Hadoop over NDN: Initial Experience and Results

Mathias Gibbens, Lei Ye, Chris Gniady, and Beichuan Zhang

The University Of Arizona

Overview

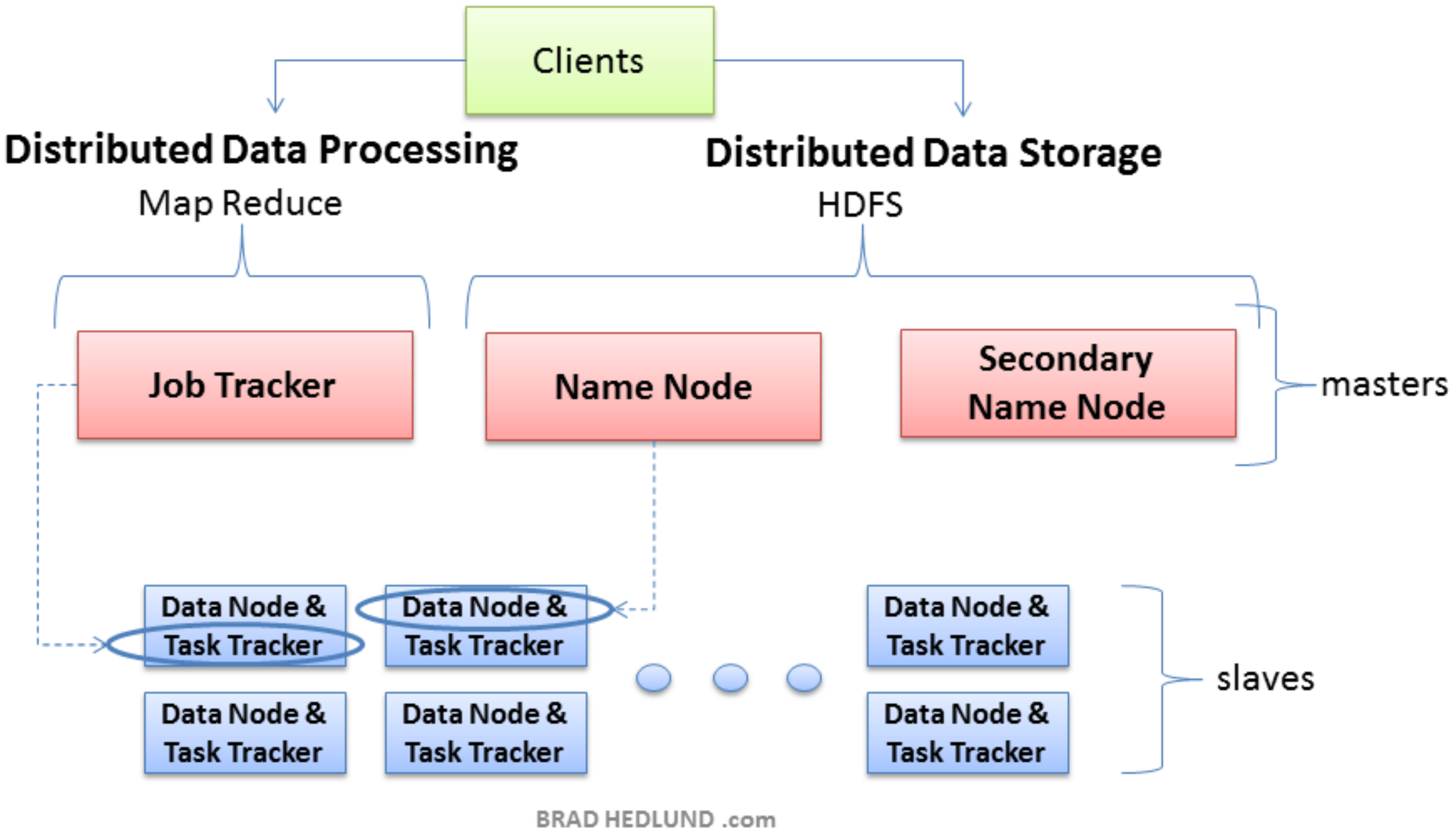
The research goal: apply NDN to the data center network environment to improve the storage, access, and processing of large amount of data.

