Tachyon: A Reliable Memory Centric Storage for Big Data Analytics

Haoyuan (HY) Li, Ali Ghodsi, Matei Zaharia, Scott Shenker, Ion Stoica

TACHYON

Outline

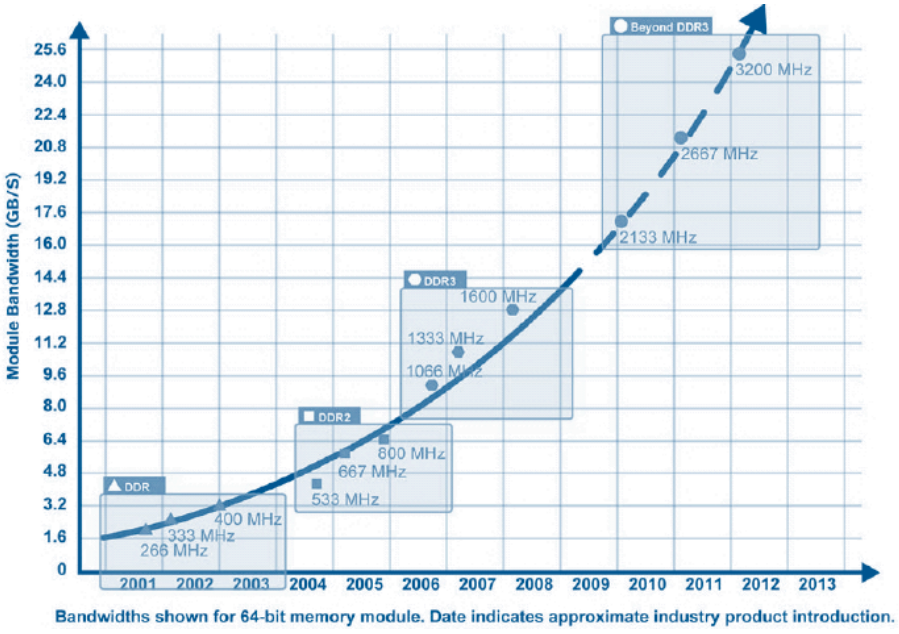
- Overview
- Research
- Open Source
- Future

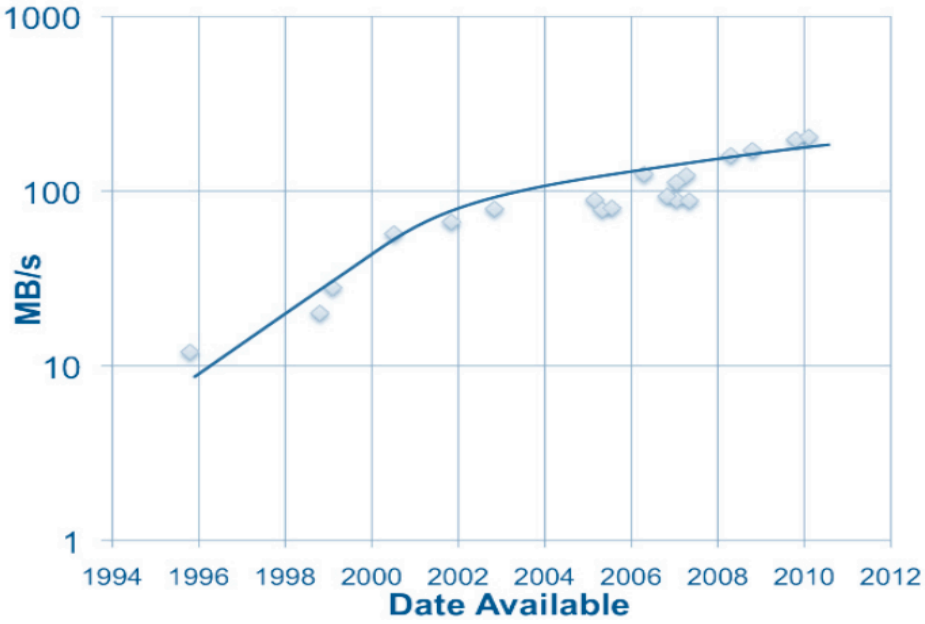
Outline

- **Overview**
- Research
- Open Source
- Future

Memory is King

- RAM throughput increasing exponentially
- Disk throughput increasing slowly

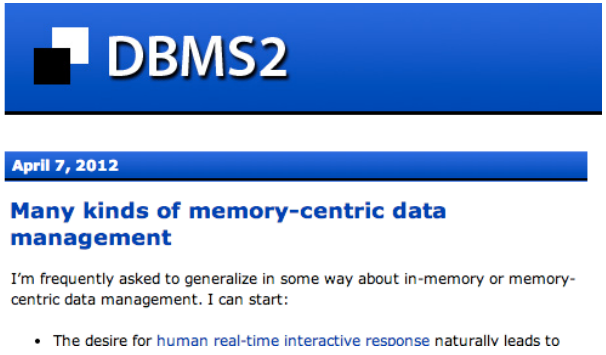




Memory-locality key to interactive response time

Realized by many…

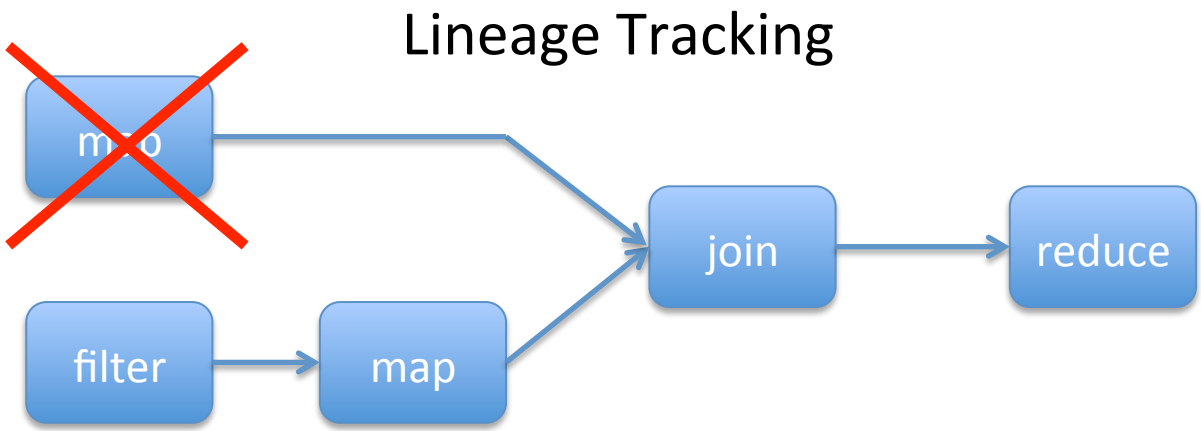
- Frameworks already leverage memory



Problem solved?

An Example: Spark

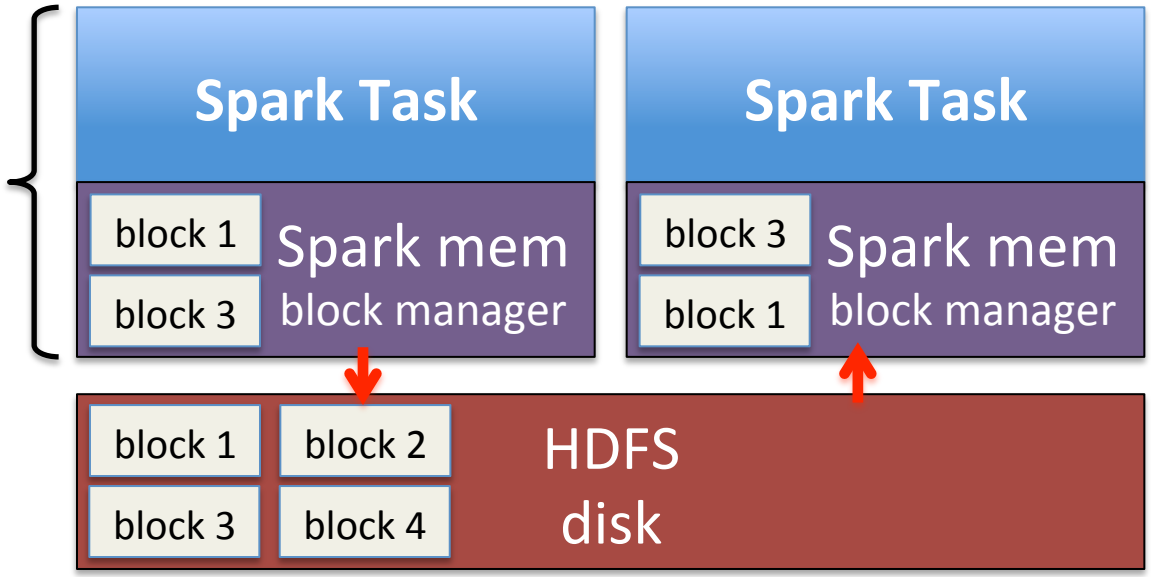
- Fast in-memory data processing framework
  - Keep one in-memory copy inside JVM
  - Track lineage of operations used to derive data
  - Upon failure, use lineage to recompute data



