Strata CONFERENCE + HADOOP WORLD

Oct. 28–30, 2013
NEW YORK, NY

#strataconf + #hw2013

Co-presented by O'REILLY cloudera

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

Haoyuan (HY) Li
UC Berkeley

amplab

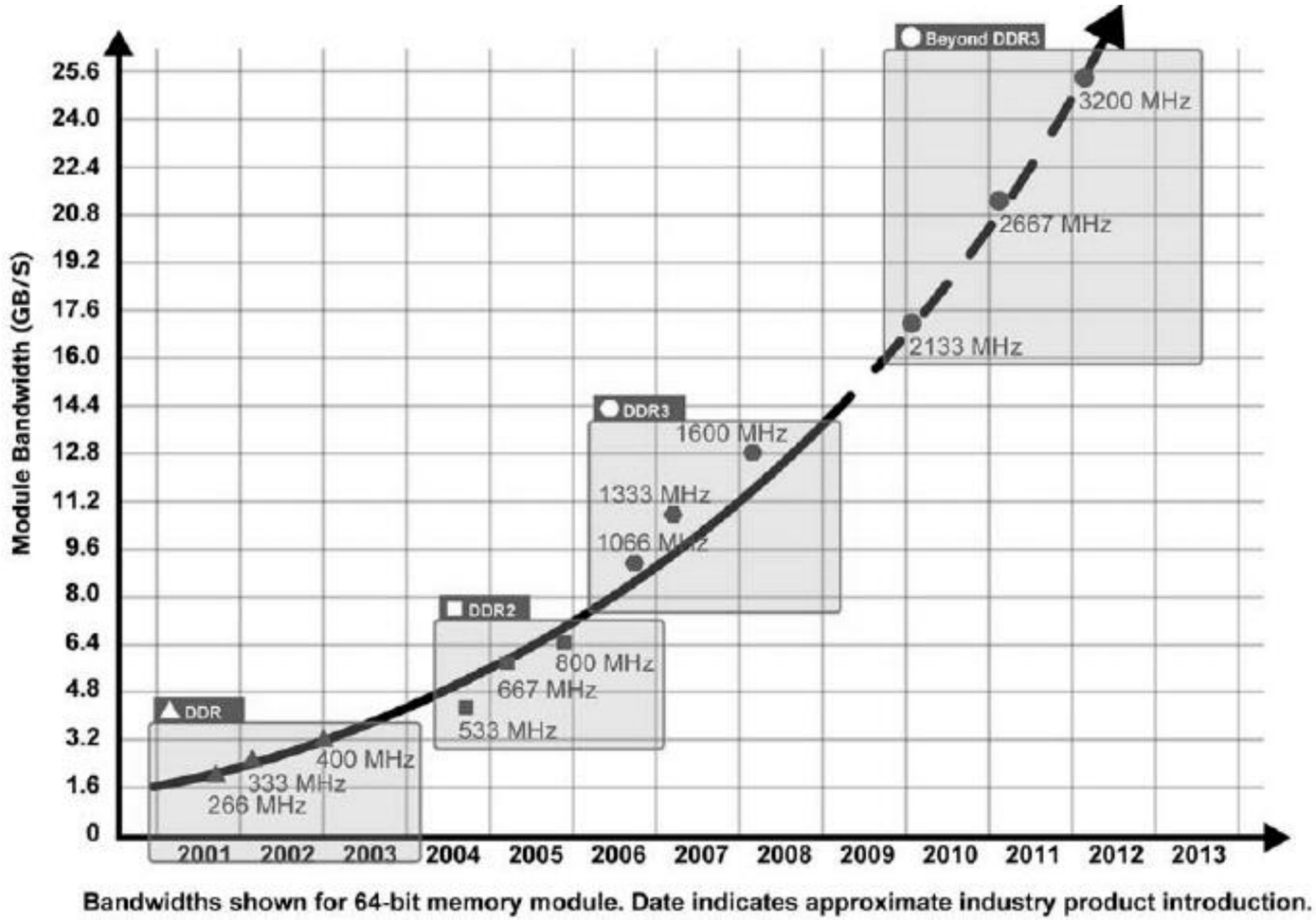
Outline

- Motivation
- System Design
- Evaluation Results
- Release Status

Memory is King

Memory Trend

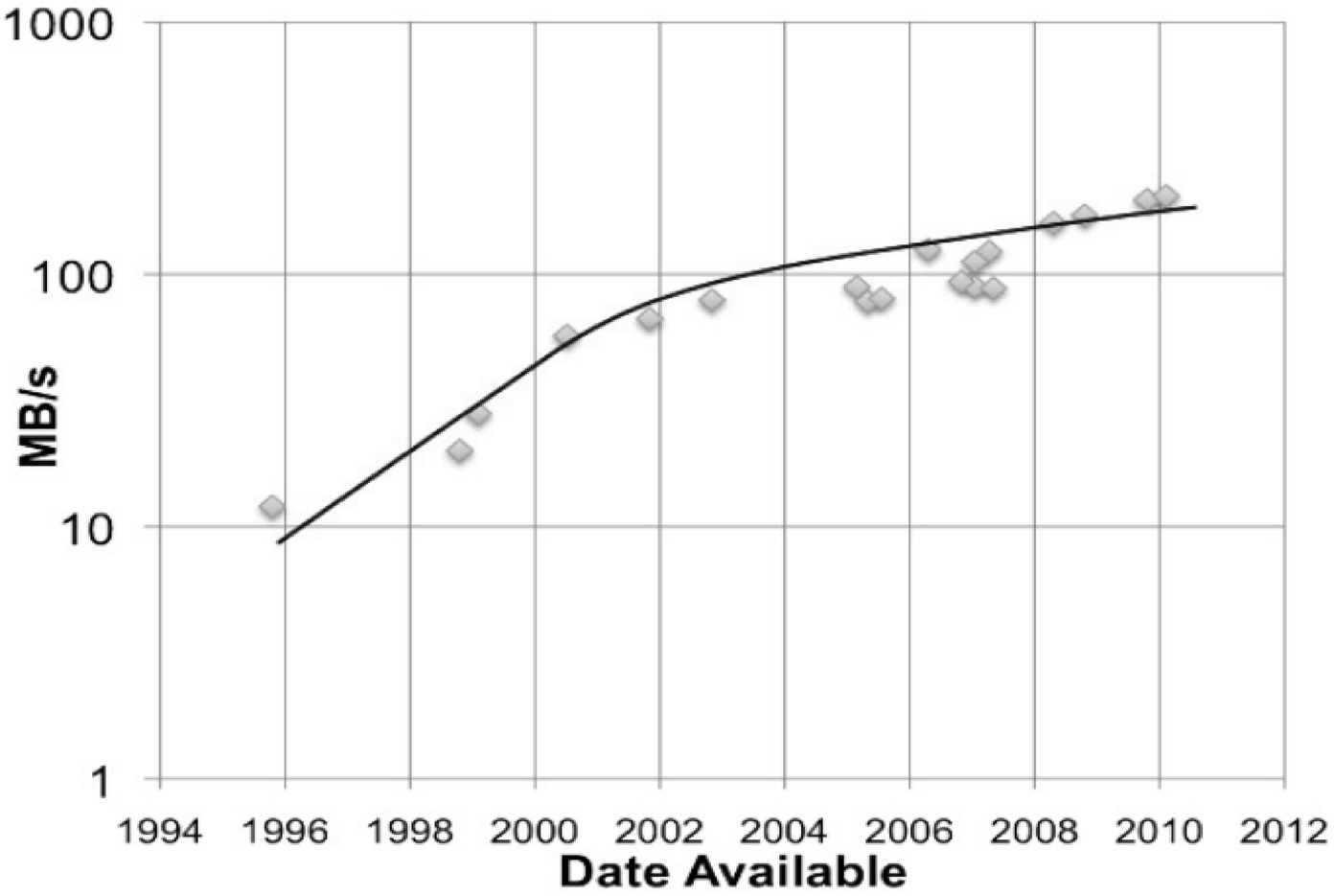
- RAM throughput increasing exponentially



Bandwidths shown for 64-bit memory module. Date indicates approximate industry product introduction.

Disk Trend

- Disk throughput increasing slowly



Consequence

- Memory locality **key** to achieve
  - Interactive queries
  - Fast query response

Current Big Data Eco-system

- Many frameworks already leverage memory
  - e.g. Spark, Shark, and other projects

- File sharing among jobs replicated to disk
  - Replication enables fault-tolerance

- **Problems**
  - Disk scan is slow for read.
  - Synchronous disk replication for write is even slower.

Current Big Data Eco-system

- Many frameworks already leverage memory
  - e.g. Spark, Shark, and other projects

- File sharing among jobs replicated to disk
  - Replication enables fault-tolerance

- **Problems**
  - Disk scan is slow for read.
  - Synchronous disk replication for write is even slower.

Tachyon Project

- Reliable file sharing at memory-speed across cluster frameworks/jobs

- Challenge
  - How to achieve reliable file sharing without replication?

