Tachyon: Reliable File Sharing at Memory-speed Across Cluster Frameworks

Haoyuan Li, Ali Ghodsi, Matei Zaharia, Eric Baldeschwieler, Scott Shenker, Ion Stoica (UC Berkeley)

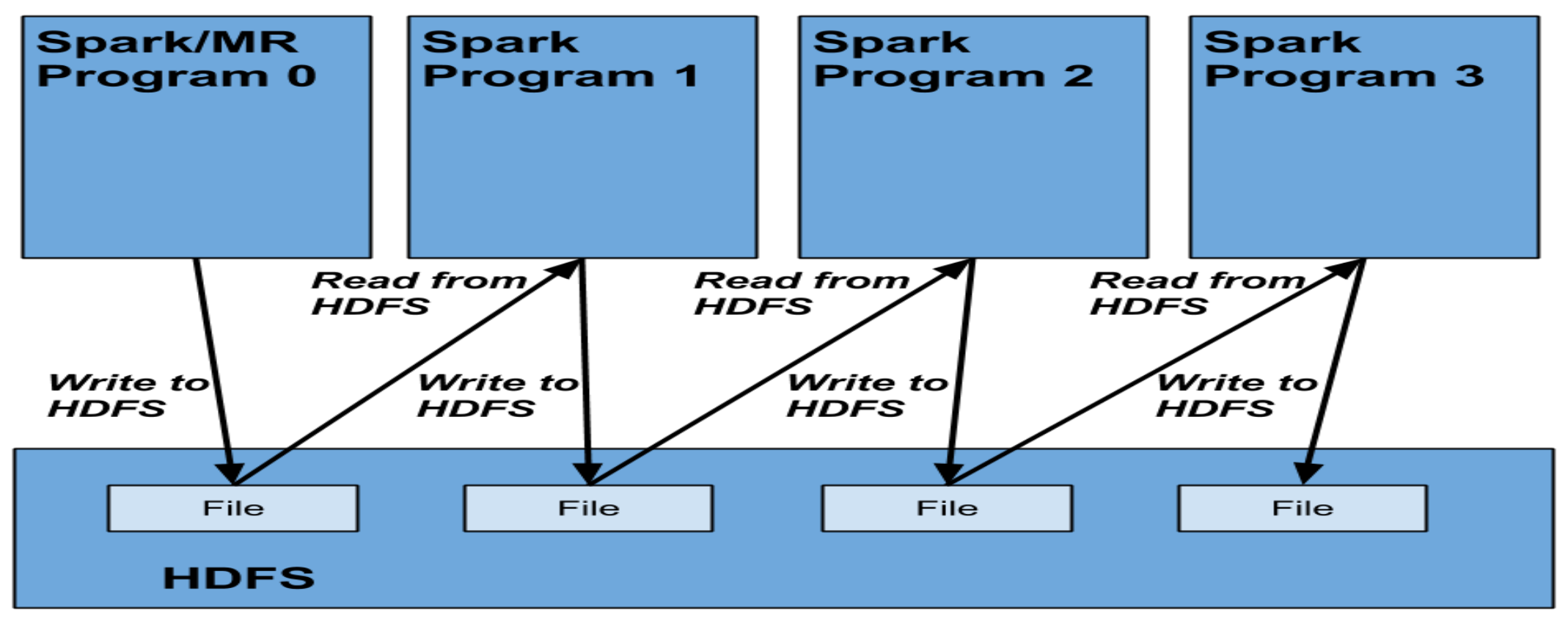
Motivation

- Storage throughput has long been the bottleneck for I/O intensive applications.
- Hardware trend suggests existing solutions would not improve significantly.
- This overhead starts to dominate the running times of many applications.
- In data analytical workloads, files are commonly immutable while tasks are stateless and deterministic

Existing Systems

- Read: HDFS reads from disk. In memory K/V stores are optimized for small value lookup, but not large sequential read.
- Write: Replication based fault tolerant is the bottleneck.