Issue 1

Different jobs share data:
Slow writes to disk

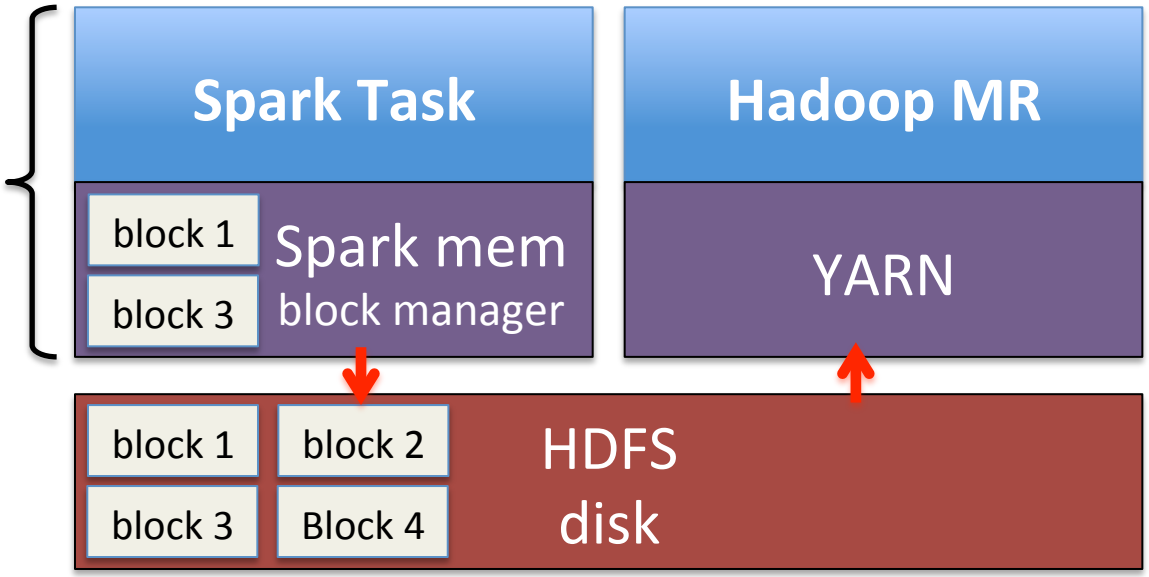
storage engine &
execution engine
same process
(slow writes)



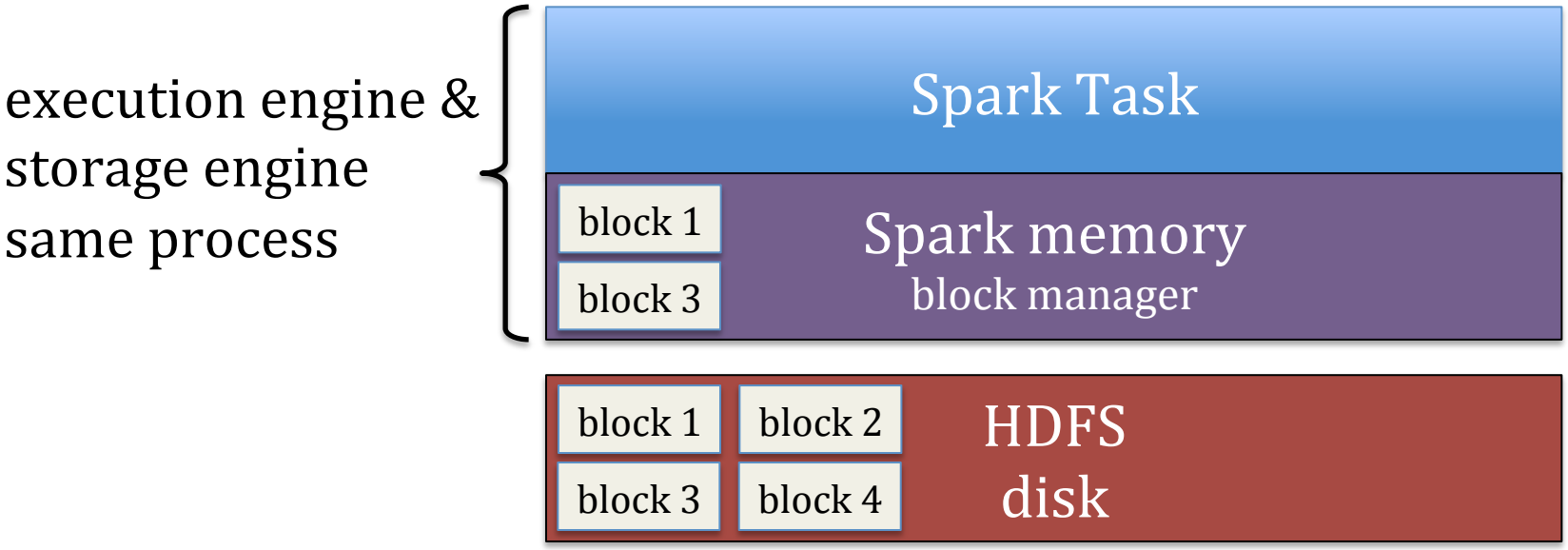
Issue 1

Different frameworks share data: Slow writes to disk

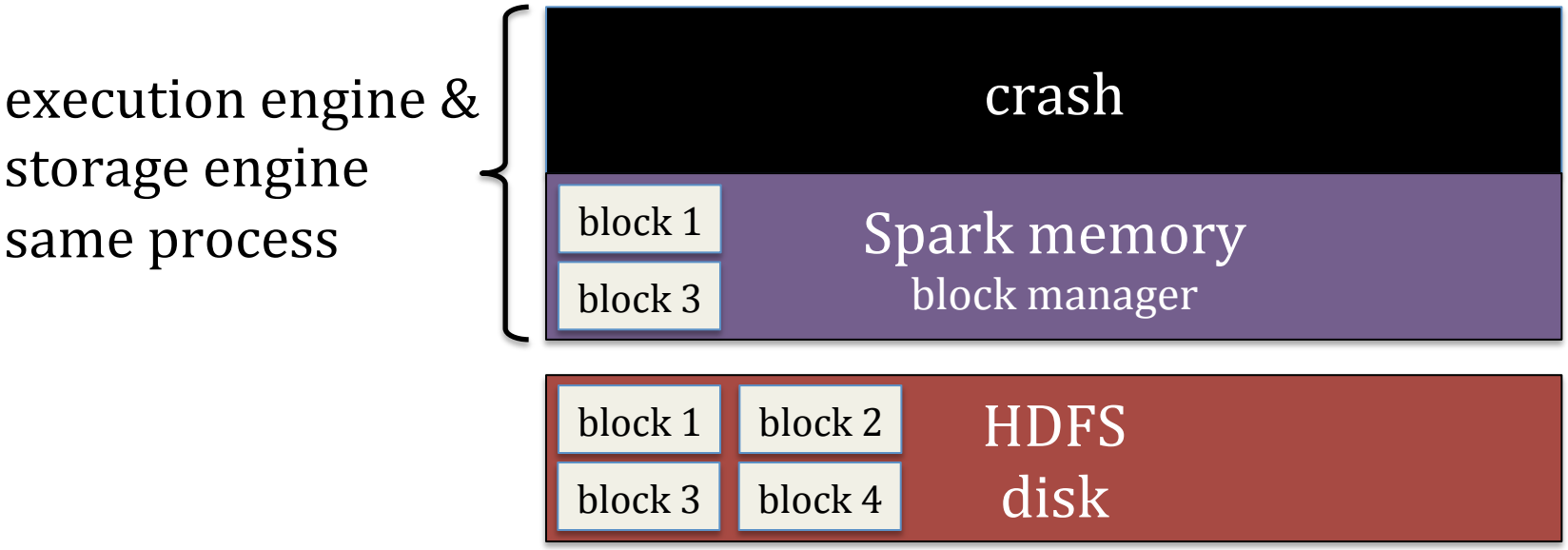
storage engine & execution engine same process (slow writes)



Issue 2

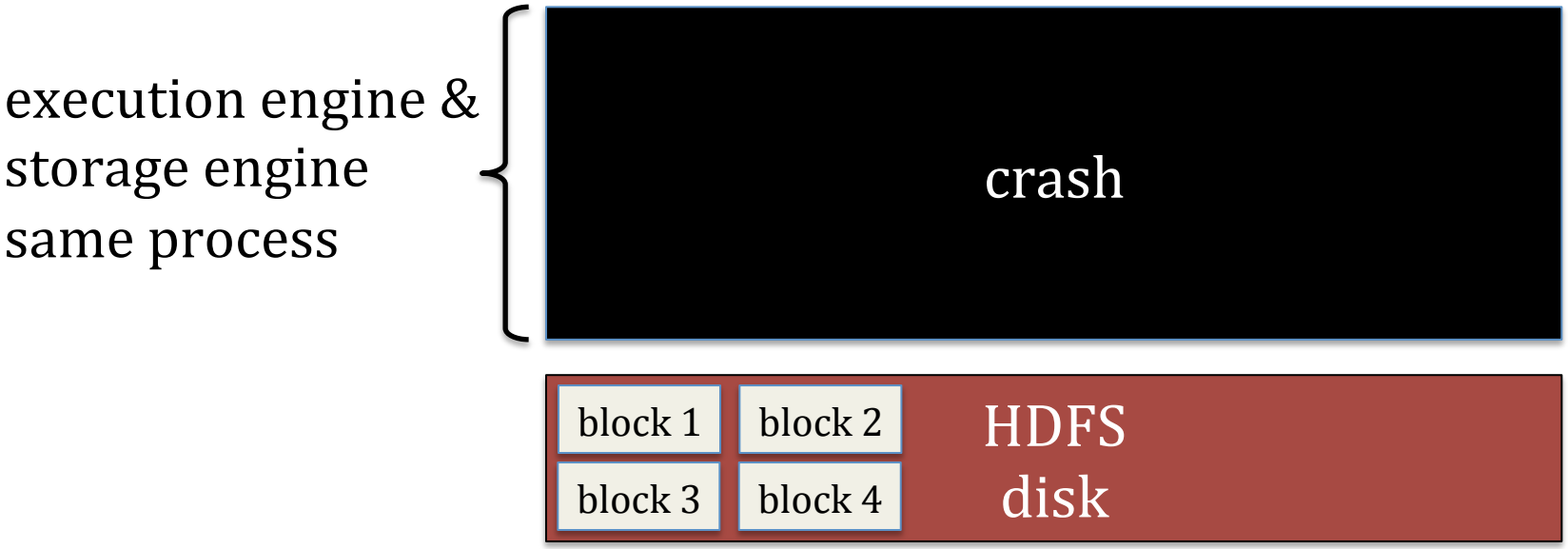

execution engine &
storage engine
same process
Spark Task
block 1
block 3
Spark memory
block manager
block 1
block 2
block 3
block 4
HDFS
disk

Issue 2


execution engine &
storage engine
same process
crash
block 1
block 3
Spark memory
block manager
block 1
block 2
block 3
block 4
HDFS
disk