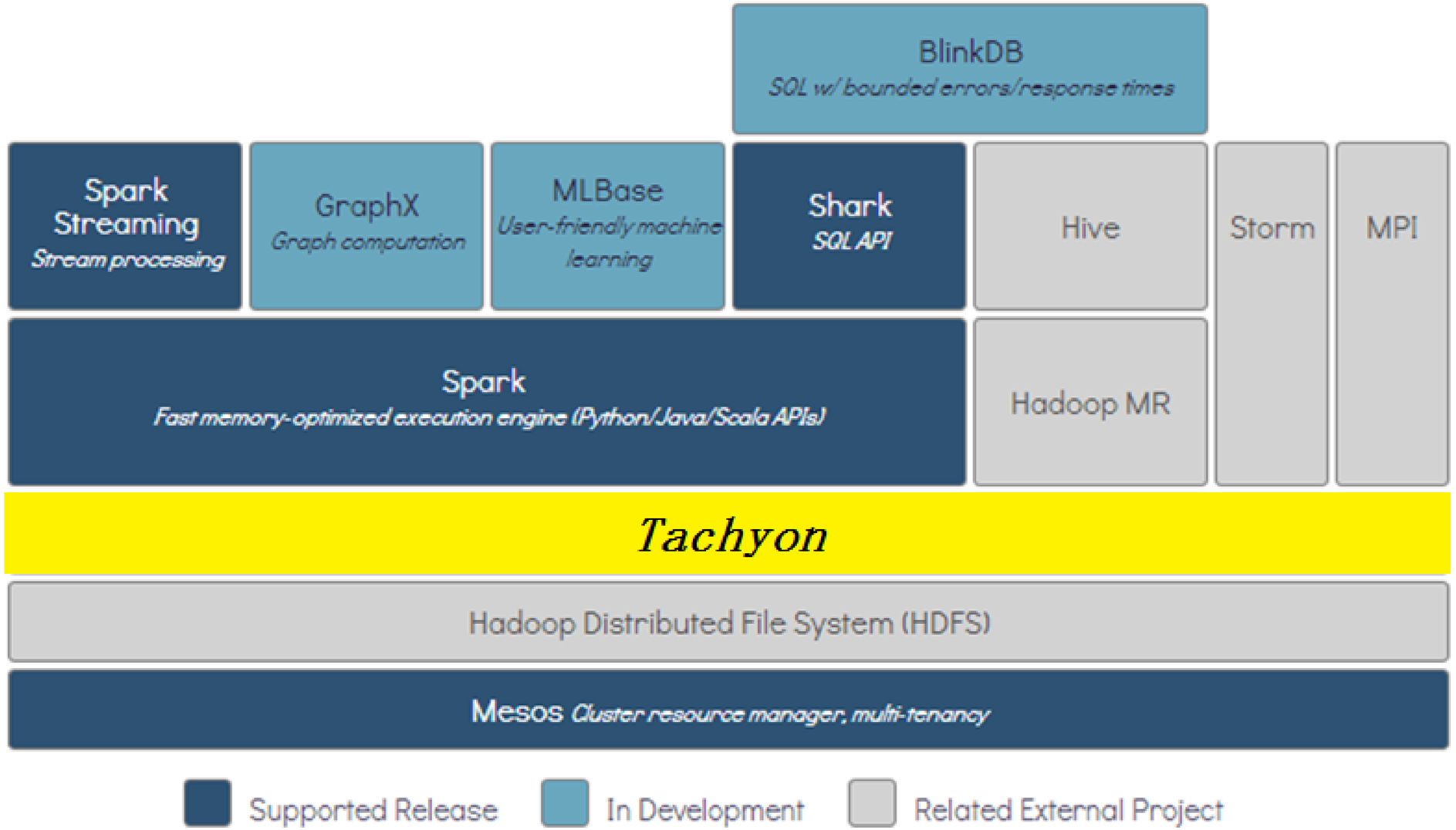
Idea

Re-computation (Lineage) based storage using memory aggressively.

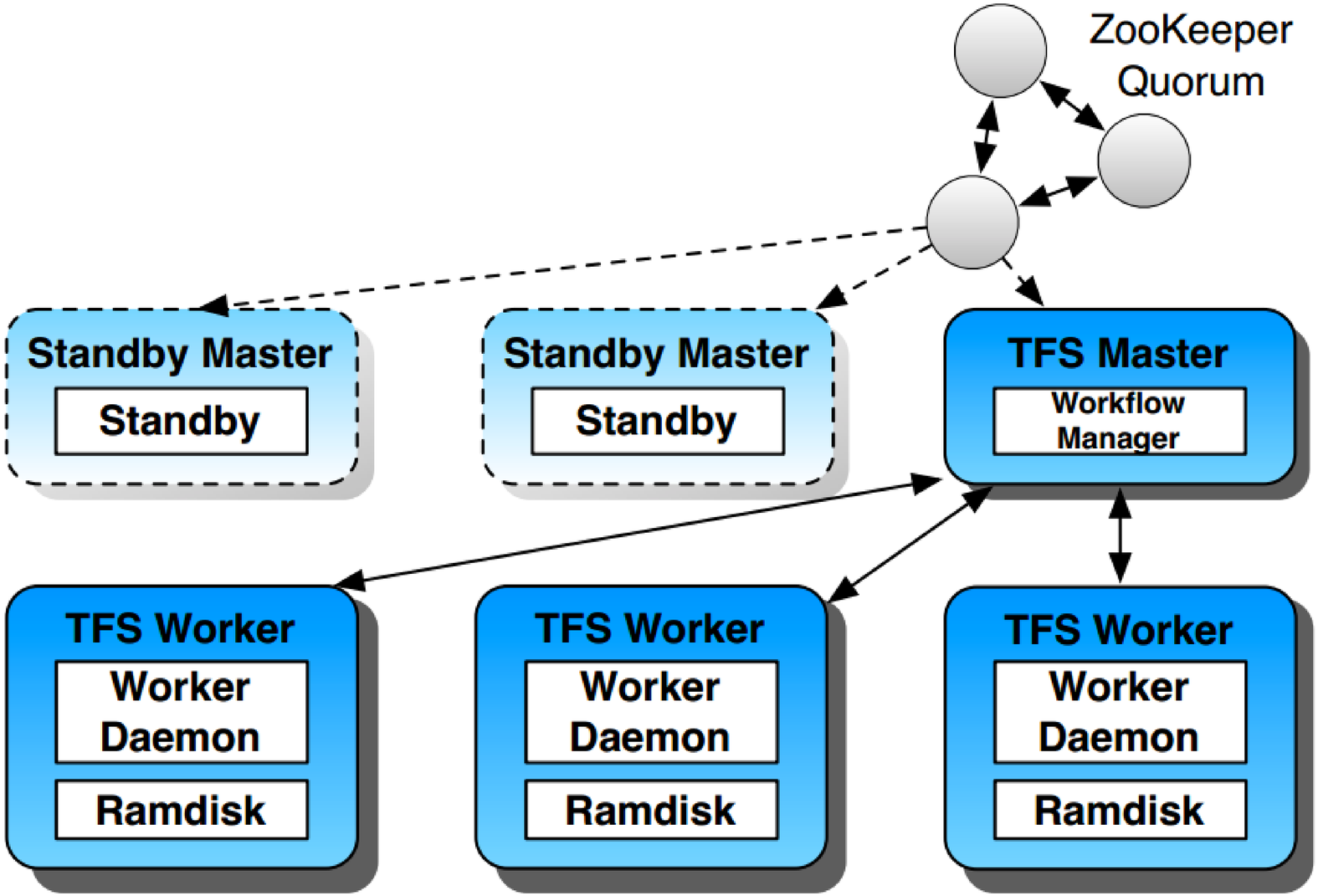
1. One copy of data in memory (Fast)