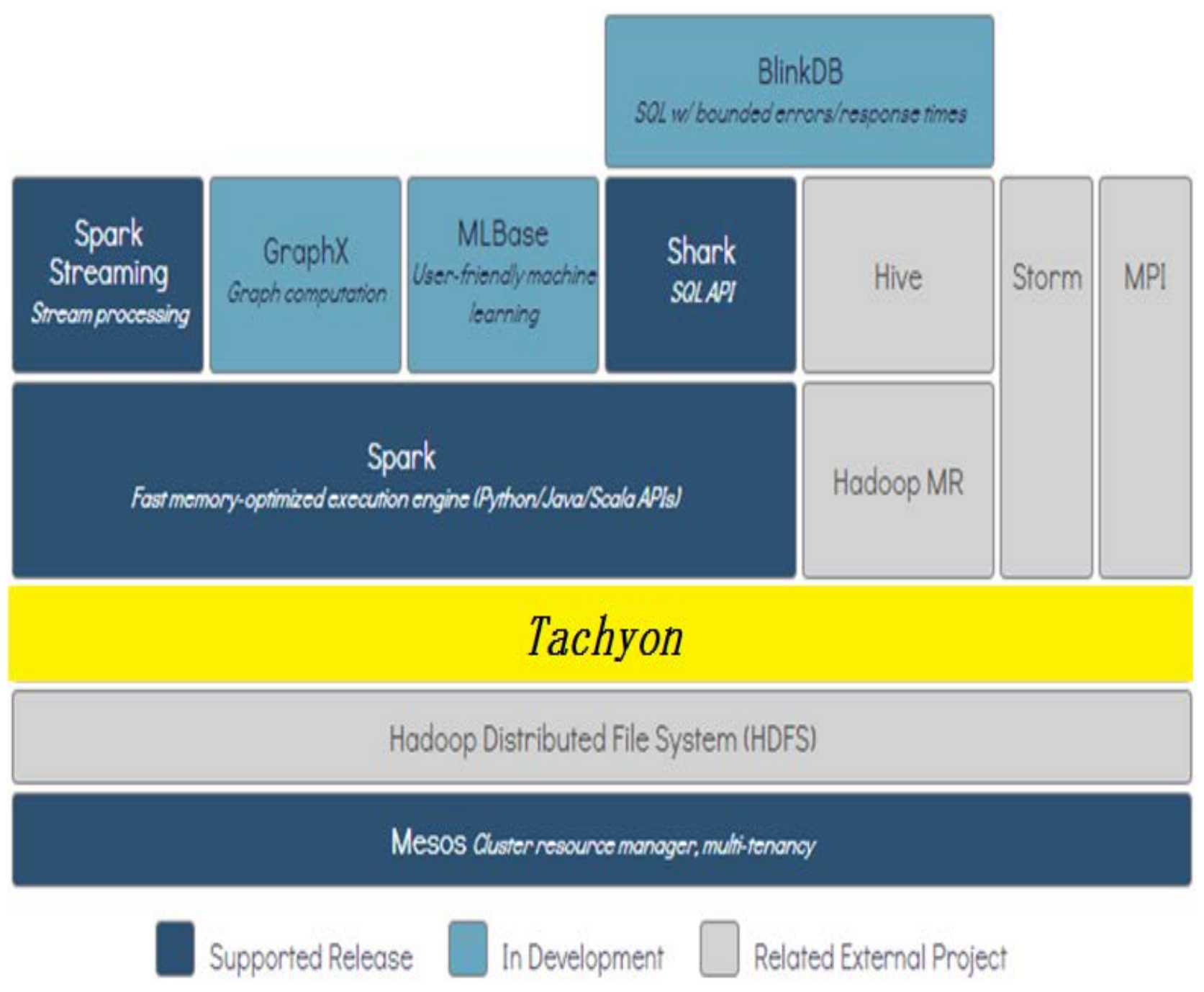
Example



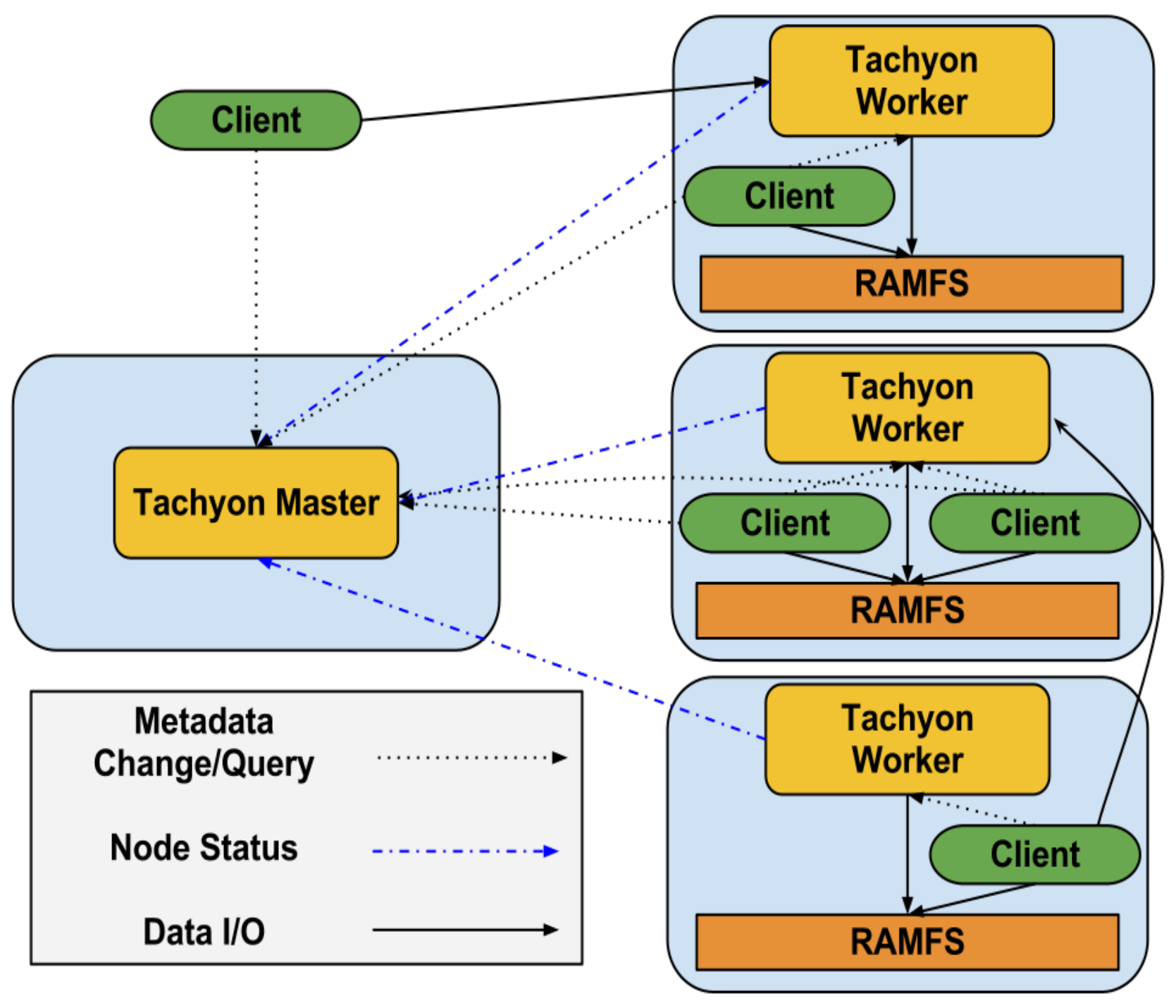
Spark/MR Program 0
Spark Program 1
Spark Program 2
Spark Program 3
Read from HDFS
Write to HDFS
File
HDFS