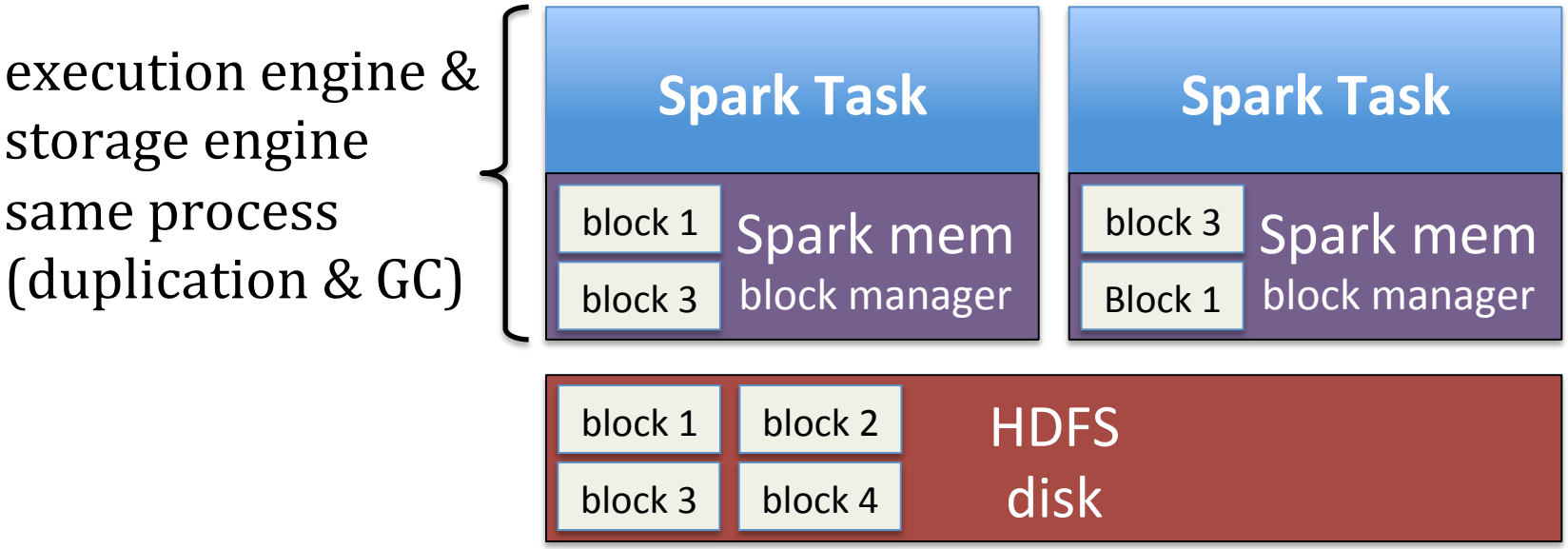
Issue 2

Process crash: lose all cache



Issue 3

duplicate memory per job & GC



Tachyon

Reliable data sharing at ***memory-speed within and across*** cluster frameworks/jobs

Solution Overview

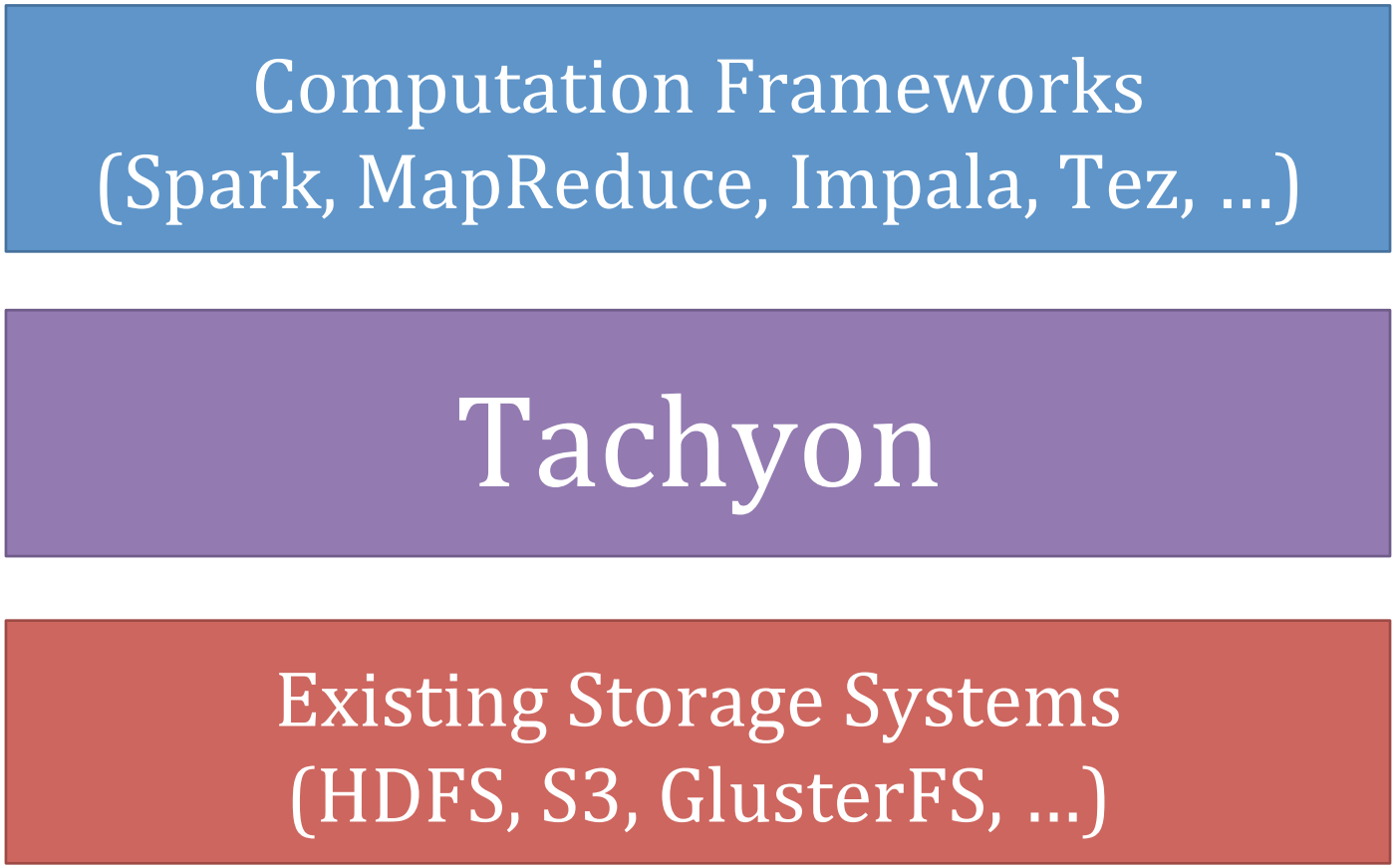
Basic idea

- Push lineage down to storage layer
- Use memory aggressively

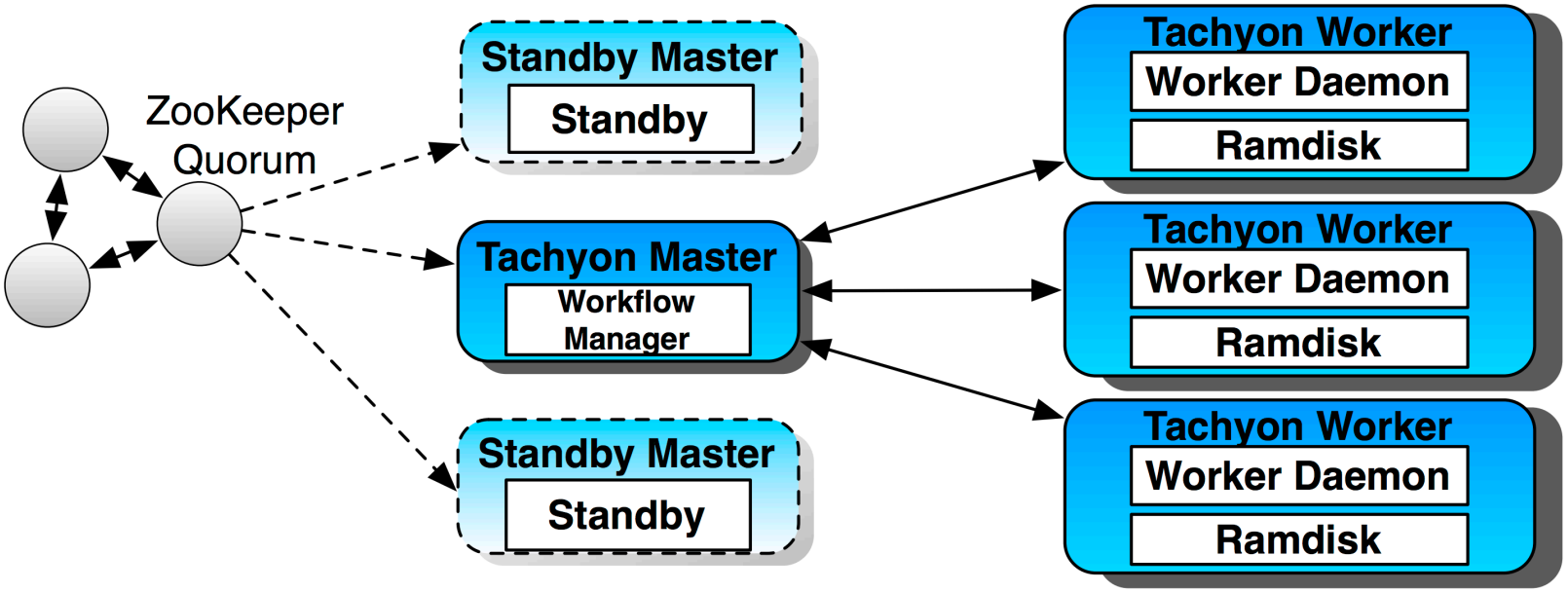
Facts

- One data copy in memory
- Rely on recomputation for fault-tolerance

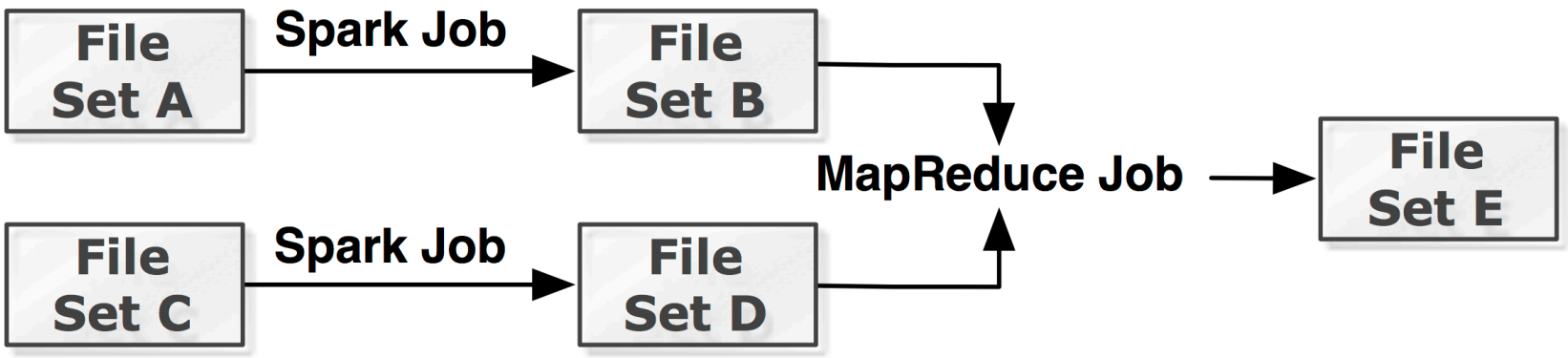
Stack


Computation Frameworks
(Spark, MapReduce, Impala, Tez, …)
Tachyon
Existing Storage Systems
(HDFS, S3, GlusterFS, …)

