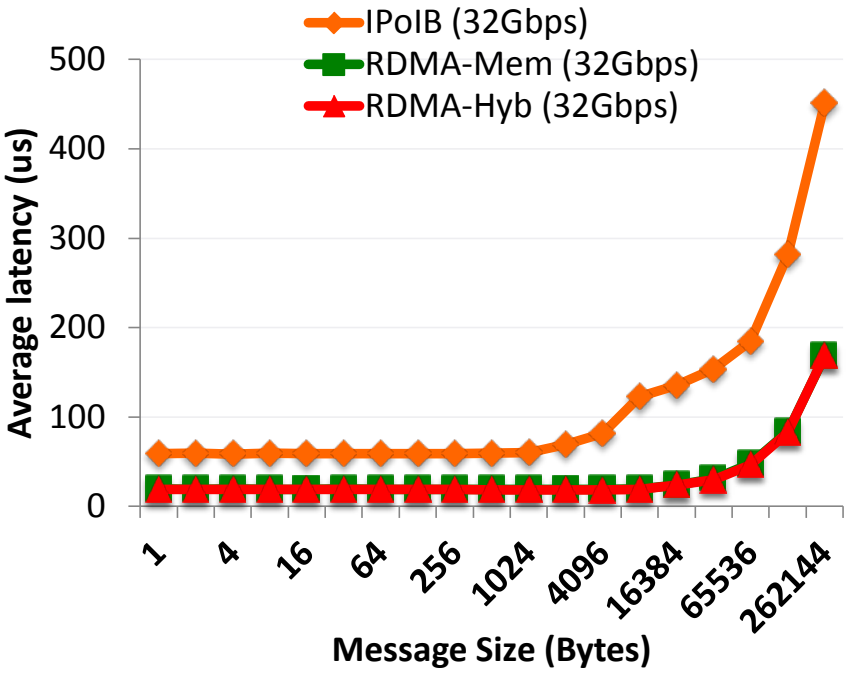
Cluster with 32 Nodes with a total of 128 maps and 57 reduces

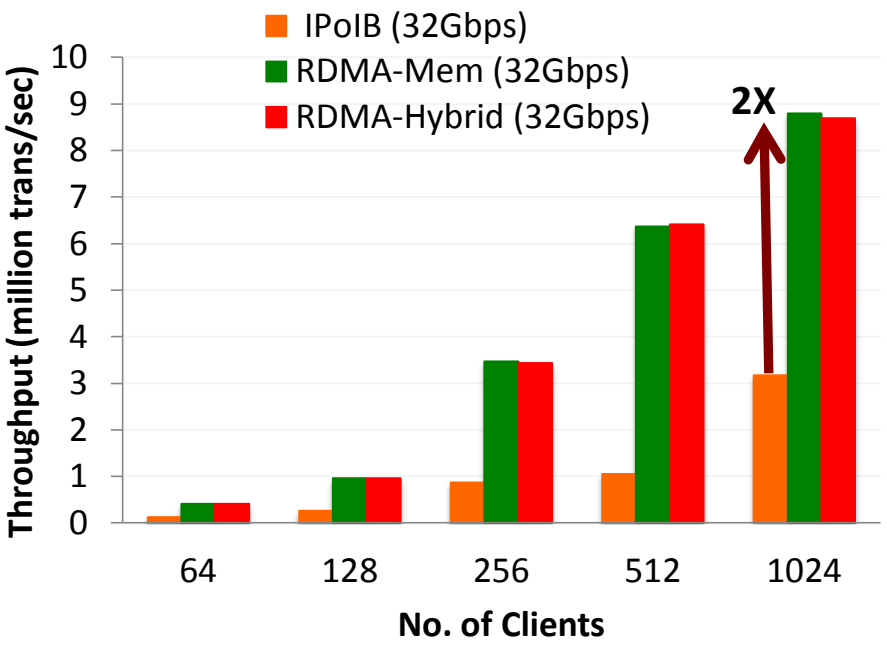
- Sort with Lustre for intermediate data
  - **10-20%** improvement over IPoIB for 80-120 GB data

Performance Numbers of RDMA for Memcached 0.9.3

- New Features
  - SSD-assisted Hybrid Memory
- Testbeds
  - SDSC-Gordon (SSD, QDR)
  - OSU RI Cluster (SSD, QDR)
- Benchmarks
  - ohb_memlat/ohb_memthr
  - ohb_memhybrid