Goal: reliable file sharing at memory speed across cluster frameworks

Architecture



BlinkDB *SQL w/ bounded errors/response times*
Spark Streaming *Stream processing*
GraphX *Graph computation*
MLBase *User-friendly machine learning*
Shark *SQL API*
Hive
Storm
MPI
Spark *Fast memory-optimized execution engine (Python/Java/Scala APIs)*
Hadoop MR
Tachyon
Hadoop Distributed File System (HDFS)
Mesos *Cluster resource manager, multi-tenancy*
Supported Release
In Development
Related External Project

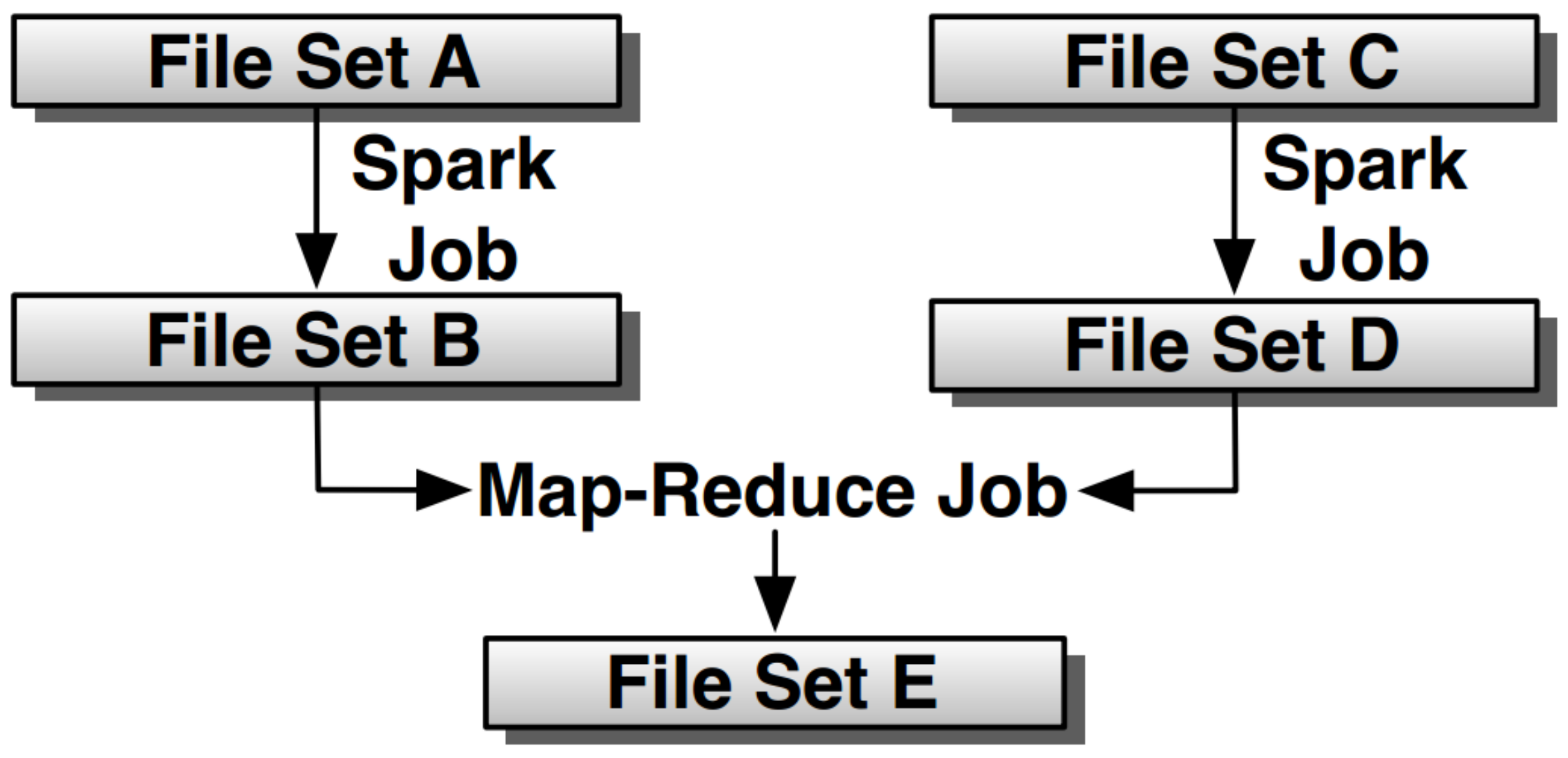


Client
Tachyon Worker
RAMFS
Tachyon Master
Metadata Change/Query