Architecture

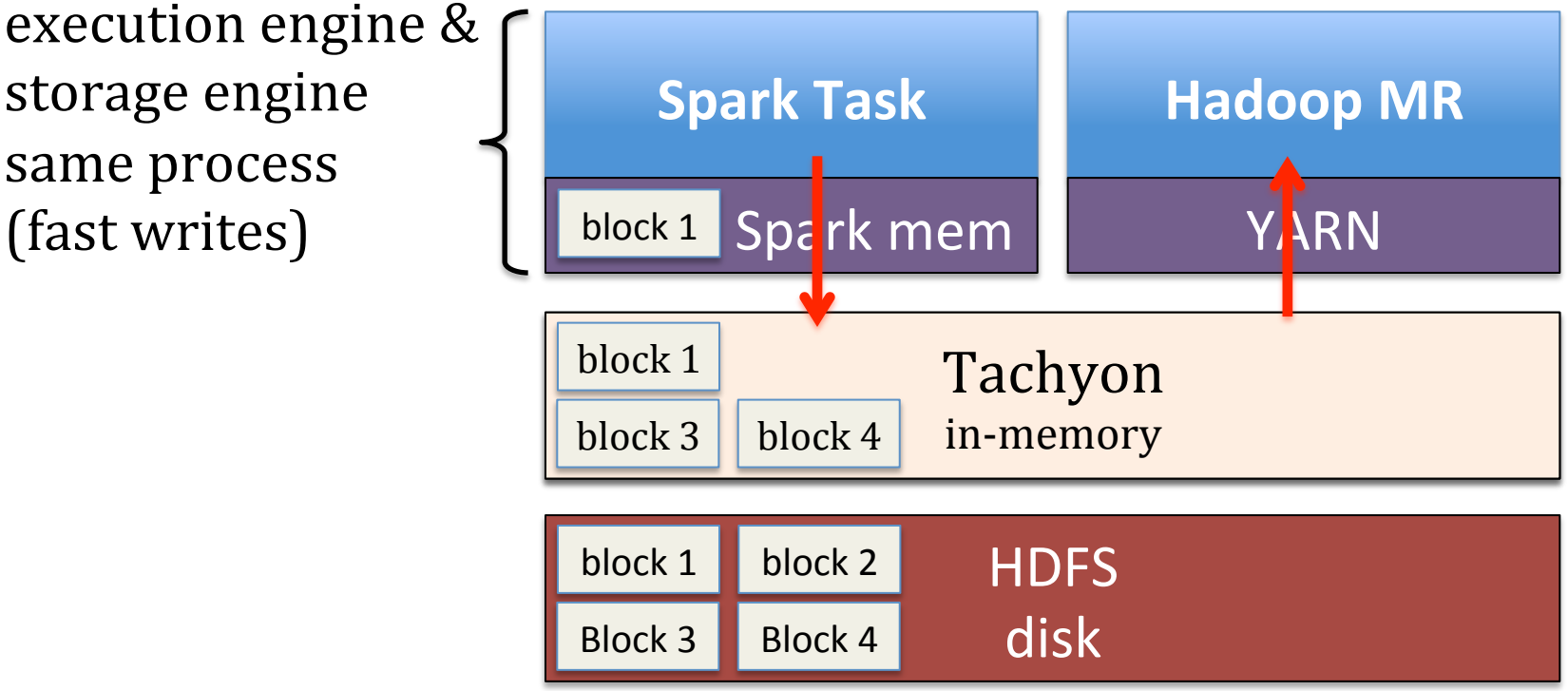


Lineage

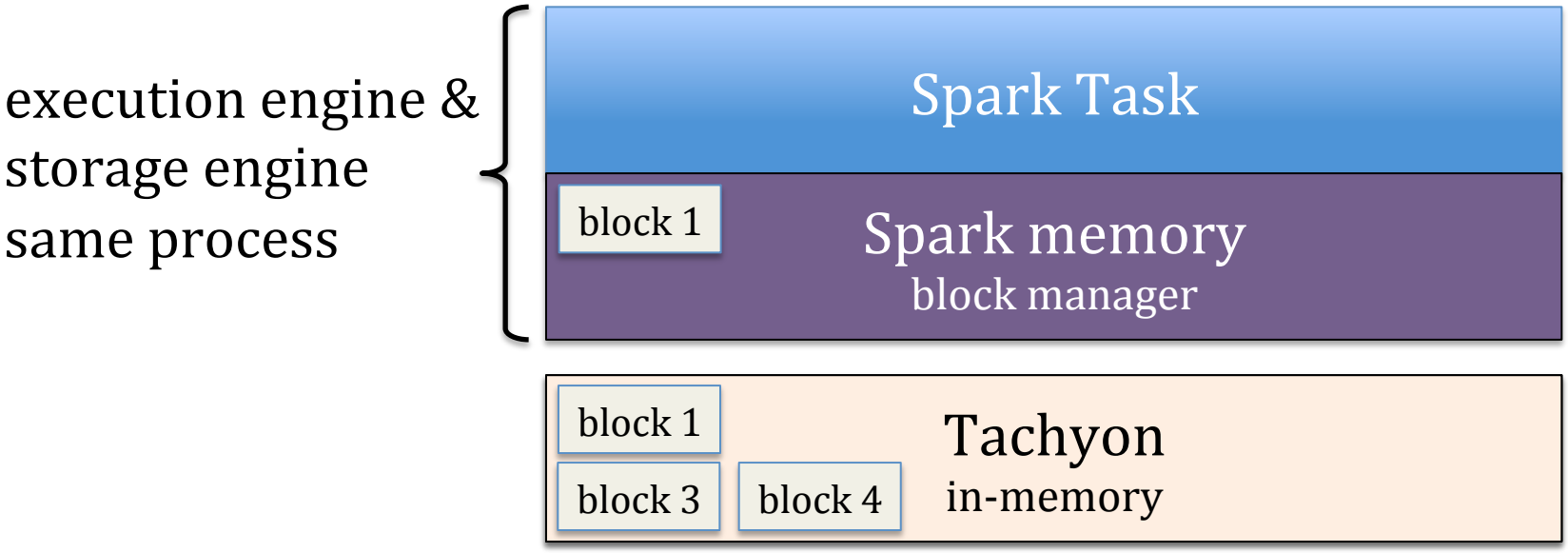

File Set A
Spark Job
File Set B
File Set C
Spark Job
File Set D
MapReduce Job
File Set E

Issue 1 revisited

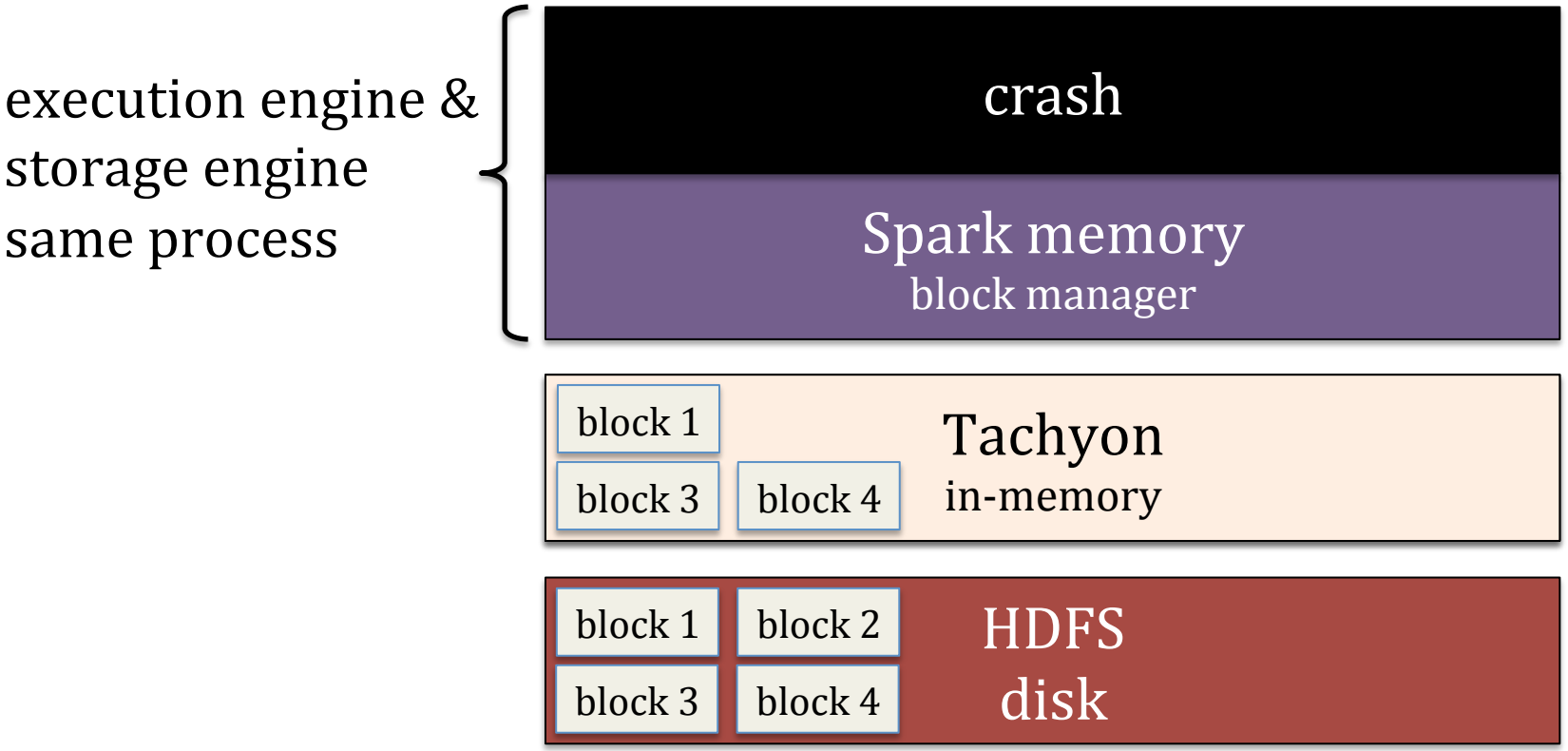
Different frameworks share at memory-speed