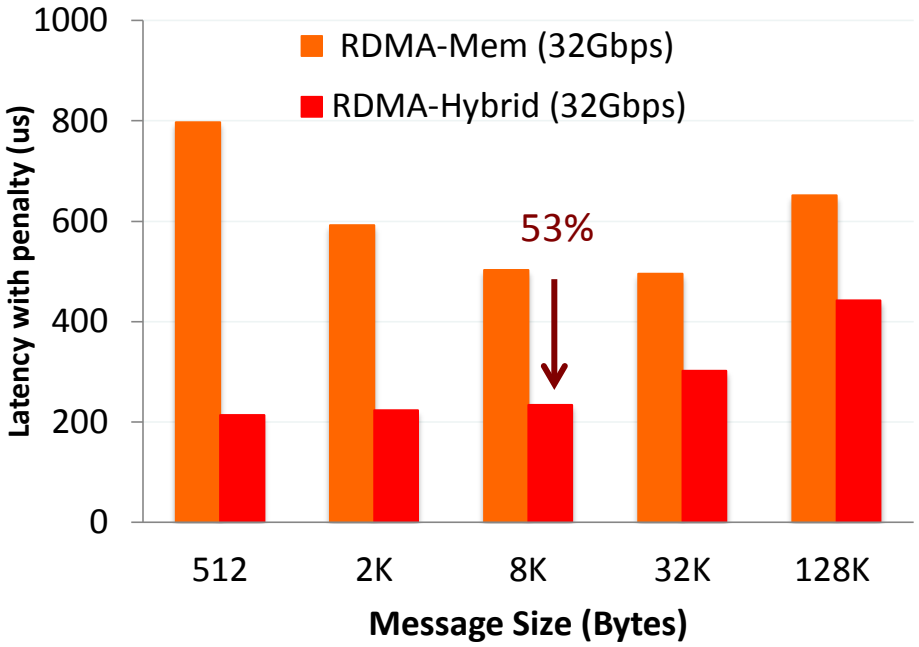
Performance Benefits on SDSC-Gordon – OHB Latency & Throughput Micro-Benchmarks

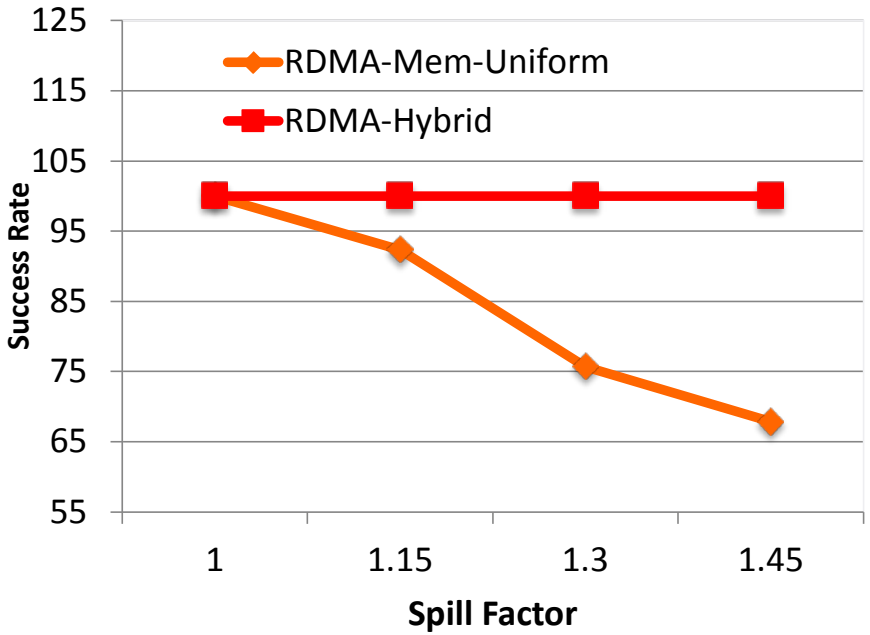




- ohb_memlat & ohb_memthr latency & throughput micro-benchmarks
- Memcached-RDMA can
  - improve query latency by up to 70% over IPoIB (32Gbps)
  - improve throughput by up to 2X over IPoIB (32Gbps)
  - No overhead in using hybrid mode when all data can fit in memory