2. Upon failure, re-compute data using ***lineage*** (Fault tolerant)

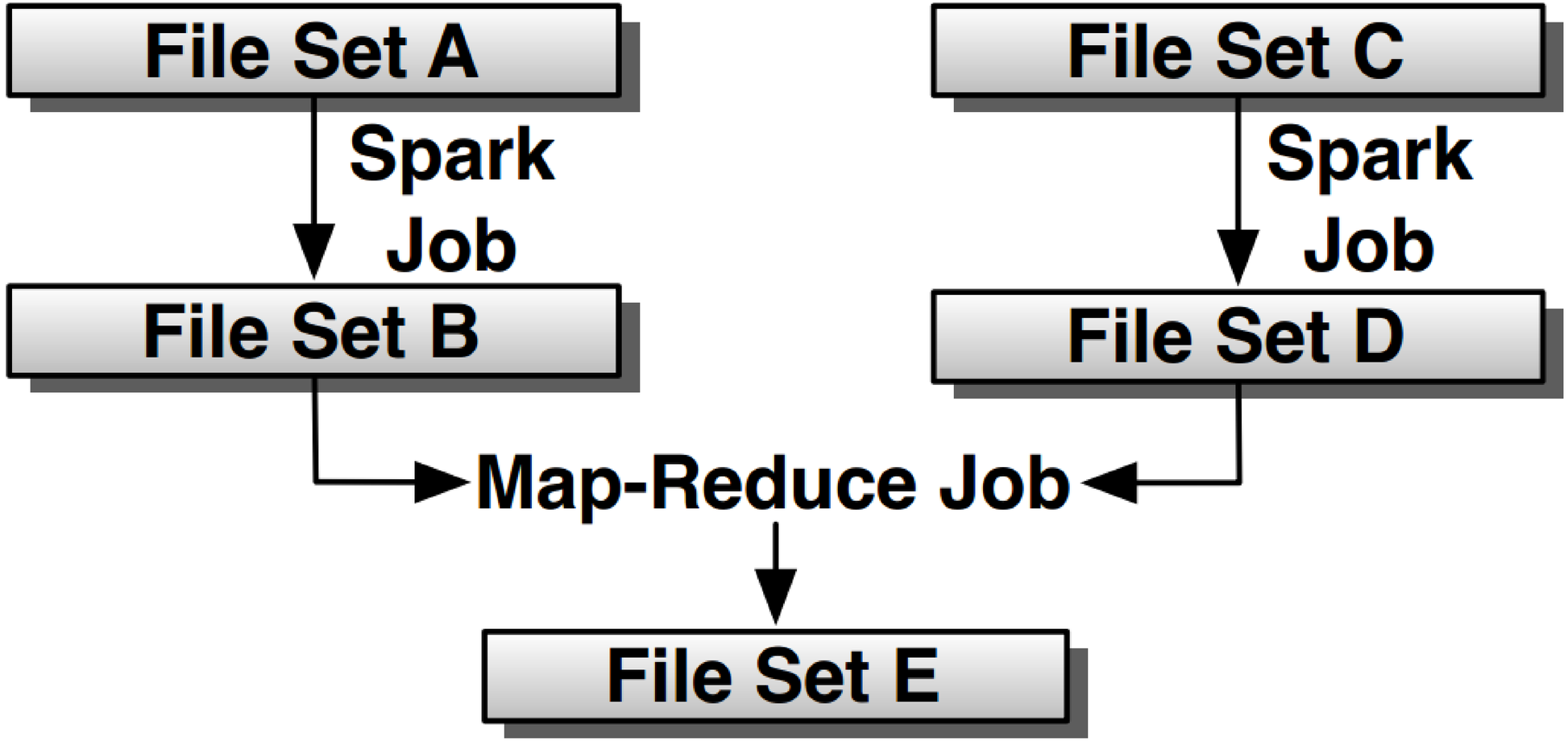
Stack


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

Architecture



Lineage


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

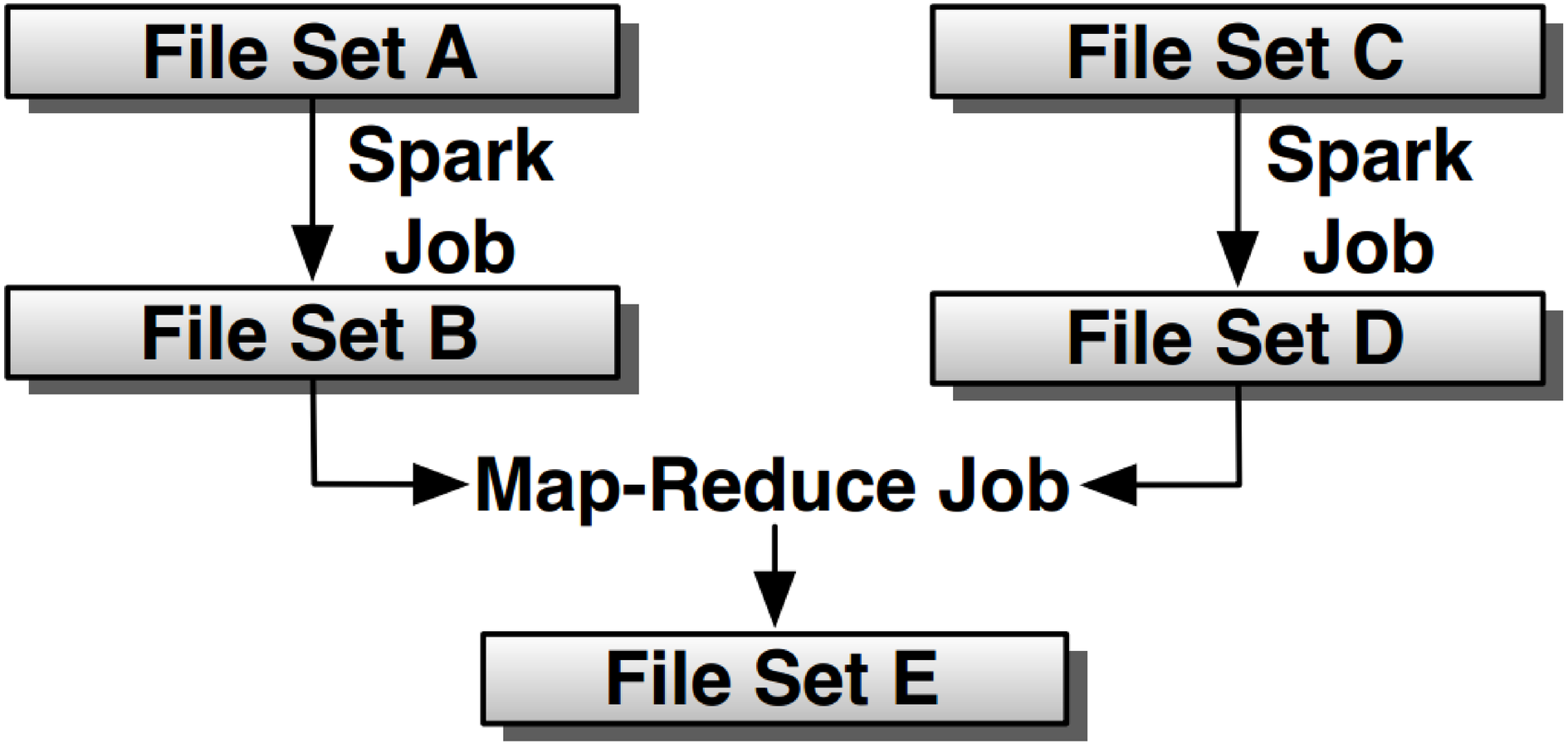
Lineage Information

- Binary program
- Configuration
- Input Files List
- Output Files List
- Dependency Type