Issue 2 revisited

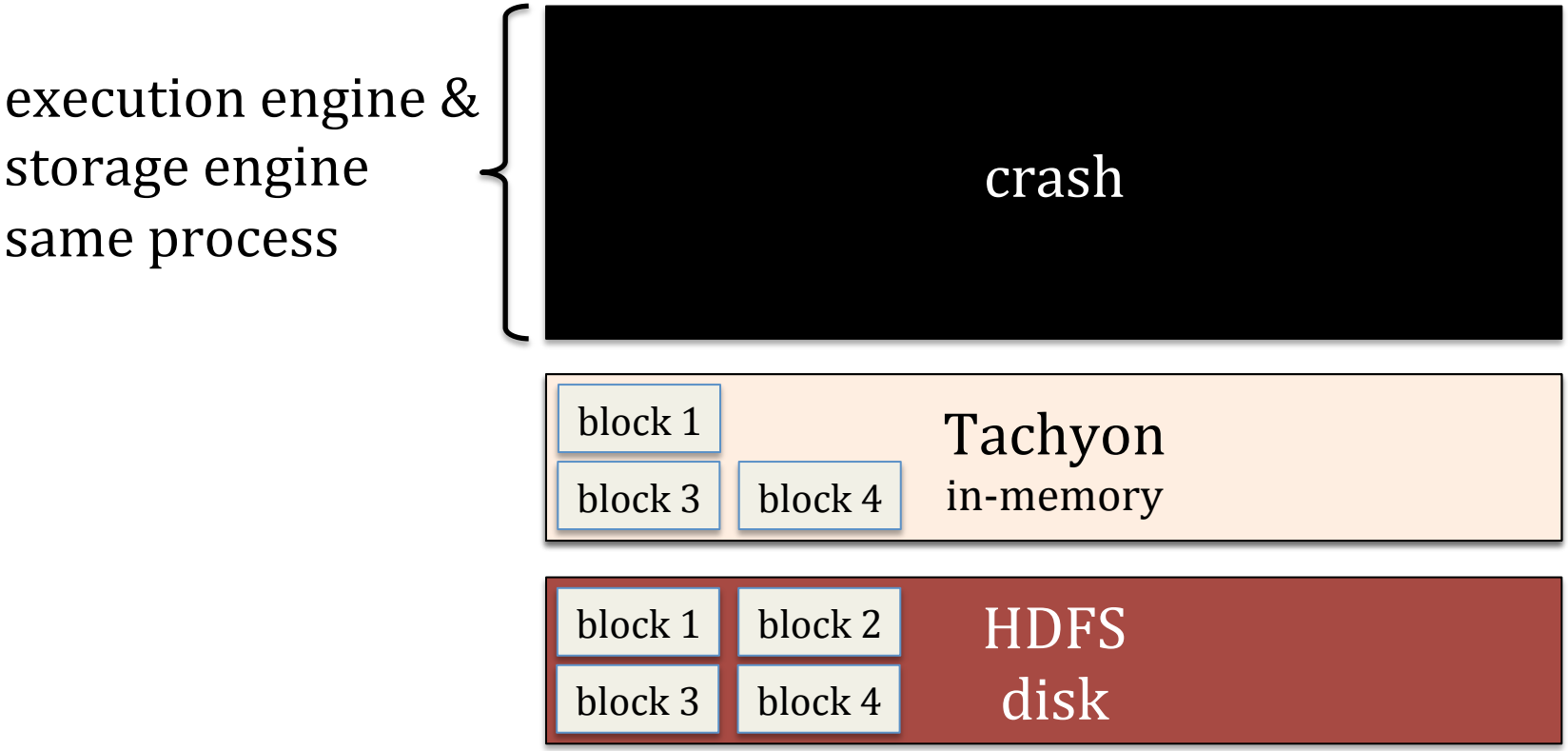


Issue 2 revisited



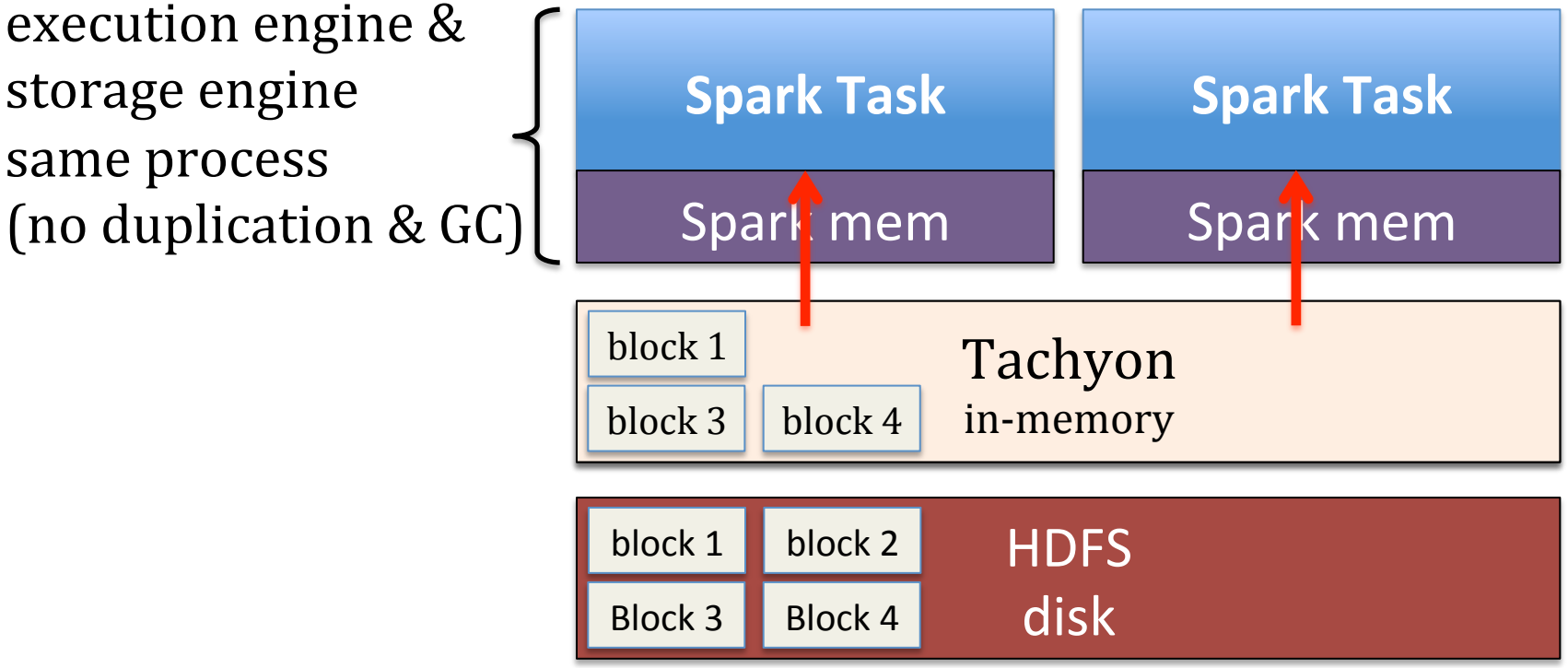
Issue 2 revisited

process crash: keep memory-cache



Issue 3 revisited

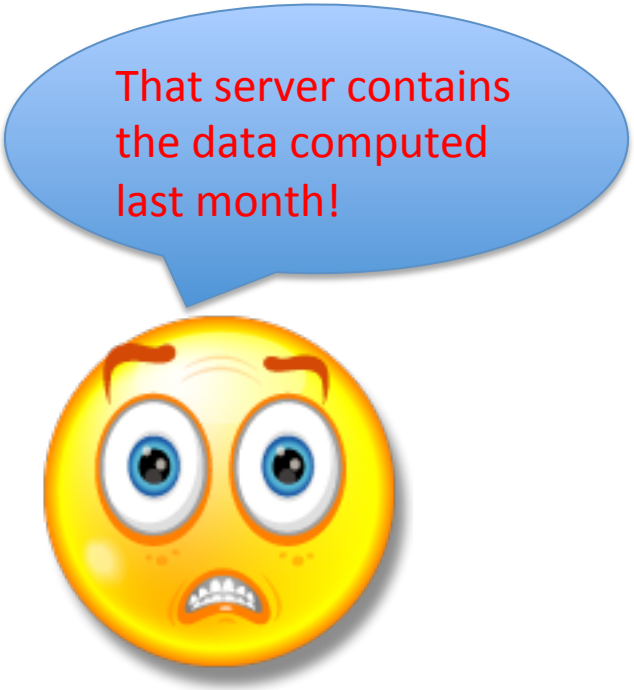
Off-heap memory storage
one memory copy & no GC