The current work: modify Hadoop to run on top of NDN to establish performance baseline, and collect research problems, still work in process.

The next step: design NDN-native distributed filesystem and network mechanisms to improve system performance and resiliency.

What is Hadoop

A popular MapReduce framework for distributed storage and processing of large data sets.



http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/

Hadoop Distributed File System

By default, data is stored in the Hadoop Distributed File System (HDFS), in the unit of Blocks.

HDFS replicates each Block to three different DataNodes along with checksums to ensure data integrity

Cluster-wide consistent states provided by NameNode

- Maintain states of entire HDFS
- All requests of data placement and retrieval go through it.
- Receiving heartbeats from DataNodes and initiate recovery after failures detected.

Why Hadoop over NDN

Hadoop is a complex piece of software that requires non-trivial configuration and tuning for good performance.

NDN can improve the performance

- Caching, multicast, multi-path and multi-source data retrieval.

Increase resiliency and failure handling

- Get data from any working node that stores the data
- Interest-data feedback loop to quickly detect failures and adapt to them by forwarding strategy

Simplify implementation

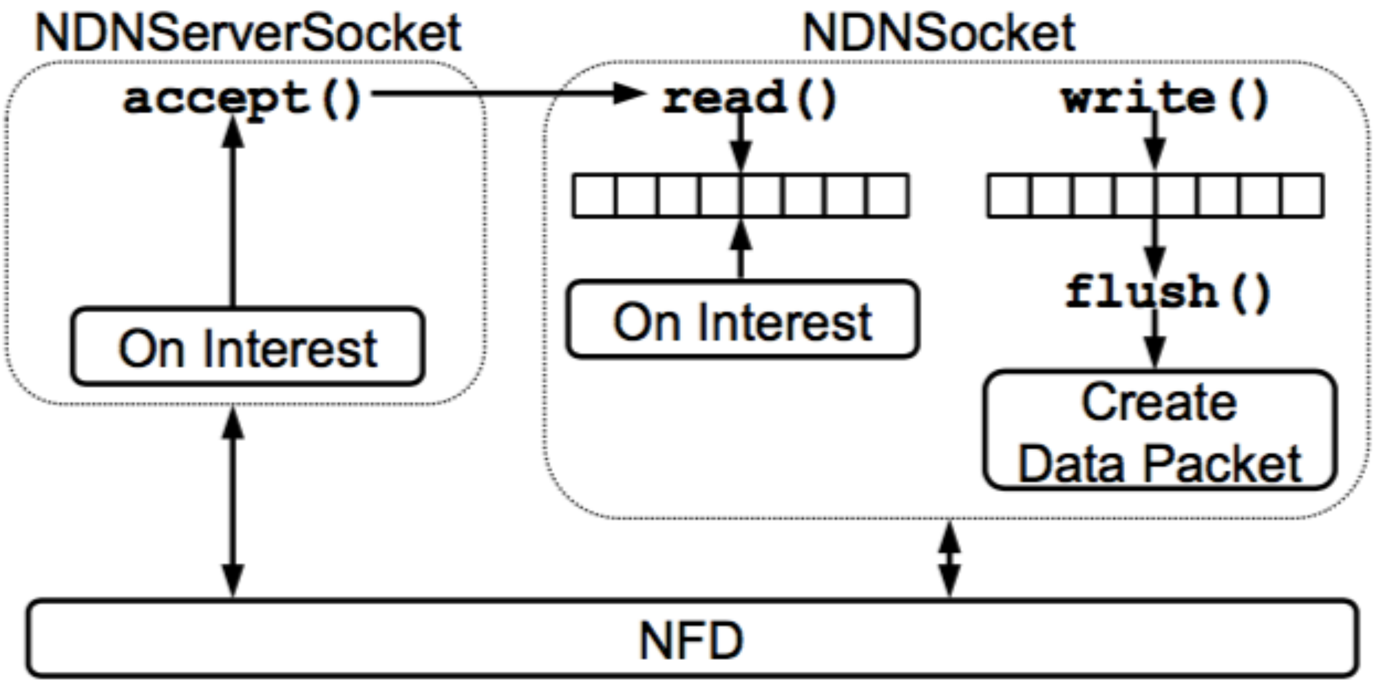
- Many network-related functions are handled by NDN.