Performance Benefits on OSU-RI-SSD – OHB Micro-benchmark for Hybrid Memcached





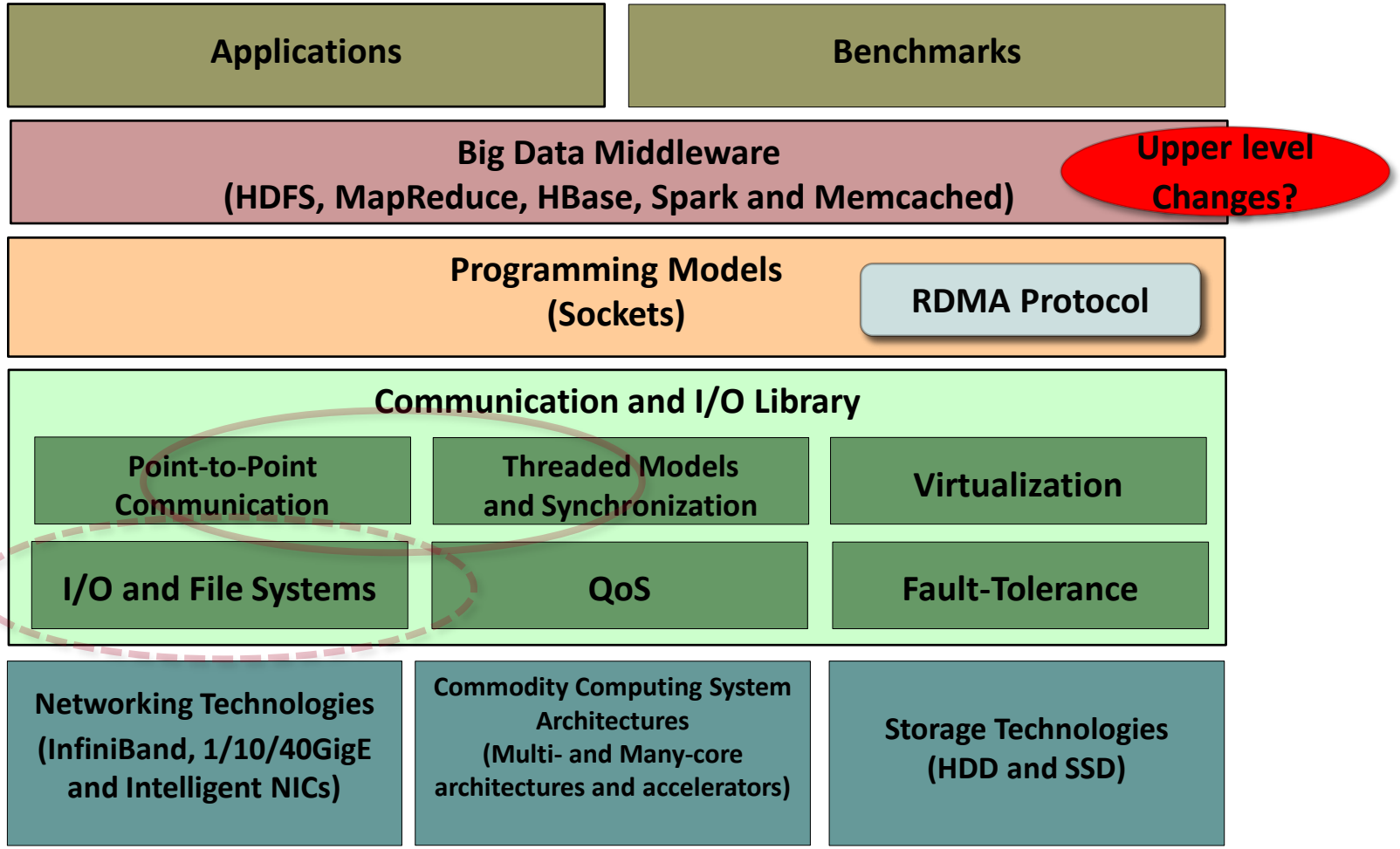
ohb_memhybrid – Uniform Access Pattern, single client and single server with 64MB

- Success Rate of In-Memory Vs. Hybrid SSD-Memory for different spill factors
  - 100% success rate for Hybrid design while that of pure In-memory degrades
- Average Latency with penalty for In-Memory Vs. Hybrid SSD-Assisted mode for spill factor 1.5.
  - up to 53% improvement over In-memory with server miss penalty as low as 1.5 ms

Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

Designing Communication and I/O Libraries for Big Data Systems: Solved a Few Initial Challenges



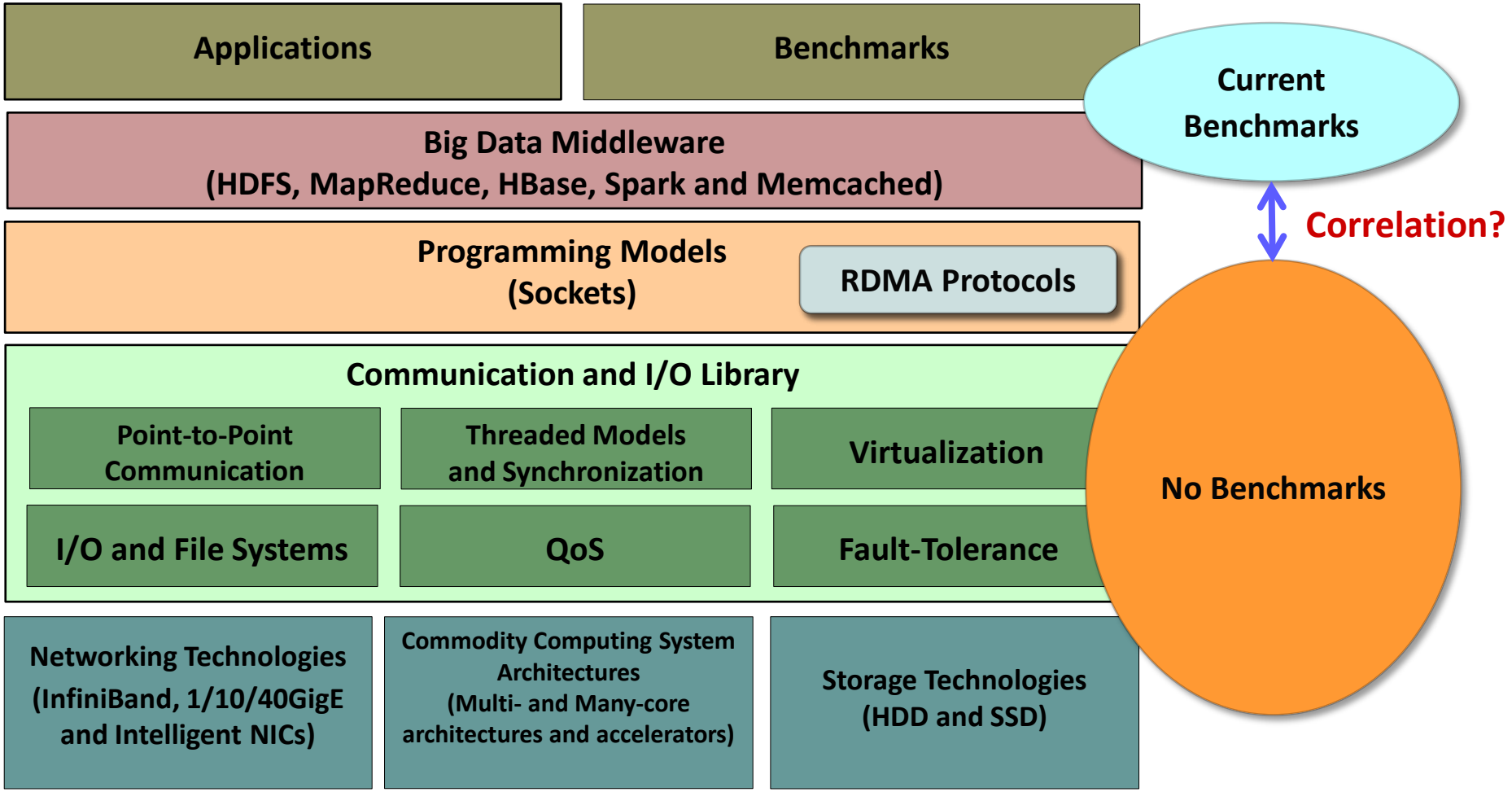
More Challenges

- Multi-threading and Synchronization
  - Multi-threaded model exploration
  - Fine-grained synchronization and lock-free design
  - Unified helper threads for different components
  - Multi-endpoint design to support multi-threading communications
- QoS and Virtualization
  - Network virtualization and locality-aware communication for Big Data middleware
  - Hardware-level virtualization support for End-to-End QoS
  - I/O scheduling and storage virtualization
  - Live migration
- Support of Accelerators
  - Efficient designs for Big Data middleware to take advantage of NVIDA GPGPUs and Intel MICs
  - Offload computation-intensive workload to accelerators
  - Explore maximum overlapping between communication and offloaded computation
- Big Data Benchmarking

Are the Current Benchmarks Sufficient for Big Data?