Node Status
Data I/O

Challenges

How to achieve reliable file sharing without replication?

Recomputation Based Fault Recovery



File Set A
Spark Job
File Set B
File Set C
Spark Job
File Set D
Map-Reduce Job
File Set E

A Lineage Example

Generic Lineage API

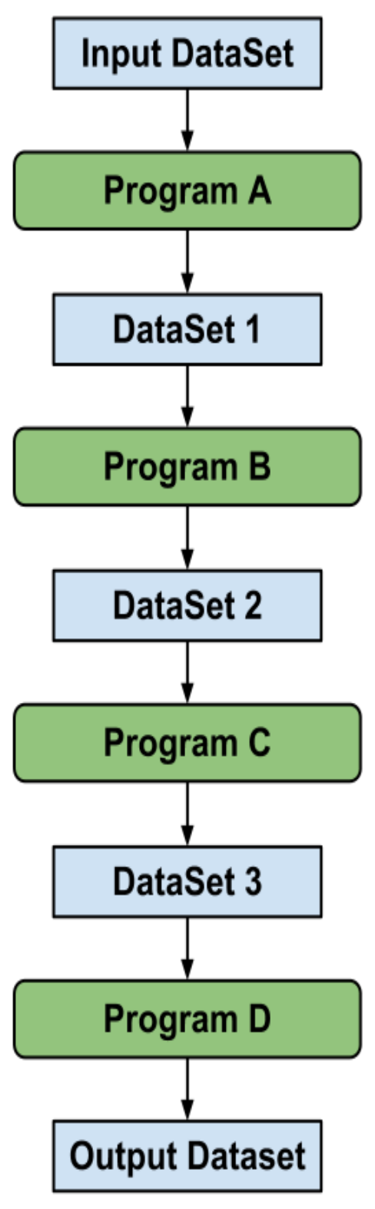
Lineage Required Information:

- Input, Output Files
- Binary Program
- Configuration: various configuration formats
- Dependency Type: wide, narrow
- Tachyon provides API to capture above information
- Spark(200-line patch), MapReduce(200-line patch)
- No burden on application programmers