Fault Recovery

Re-computation Cost?

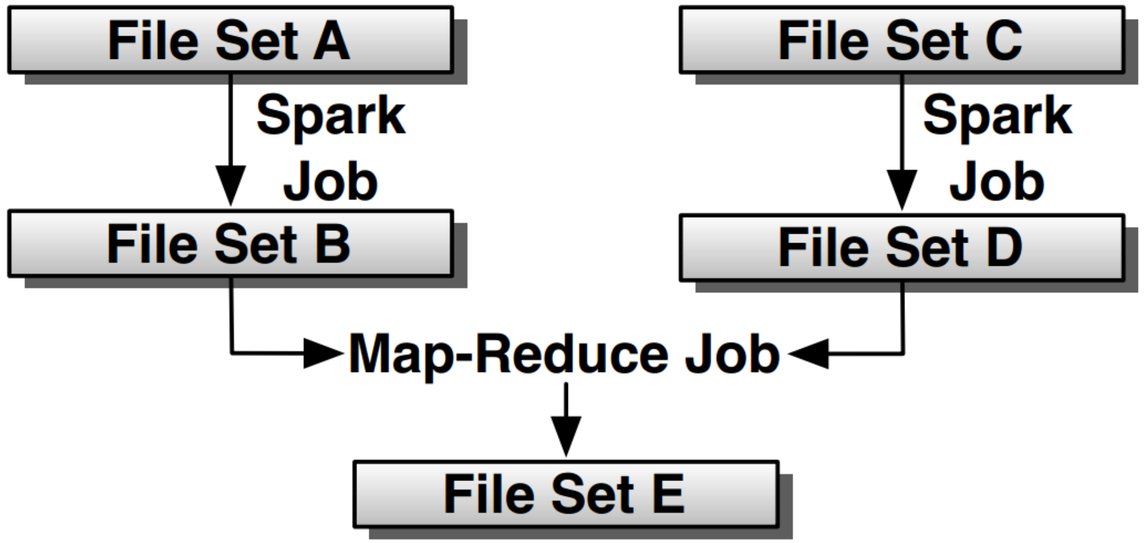
Example


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

Asynchronous Checkpoint

- Better than using existing solutions even under failure.

- Bounded recovery cost (Snapshot algorithm).



Master Fault Tolerance

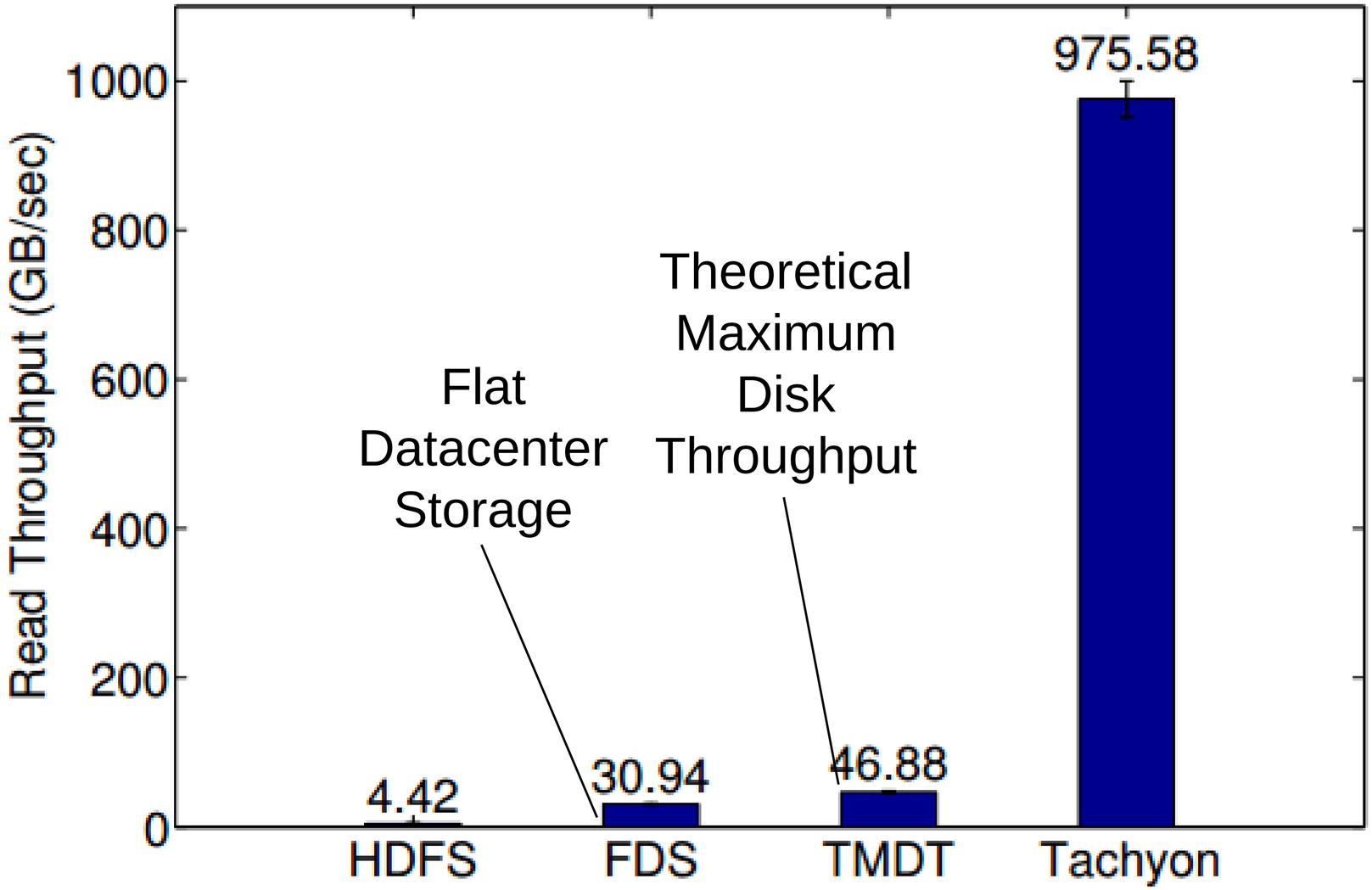
- Multiple masters
  - Use ZooKeeper to elect a leader