Signature for data integrity and security

Making Hadoop running on NDN

A challenging task to modify a complex piece of software

- As the first step, simply convert all the communication to "NDN Sockets" using address/port in the names.
- Future work is to make the application logic NDN-native.



Making Hadoop running on NDN

Remote Procedure Calls (RPC)

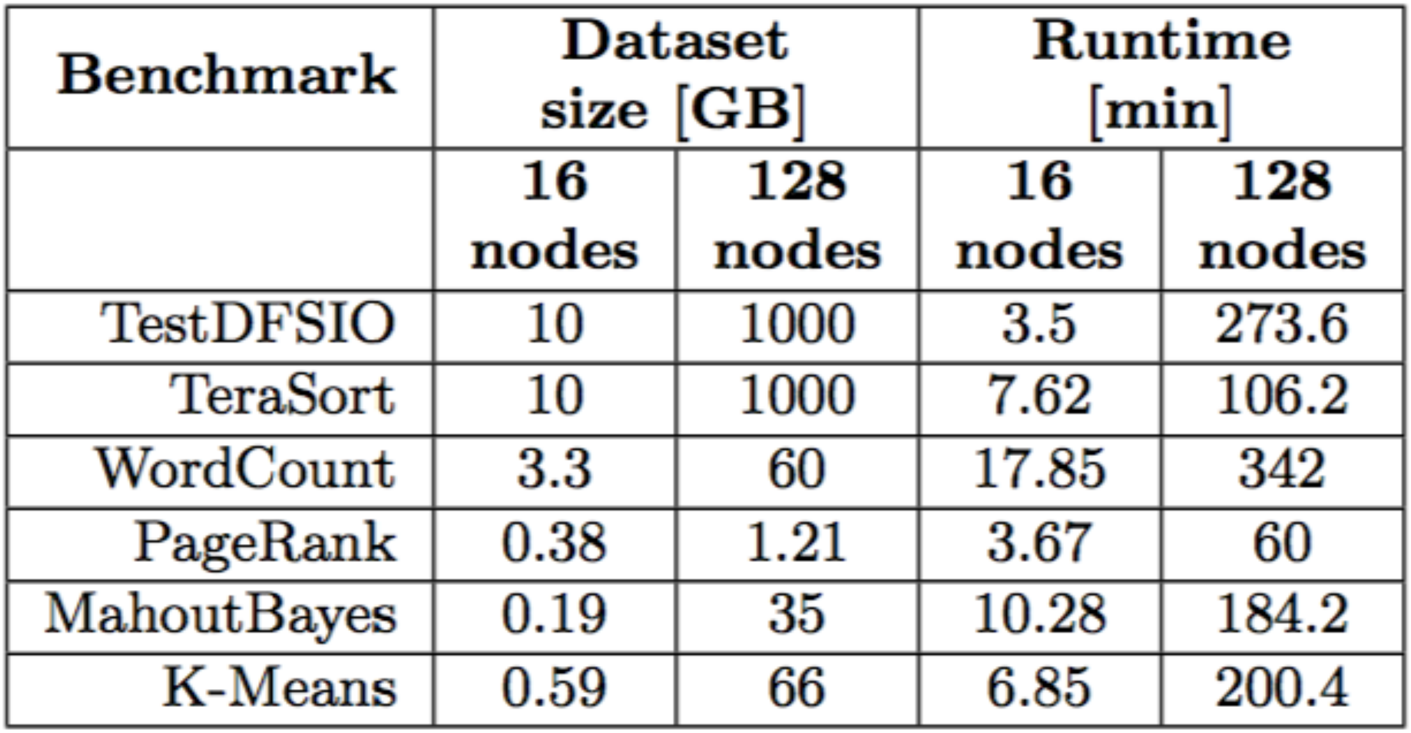
- Used between NameNode and DataNodes
- RPC requests and responses can be naturally mapped to NDN Interests and Data.
    - A name contains address, port, timestamp, and nonce to make it unique.