Recomputation Cost

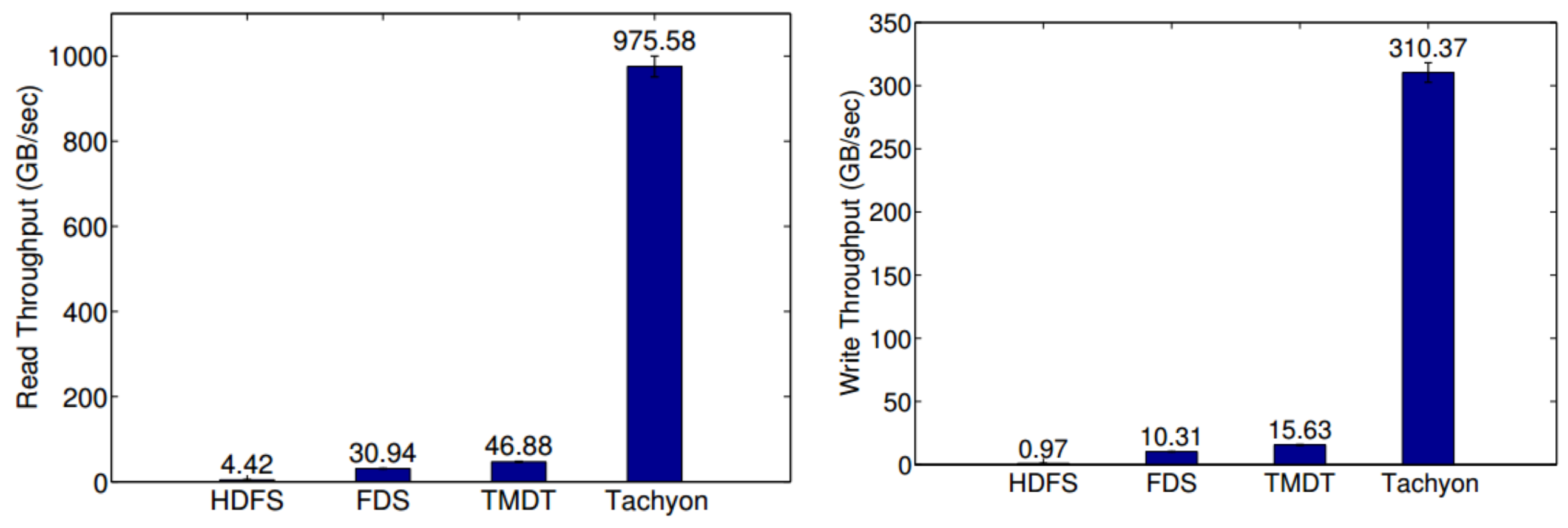
Checkpoint in the background

- Naïve: checkpoint based on creation order.
- Better: Snapshot checkpointing.
- Even under Naïve algorithm, performance is better than HDFS under failures.
- Under snapshot checkpointing, the recovery time is bounded.