Outline

- Overview
- **Research**
- Open Source
- Future

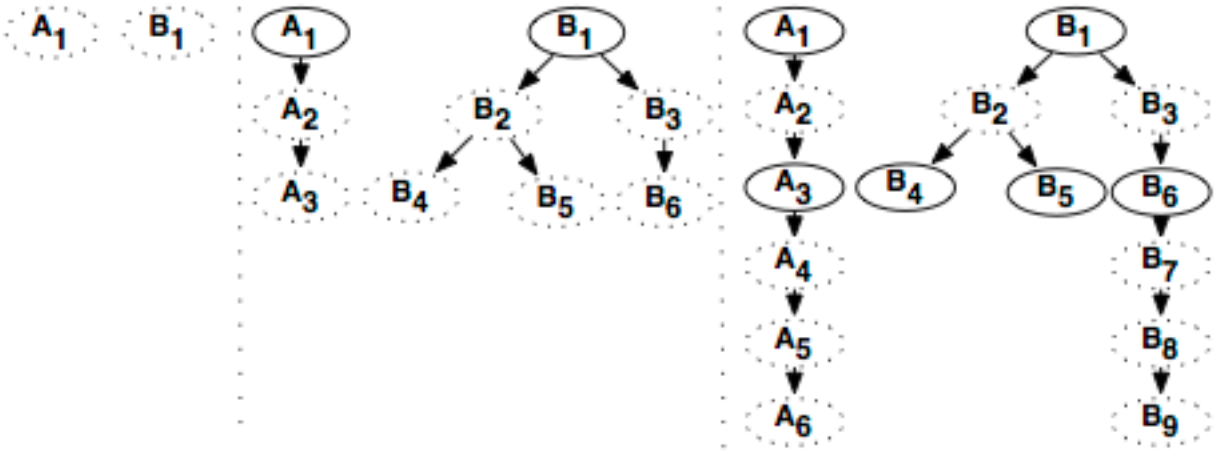
Question 1: How long to get missing data back?



Lineage enables Asynchronous Checkpointing

Edge Algorithm

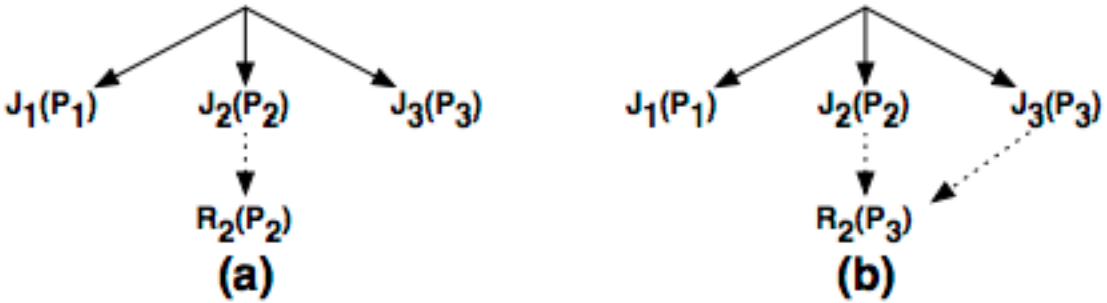
- Checkpoint leaves
- Checkpoint hot files
- Bounded Recovery Cost



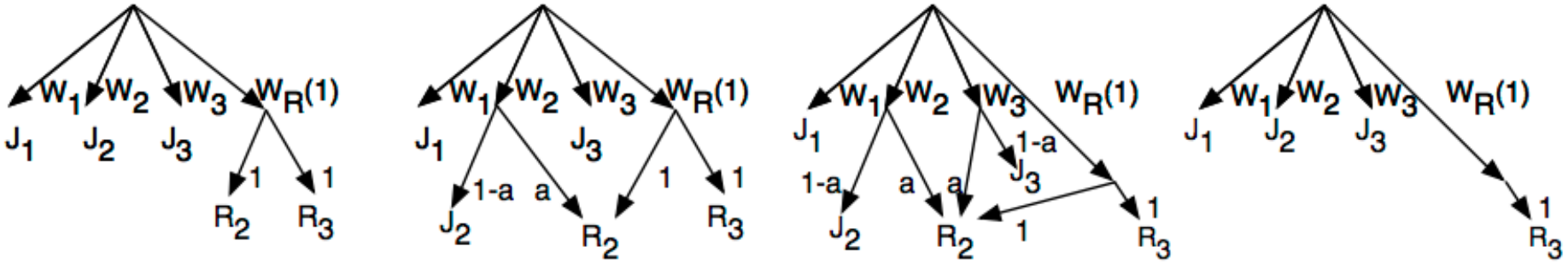
Question 2: How to allocate recomputation resource?