TCP data transfer

- Between DataNodes for bulk data transfer
- Writing a Block in HDFS requires 2 other replicas.
- Need to convert the “push” model to “pull”, which becomes multicast to the replicas.

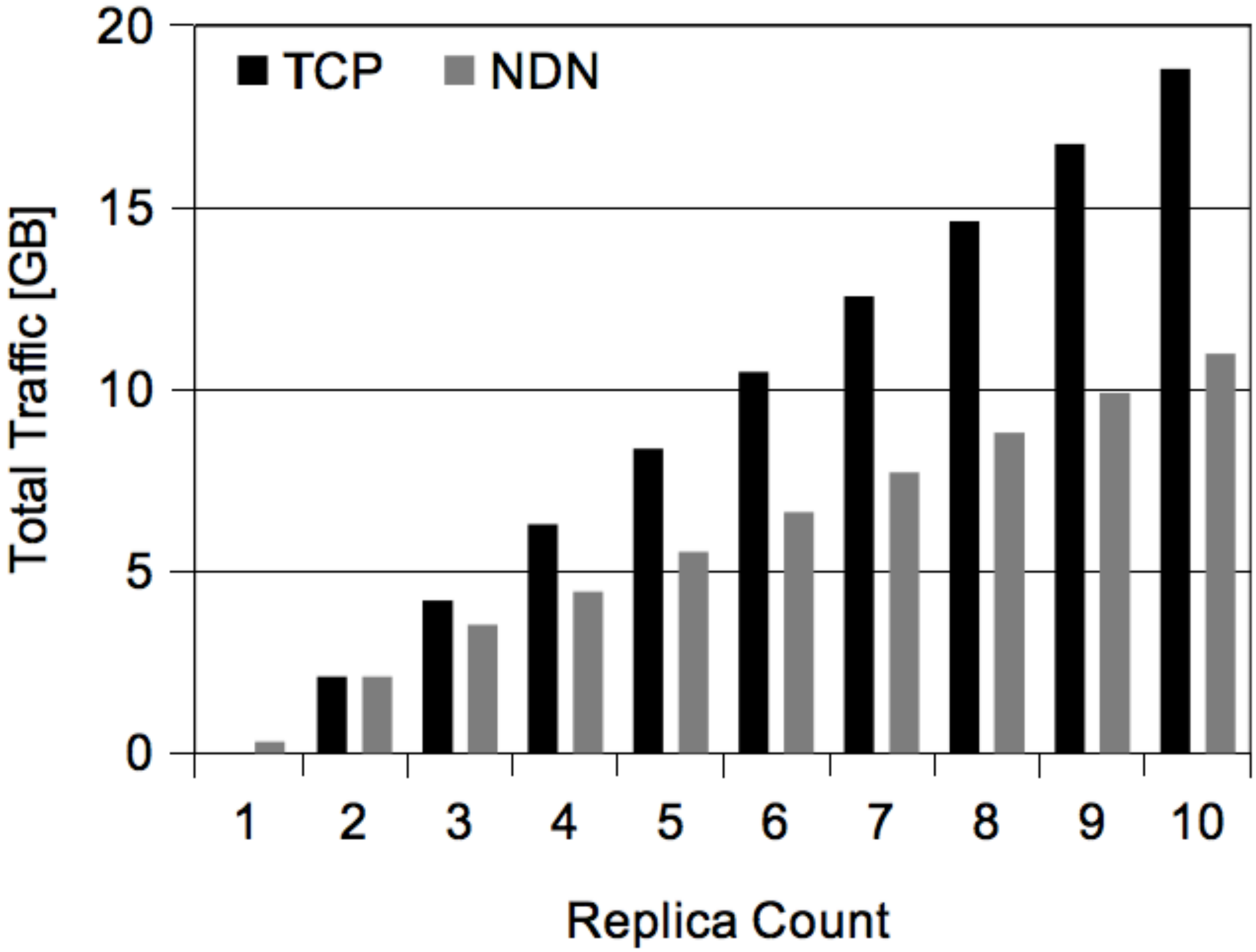
Experiments

Run a diverse set of benchmarks on two Hadoop clusters.