- The current benchmarks provide some performance behavior
- However, do not provide any information to the designer/developer on:
  - What is happening at the lower-layer?
  - Where the benefits are coming from?
  - Which design is leading to benefits or bottlenecks?
  - Which component in the design needs to be changed and what will be its impact?
  - Can performance gain/loss at the lower-layer be correlated to the performance gain/loss observed at the upper layer?

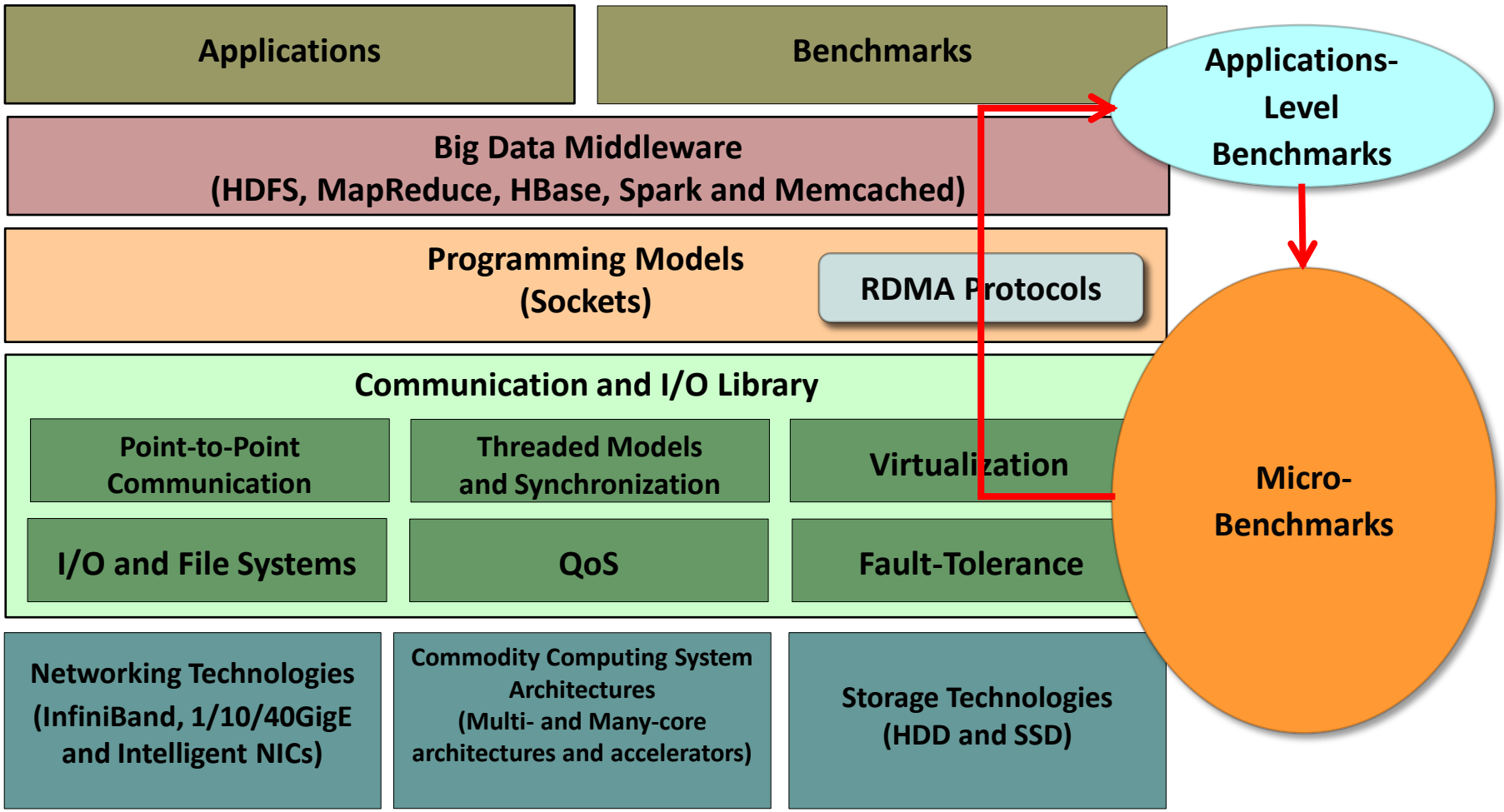
Challenges in Benchmarking of RDMA-based Designs