Recomputation Resource Allocation

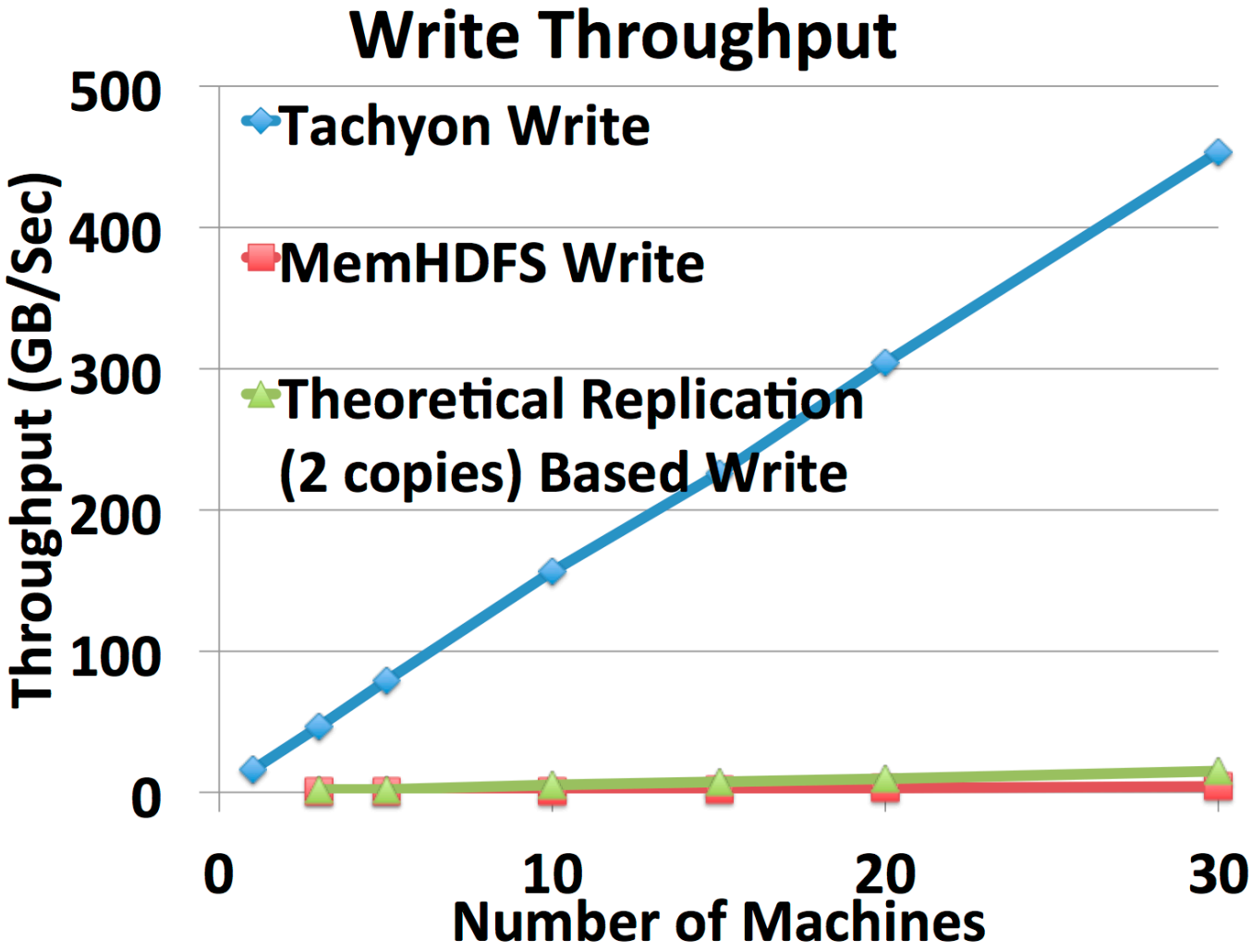
- Priority Based Scheduler



- Fair Sharing Based Scheduler

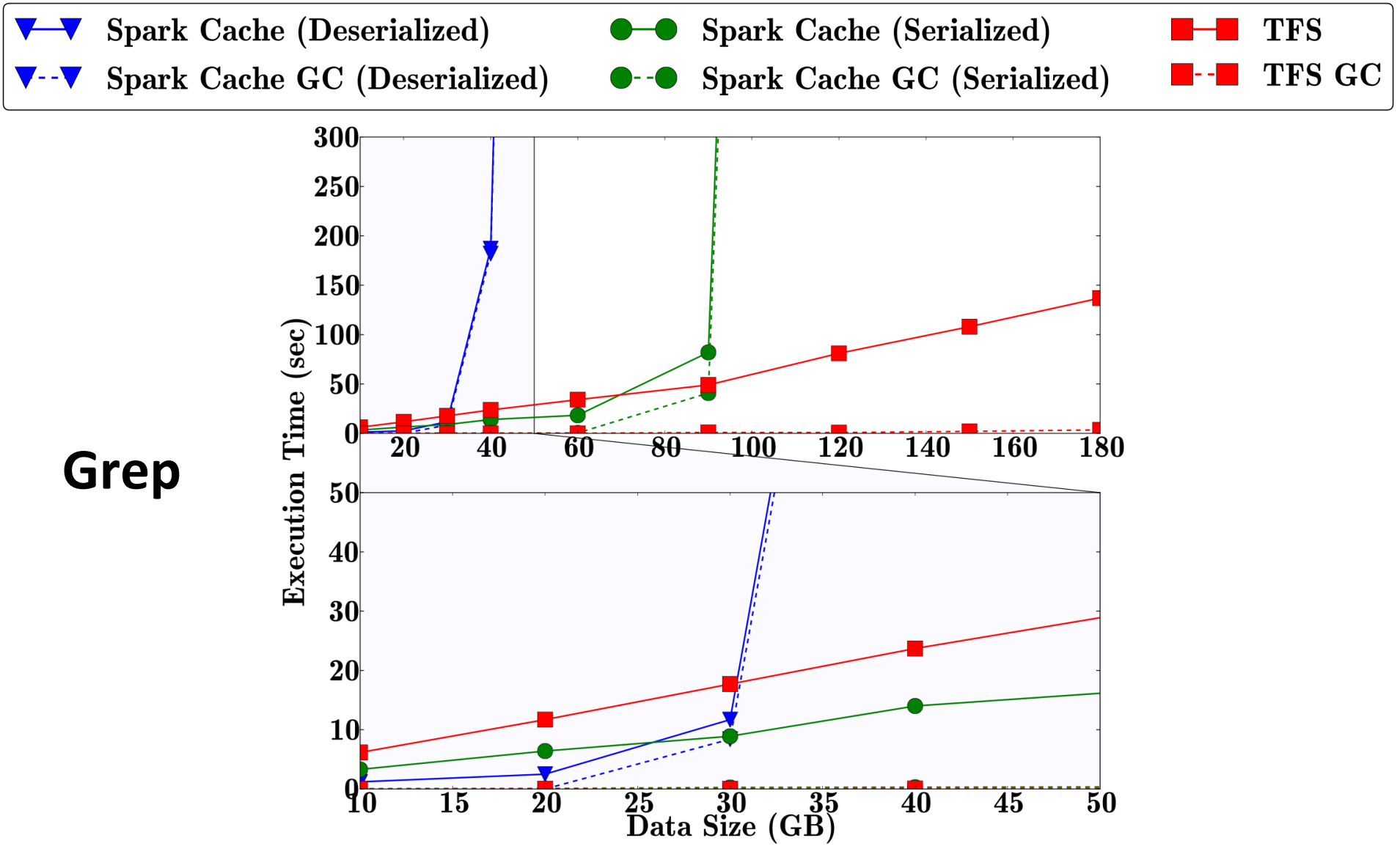


Comparison with in Memory HDFS

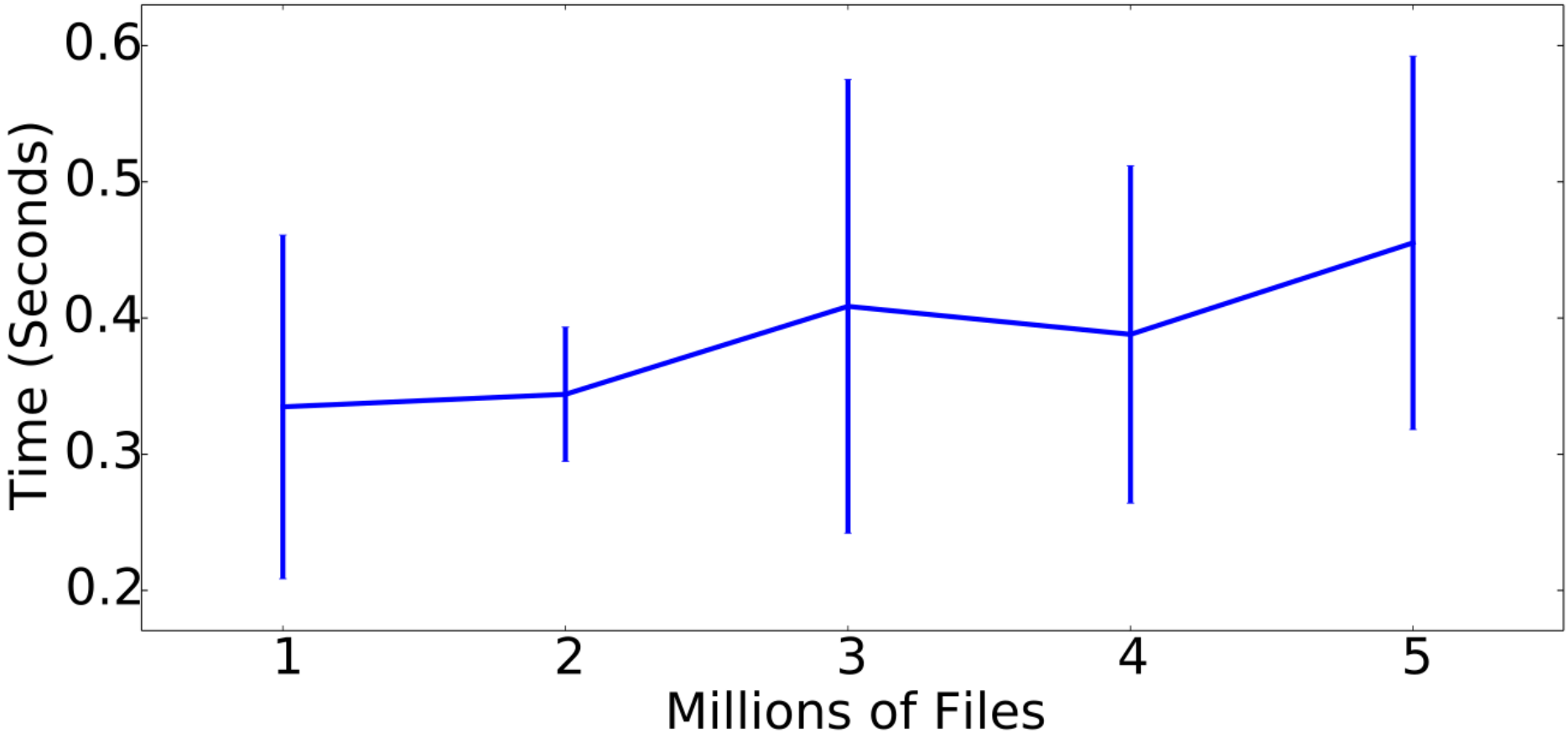


Further Improve Spark's Performance

Grep



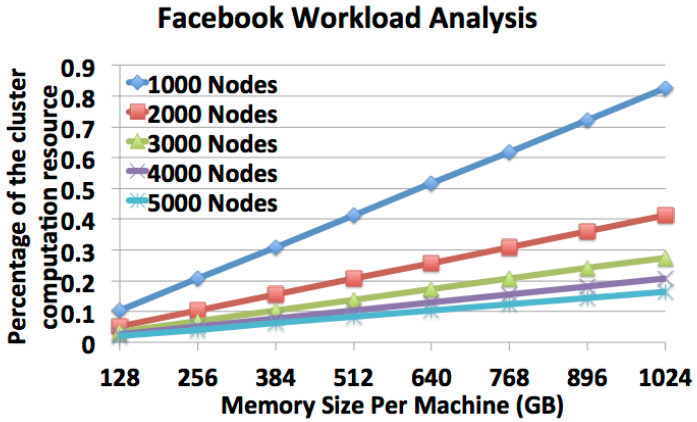
Master Faster Recovery

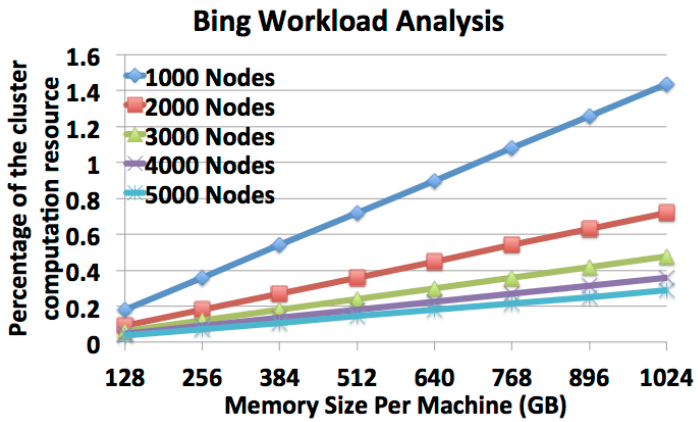


Recomputation Resource Consumption

Bin	Tasks	% of Jobs	
		Facebook	Bing
1	1 - 10	85%	43%
2	11 - 50	4%	8%
3	51 - 150	8%	24%
4	151 - 500	2%	23%
5	> 500	1%	2%