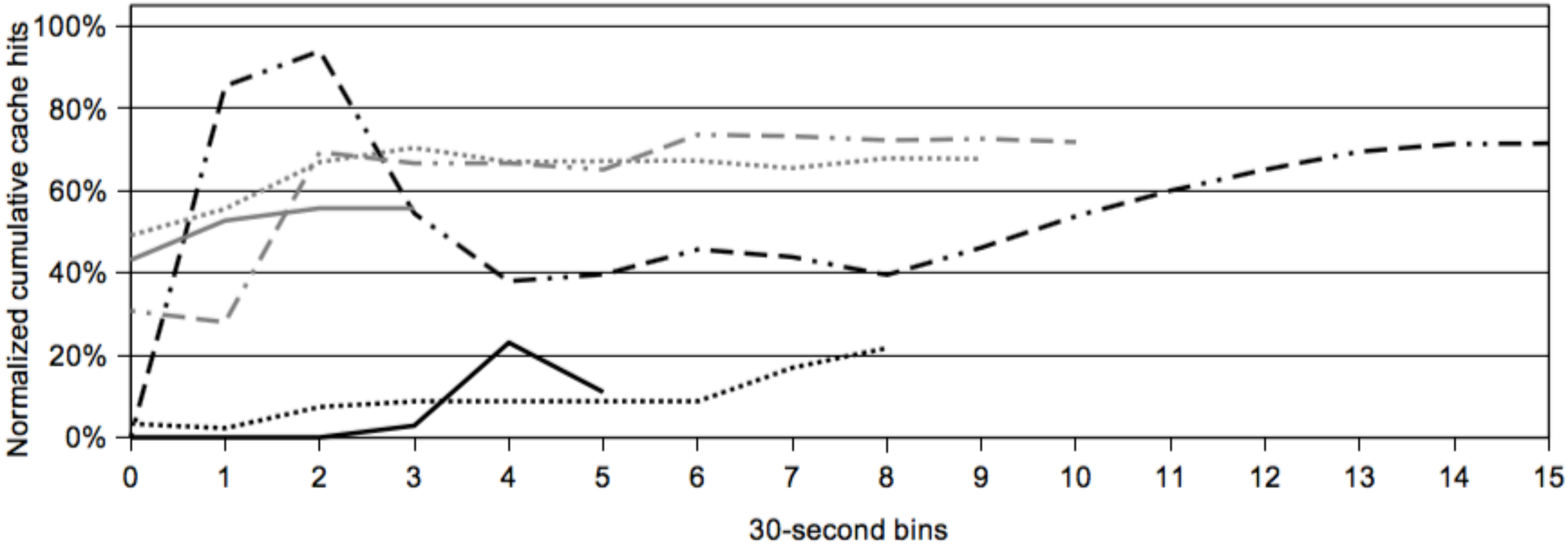


Benchmark	Dataset size [GB]		Runtime [min]	
	16 nodes	128 nodes	16 nodes	128 nodes
TestDFSIO	10	1000	3.5	273.6
TeraSort	10	1000	7.62	106.2
WordCount	3.3	60	17.85	342
PageRank	0.38	1.21	3.67	60
MahoutBayes	0.19	35	10.28	184.2
K-Means	0.59	66	6.85	200.4