- After crash workers contact new leader
  - Update the state of leader with contents in memory

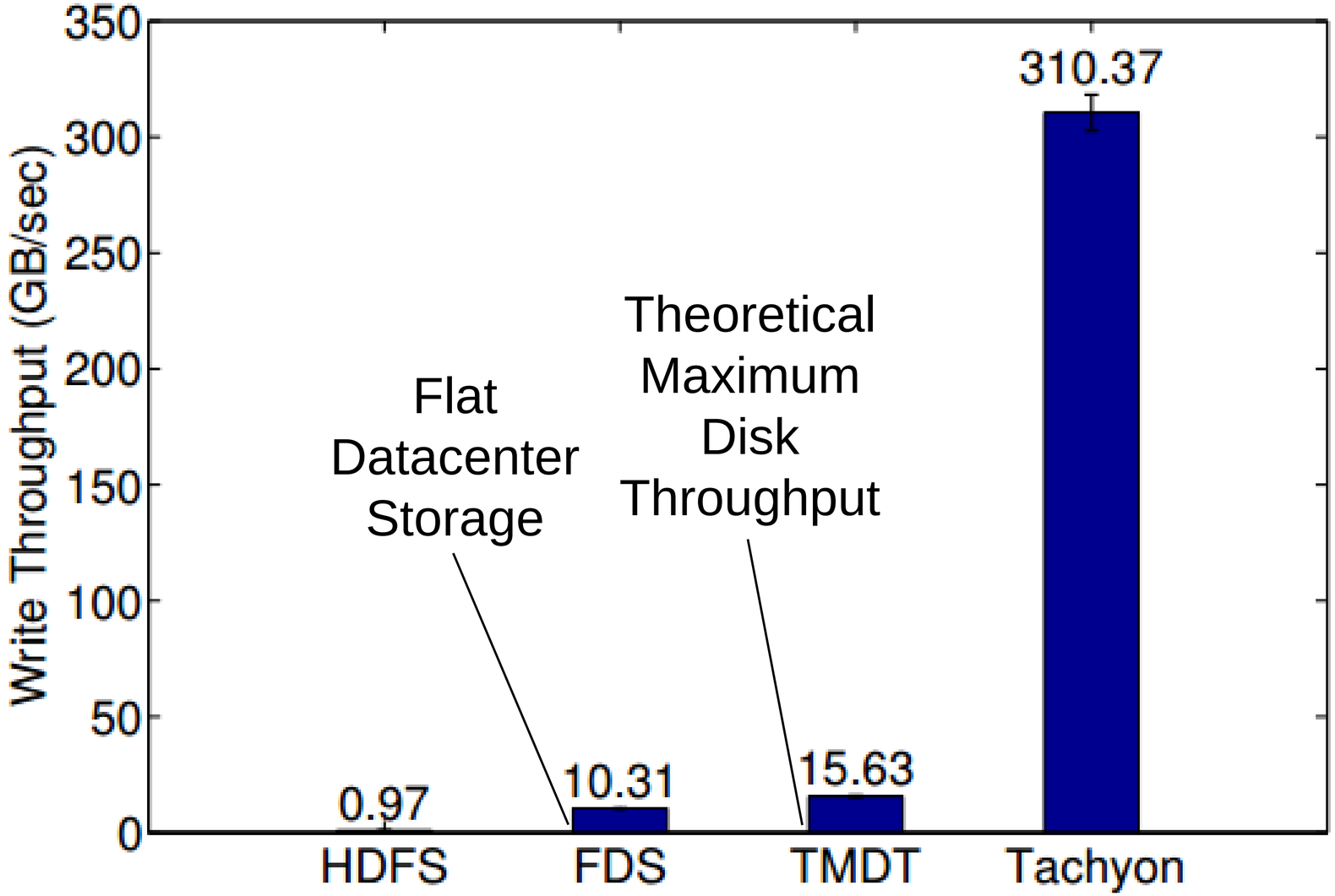
Implementation

- Java
- Thrift
- HDFS, S3, localFS

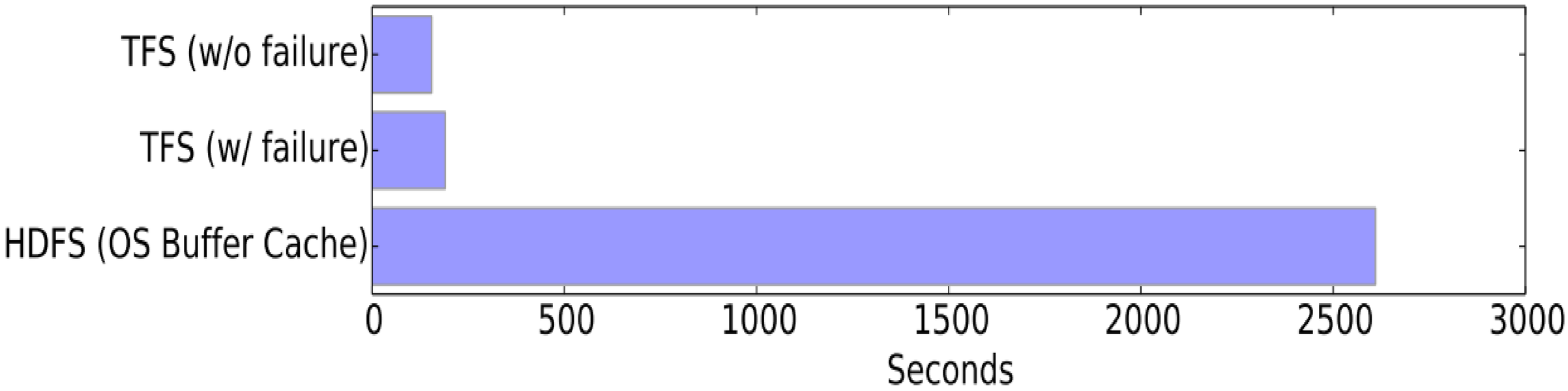
Sequential Read using Spark



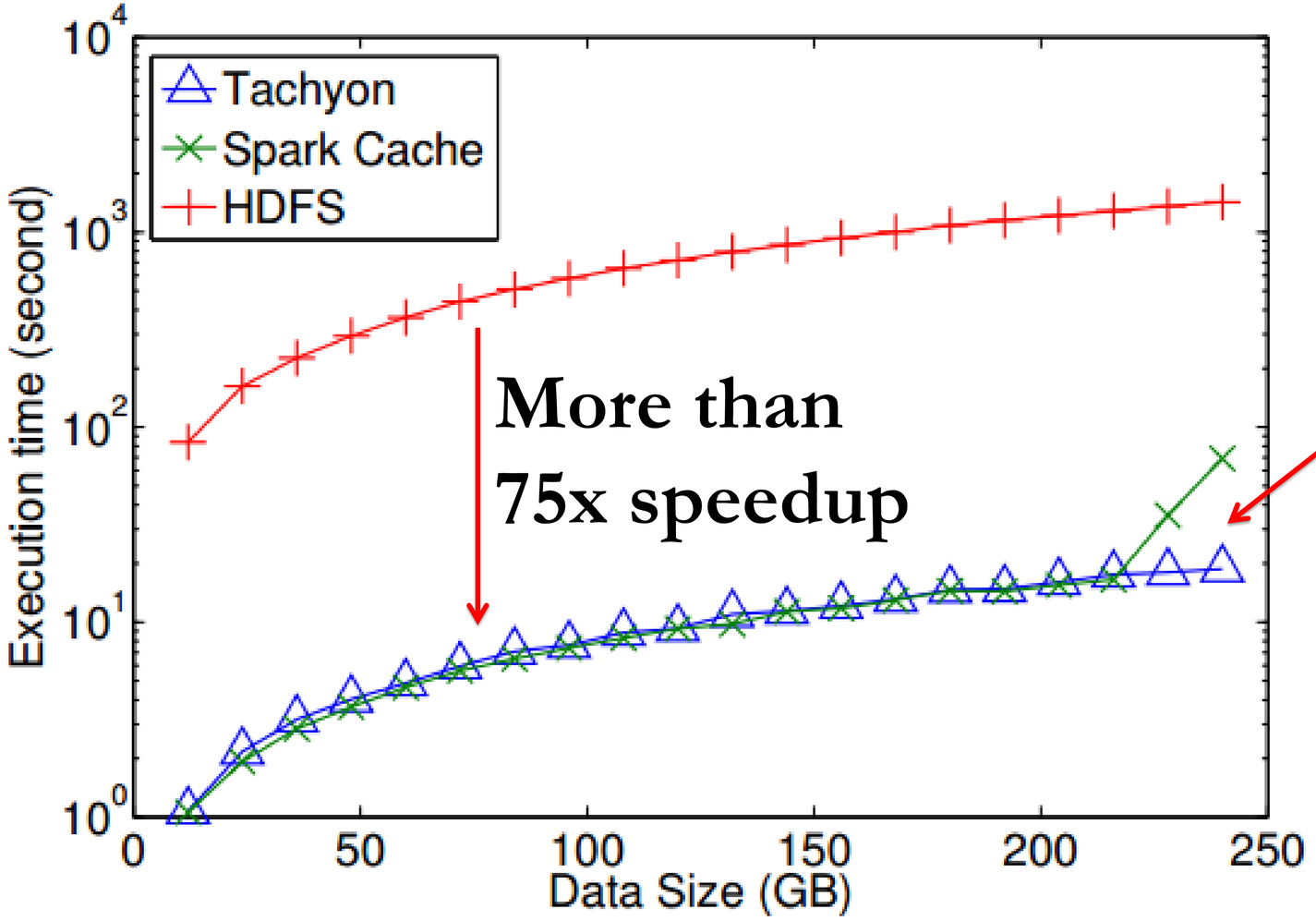
Sequential Write using Spark



Realistic Workflow using Spark

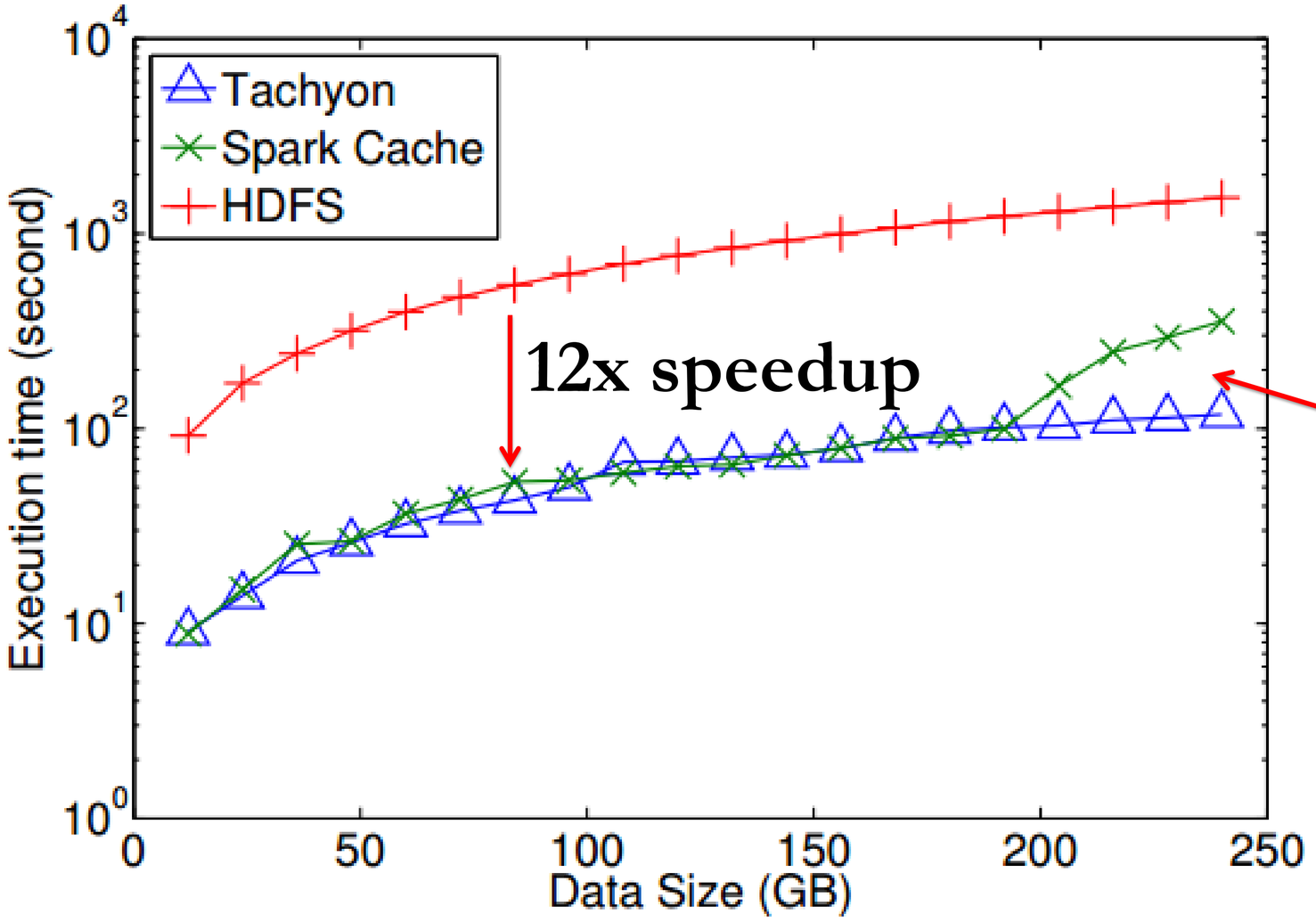


Conviva Spark Query (I/O intensive)



Tachyon outperforms Spark cache because of Java Garbage Collection

Conviva Spark Query (less I/O intensive)



Garbage collection kicks in earlier for Spark cache

Alpha Status

- Developer Preview: V0.3.0 (October 2013)
  - First read of files cached in-memory
  - Writes go synchronously to HDFS (No lineage information in Developer Preview release)