OSU MPI Micro-Benchmarks (OMB) Suite

- A comprehensive suite of benchmarks to
  - Compare performance of different MPI libraries on various networks and systems
  - Validate low-level functionalities
  - Provide insights to the underlying MPI-level designs
- Started with basic send-recv (MPI-1) micro-benchmarks for latency, bandwidth and bi-directional bandwidth
- Extended later to
  - MPI-2 one-sided
  - Collectives
  - GPU-aware data movement
  - OpenSHMEM (point-to-point and collectives)
  - UPC
- Has become an industry standard
- Extensively used for design/development of MPI libraries, performance comparison of MPI libraries and even in procurement of large-scale systems
- Available from http://mvapich.cse.ohio-state.edu/benchmarks
- Available in an integrated manner with MVAPICH2 stack

Iterative Process – Requires Deeper Investigation and Design for Benchmarking Next Generation Big Data Systems and Applications



Upcoming HiBD Releases and Future Activities

- Upcoming Releases of RDMA-enhanced Packages will support
  - CDH plugin
  - Spark
  - HBase
- Upcoming Releases of OSU HiBD Micro-Benchmarks (OHB) will support
  - MapReduce, RPC
- Exploration of other components (Threading models, QoS, Virtualization, Accelerators, etc.)
- Advanced designs with upper-level changes and optimizations

Presentation Outline

- Overview
  - MapReduce and RDD Programming Models
  - Apache Hadoop, Spark, and Memcached
  - Modern Interconnects and Protocols
- Challenges in Accelerating Hadoop, Spark, and Memcached
- Benchmarks and Applications using Hadoop, Spark, and Memcached
- Acceleration Case Studies and In-Depth Performance Evaluation
- The High-Performance Big Data (HiBD) Project and Associated Releases
- Ongoing/Future Activities for Accelerating Big Data Applications
- Conclusion and Q&A

Concluding Remarks

- Presented an overview of Big Data, Hadoop (MapReduce, HDFS, HBase, Spark, RPC) and Memcached
- Provided an overview of Networking Technologies
- Discussed challenges in accelerating Hadoop and Memcached
- Presented initial designs to take advantage of InfiniBand/RDMA for HDFS, MapReduce, HBase, Spark, RPC and Memcached
- Results are promising
- Many other open issues need to be solved
- Will enable Big Data community to take advantage of modern HPC technologies to carry out their analytics in a fast and scalable manner

Personnel Acknowledgments

Current Students

- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- A. Bhat (M.S.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- N. Islam (Ph.D.)
- K. Kulkarni (M.S.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Past Students

- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Post-Docs

- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne

Current Senior Research Associates

- K. Hamidouche
- X. Lu
- H. Subramoni

Current Post-Doc

- J. Lin
- D. Shankar

Current Programmer

- J. Perkins

Current Research Specialist

- M. Arnold

Past Research Scientist

- S. Sur

Past Programmers

- D. Bureddy

International Workshop on High-Performance Big Data Computing (HPBDC 2015)

Held with Int'l Conference on Distributed Computing Systems (ICDCS '15)

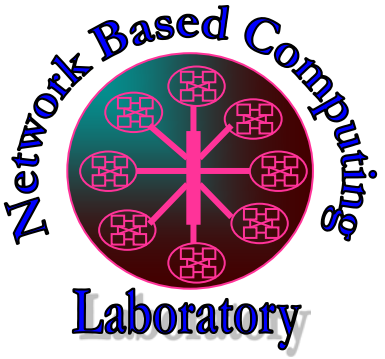
In Hilton Downtown, Columbus, Ohio, USA, Monday, June 29th, 2015

Two Keynote Talks: Dan Stanzione (TACC) and Zhiwei Xu (ICT/CAS)
Two Invited Talks: Jianfeng Zhan (ICT/CAS), Raghunath Nambiar (Cisco)
Panel: Jianfeng Zhan (ICT/CAS)
Four Research Papers

http://web.cse.ohio-state.edu/~luxi/hpbdc2015

Thank You!

{panda, luxi}@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/