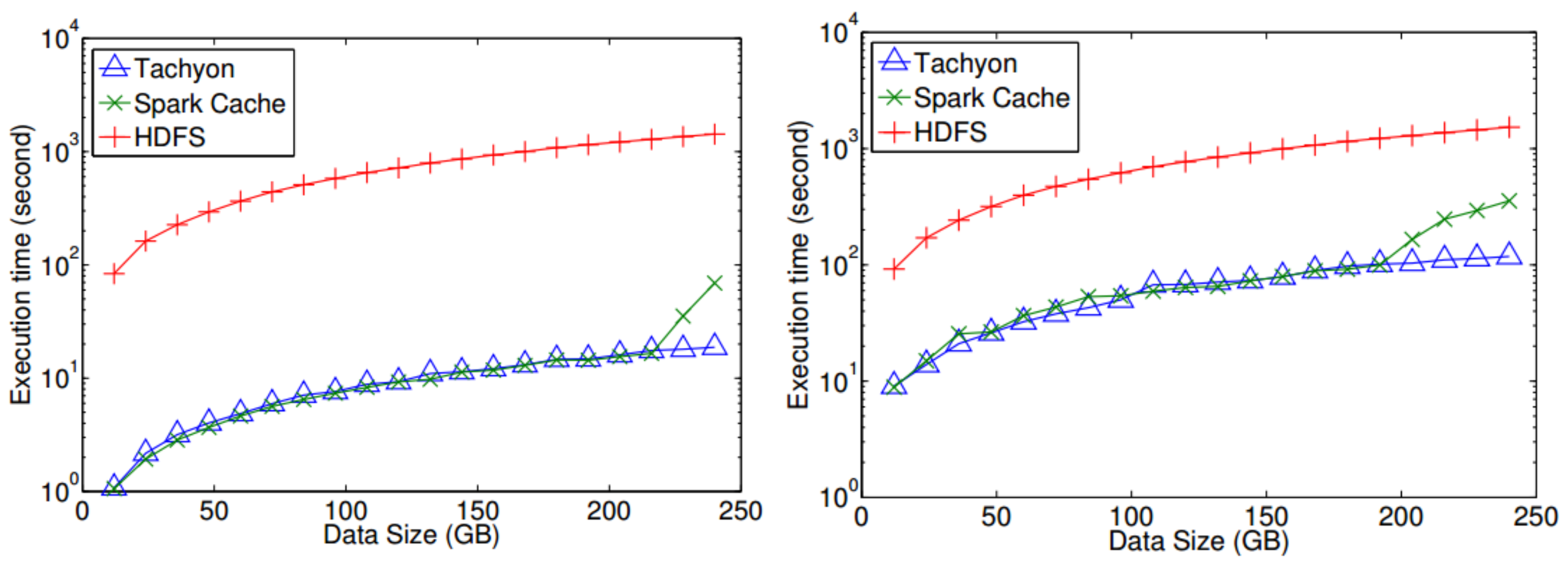


Input DataSet
Program A
DataSet 1
Program B
DataSet 2
Program C
DataSet 3
Program D
Output Dataset

Read/Write Throughput



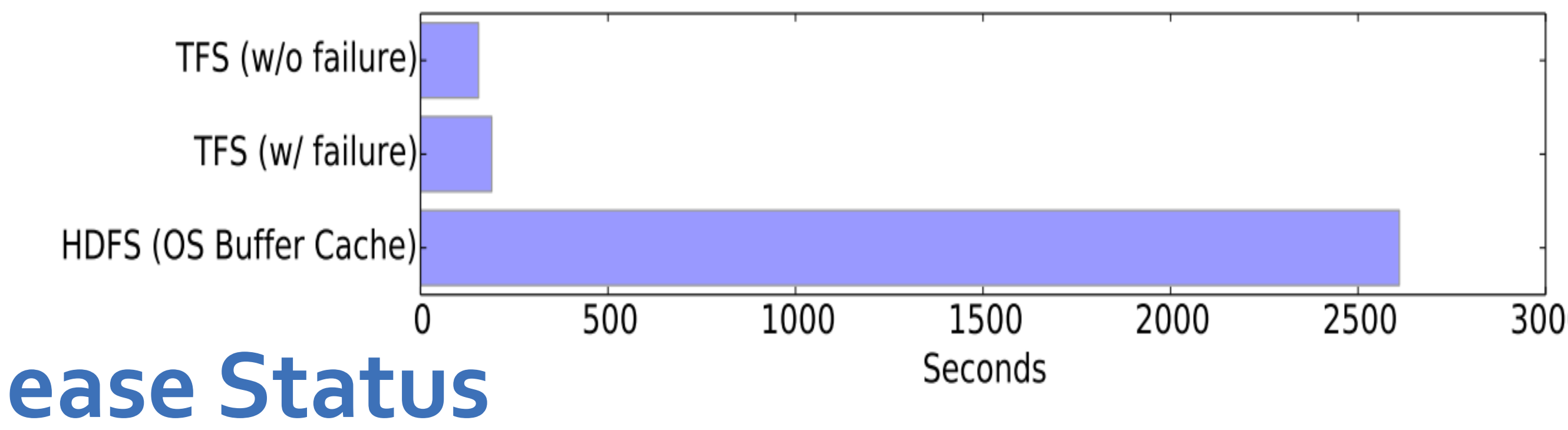
Real Workload



Simple Query

Complex Query

Realistic Workflow



Release Status

- Developer Preview v0.3.0 (October 2013):
  - First read of files cached in memory
  - Write synchronously to UnderlayerFS (no lineage)
  - Spark and MapReduce can use it without changing any code:
- Current Features: Java-like file API; Compatible with Hadoop; Master fault tolerance, Native support for raw table; Pluggable underlayer file system; CLI; Web user interface; Whitelist, Pinlist.

Conclusion

- Tachyon pushes lineage into file system layer to enable memory throughput read and write

Intel Science & Technology Center for Cloud Computing