  - MapReduce and Spark can run without any code change (ser/de becomes the new bottleneck)

Current Features

- Java-like file API
- Compatible with Hadoop
- Master fault tolerance
- Native support for raw tables
- WhiteList, PinList
- CLI, Web UI

Spark without Tachyon

```
val file = sparkcontext.textFile(“hdfs://ip:port/path”)
```

Spark with Tachyon

```
val file = sparkcontext.textFile(“tachyon:// ip:port/path”)
```

Shark without Tachyon

```
CREATE TABLE orders_cached AS SELECT * FROM
orders;
```

Shark with Tachyon

```
CREATE TABLE orders_tachyon AS SELECT * FROM
orders;
```

More Experiments with Shark

- Shark (from 0.7) can store tables in Tachyon with fast columnar Ser/De

20 GB data / 5 machines	Spark Cache	Tachyon
Table Full Scan	1.4 sec	1.5 sec
GroupBys (10 GB Shark Memory)	70 sec	47.5 sec
GroupBys (15 GB Shark Memory)	46 sec	41 sec

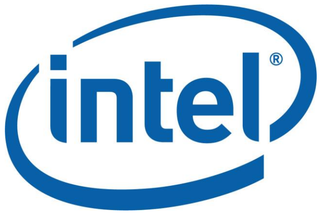
4 * 100 GB TPC-H data / 17 machines	Spark Cache	Tachyon
TPC-H Q1	65.68 sec	24.75 sec
TPC-H Q2	438.49 sec	139.25 sec
TPC-H Q3	467.79 sec	55.99 sec
TPC-H Q4	457.50 sec	111.65 sec

Future

- Efficient Ser/De support
- Fair sharing for memory
- Full support for lineage

Acknowledgment

Ali Ghodsi, Matei Zaharia, Scott Shenker, Ion Stoica , Eric Baldeschwieler, Calvin Jia, Bill Zhao, Nick Lanham, Mark Hamstra, Rong Gu, Hobin Yoon, Vamsi Chitters, Joseph Jin-Chuan Tang, Grace Huang, Reynold Xin, Achal Soni, Dilip Joseph, Srinivas Parayya



Summary

- High-throughput, fault-tolerant in-memory storage
- Interface compatible to HDFS
- Further improve performance for Spark, Shark, and Hadoop
- Growing community with 8 companies contributing since its first public release six months ago
- More information: www.tachyonproject.org