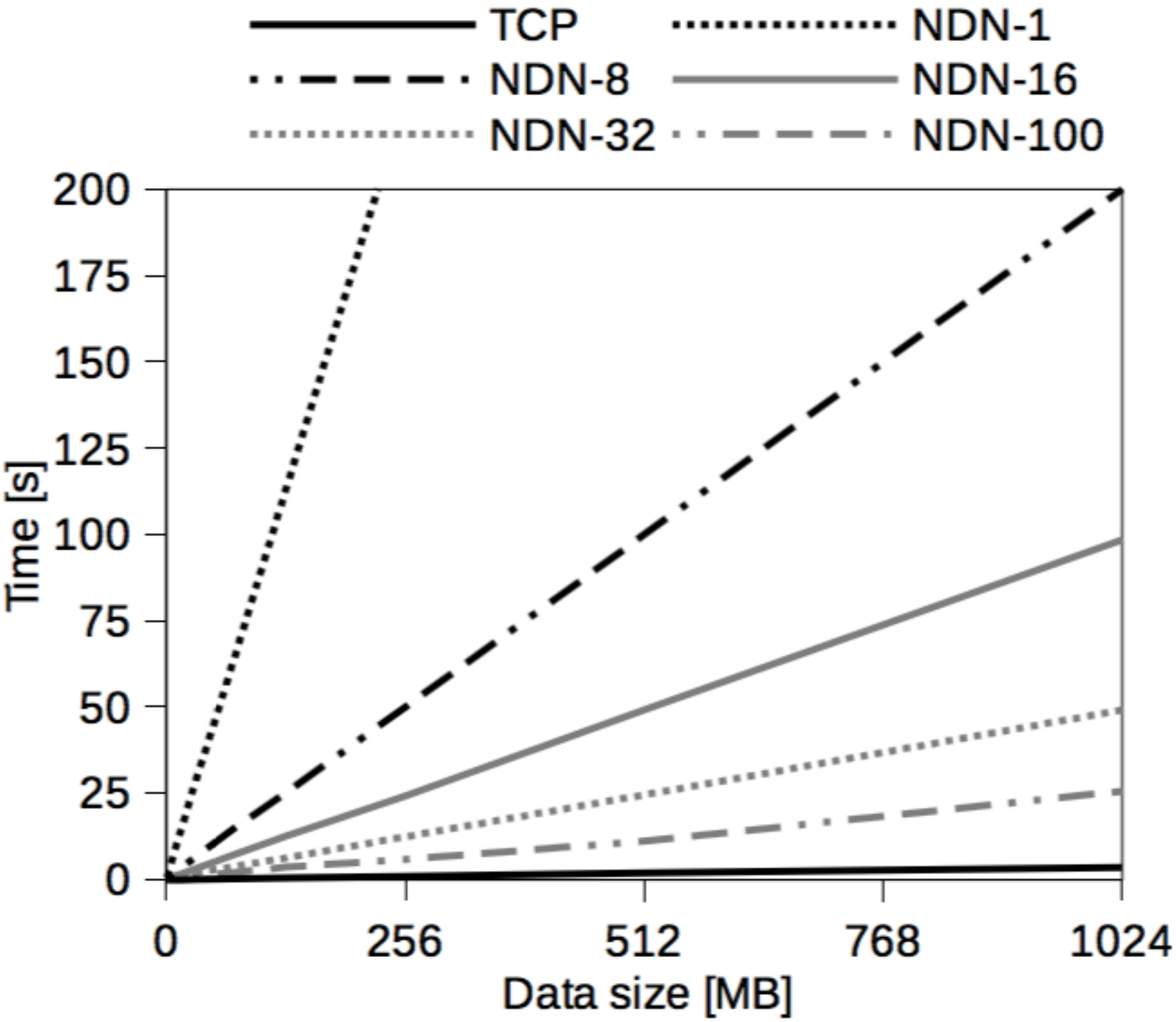
Writing 1GB data



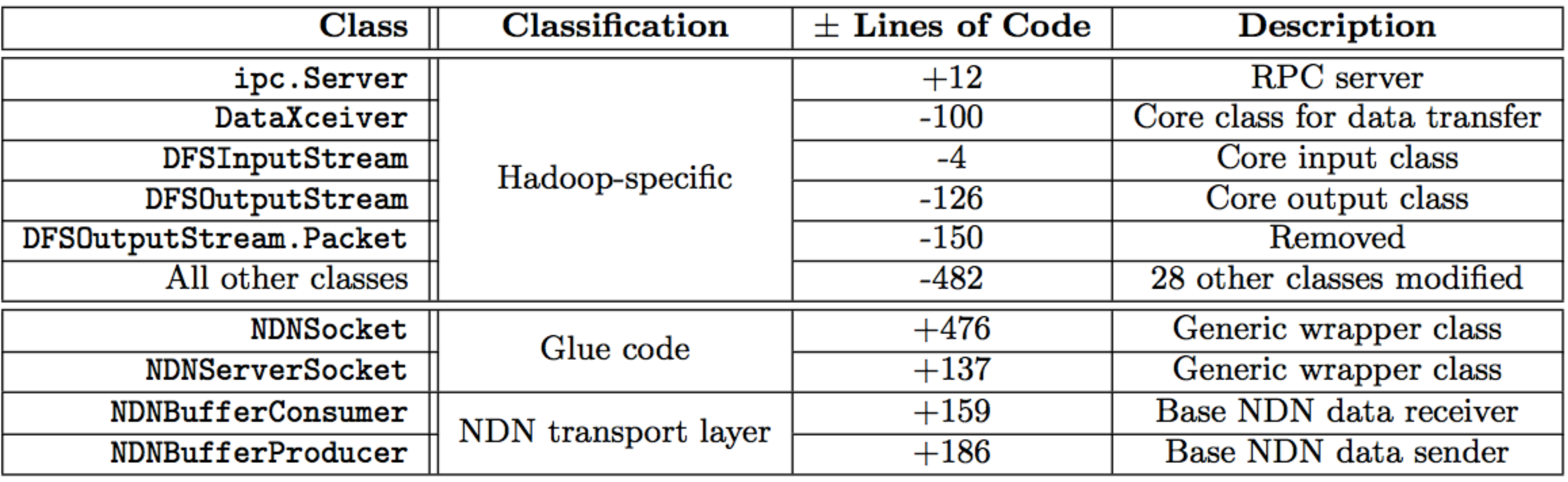
Cache hit over 30-second bins



A missing piece: congestion control



Code changes



Class	Classification	± Lines of Code	Description
ipc.Server	Hadoop-specific	+12	RPC server
DataXceiver		-100	Core class for data transfer
DFSInputStream		-4	Core input class
DFSOutputStream		-126	Core output class
DFSOutputStream.Packet		-150	Removed
All other classes		-482	28 other classes modified
NDNSocket	Glue code	+476	Generic wrapper class
NDNServerSocket		+137	Generic wrapper class
NDNBufferConsumer	NDN transport layer	+159	Base NDN data receiver
NDNBufferProducer		+186	Base NDN data sender

Conclusions

Opportunities for traffic reduction

- Caching and multicast.

Other potentials

- Multipath, multi-source data transfer
- Resiliency: failure detection and recovery
- Code simplification

Challenges

- Routing, forwarding strategy, etc. to realize the potentials.

Comments and Suggestions?