Trace Summary





Outline

- Overview
- Research
- **Open Source**
- Future

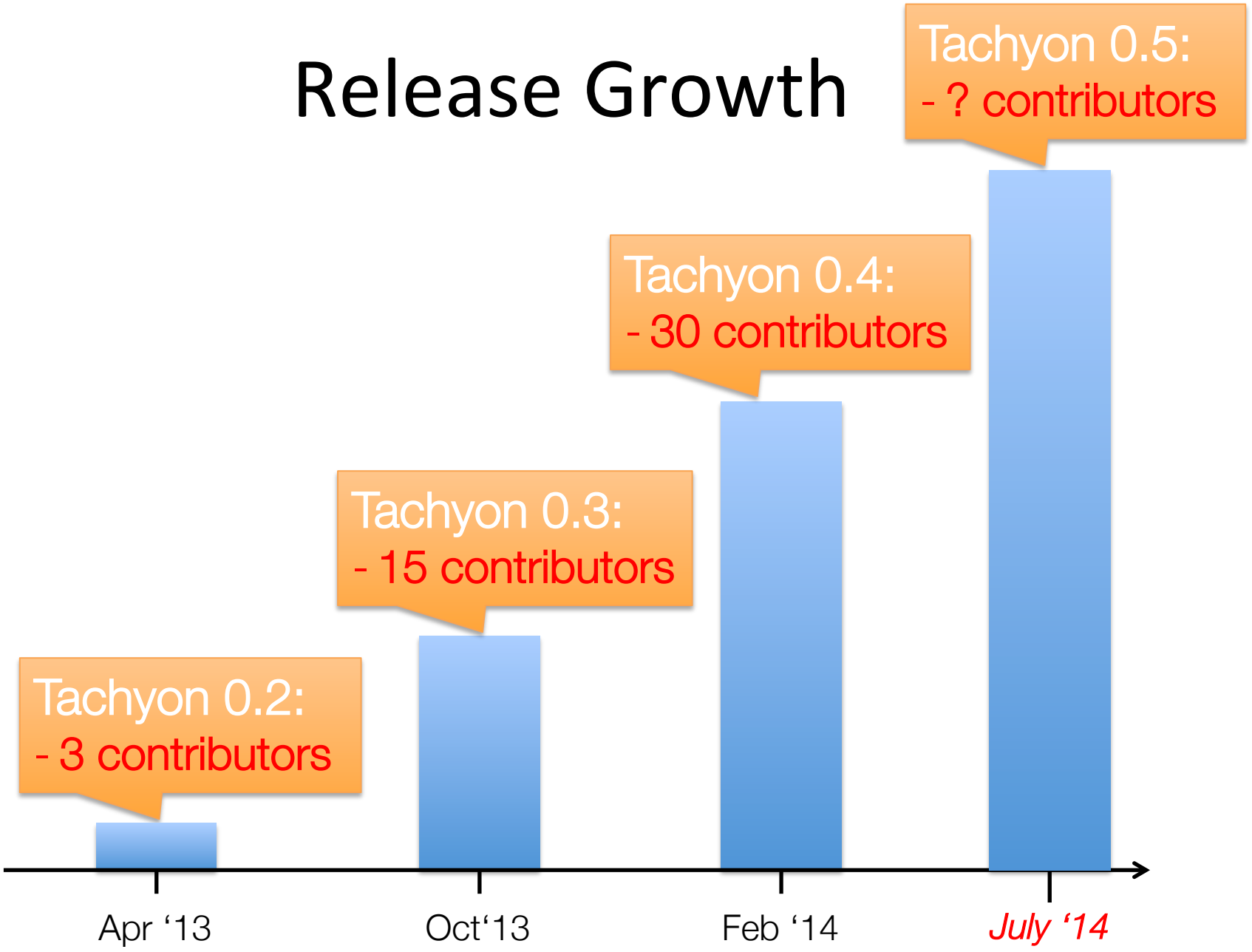
TACHYON Open Source Status

- Apache License, Version 0.4.1 (Feb 2014)

- 15+ Companies

- Spark and MapReduce applications can run without any code change

Release Growth



Contributors Outside of Berkeley

More than 75%

Tachyon is the Default Off-Heap Storage Solution for Spark

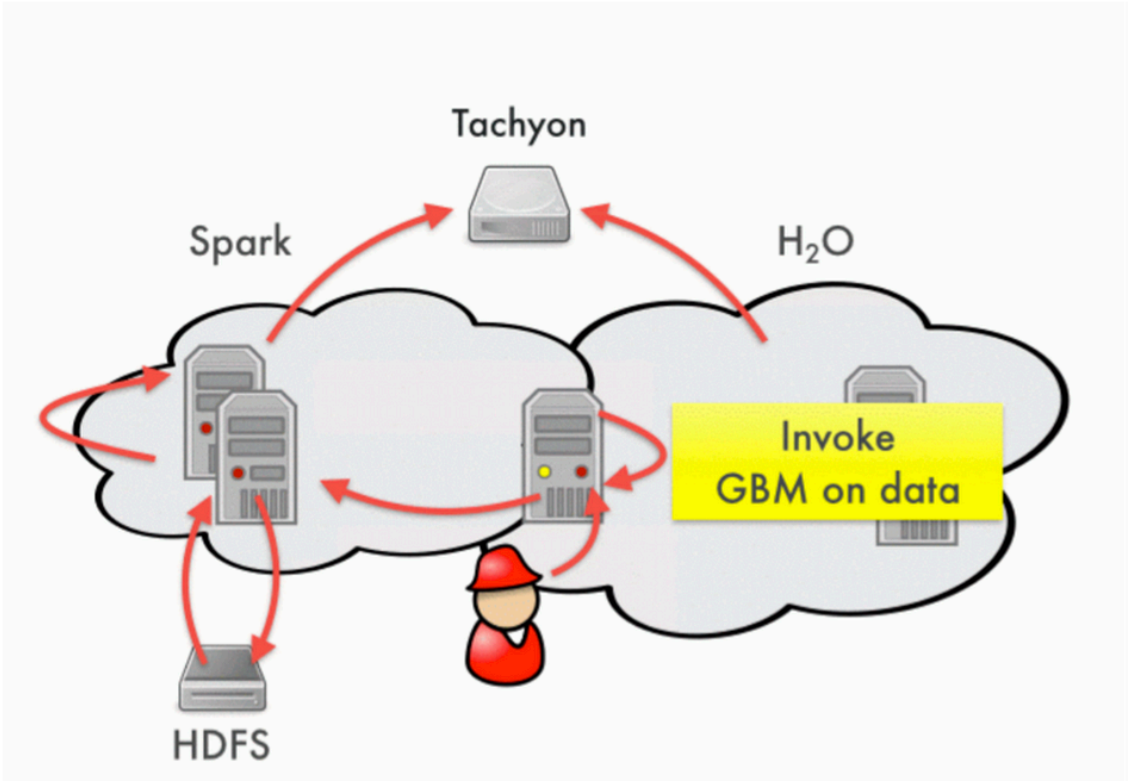
Tachyon
is in Fedora 20

Thanks to Redhat!

Commercially Supported By

TACHYON in Spark H2O

Today, data gets parsed and exchanged between Spark and H2O via Tachyon. Users can interactively query big data both via SQL and ML from within the same context.



Spark/MapReduce/Shark without Tachyon

- Spark
  - val file = sc.textFile(“hdfs://ip:port/path”)
- Hadoop MapReduce
  - hadoop jar hadoop-examples-1.0.4.jar wordcount hdfs://localhost:19998/input hdfs://localhost:19998/output
- Shark
  - CREATE TABLE orders_cached AS SELECT * FROM orders;

Spark/MapReduce/Shark with Tachyon

- Spark
  - val file = sc.textFile("tachyon://ip:port/path")
- Hadoop MapReduce
  - hadoop jar hadoop-examples-1.0.4.jar wordcount tachyon://localhost:19998/input tachyon://localhost:19998/output
- Shark
  - CREATE TABLE orders_tachyon AS SELECT * FROM orders;

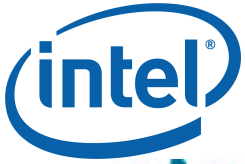
Spark OFF_HEAP with Tachyon

```
// Input data from Tachyon's Memory
val file = sc.textFile("tachyon://ip:port/path")

// Store RDD OFF_HEAP in Tachyon's Memory
file.persist(OFF_HEAP)
```

Thanks to our Code Contributors!

Aaron Davidson
Achal Soni
Ali Ghodsi
Andrew Ash
Anurag Khandelwal
Aslan Bekirov
Bill Zhao
Calvin Jia
Colin Patrick McCabe
Shivaram Venkataraman
Chang Cheng
Du Li
Fei Wang
Gerald Zhang
Grace Huang
Hao Cheng
Haoyuan Li
Henry Saputra
Hobin Yoon
Huamin Chen
Jey Kottalam
Joseph Tang
Lukasz Jastrzebski
Manu Goyal
Mark Hamstra
Nick Lanham
Orcun Simsek
Pengfei Xuan
Qifan Pu
Qianhao Dong
Raymond Liu
Reynold Xin
Robert Metzger
Rong Gu
Sean Zhong
Srinivas Parayya
Tao Wang
Timothy St. Clair
Vamsi Chitters
Xi Liu
Xiaomin Zhang

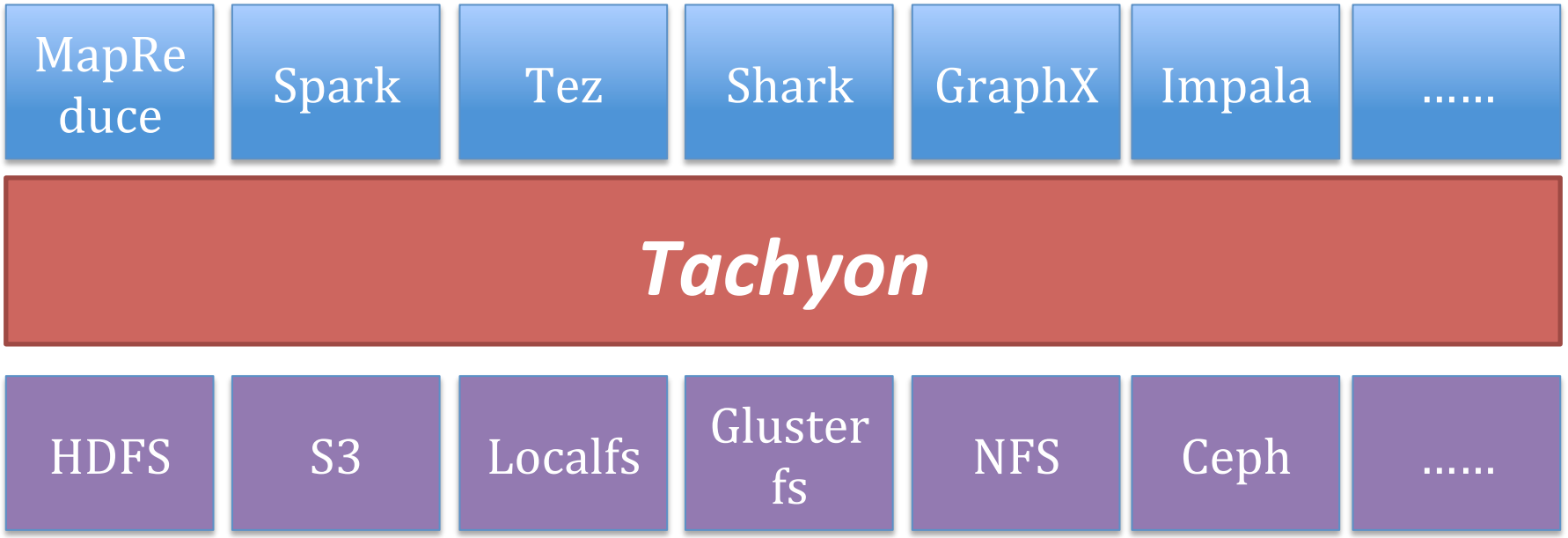



Outline

- Overview
- Research
- Open Source
- **Future**

Goal?

Better Assist Other Components



Welcome Collaboration!

Short Term Roadmap

- Ceph Integration (Redhat)
- Hierarchical Local Storage (Intel)
- Further improver Shark Performance (Yahoo)
- Better support for Multi-tenancy (AMPLab)
- ***Many more*** from AMPLab and Industry Collaborators.

Your Requirements?

Tachyon Summary

- High-throughput, fault-tolerant memory centric storage, with lineage as a first class citizen
- Further improve performance for frameworks such as Spark, Hadoop, and Shark etc.
- Healthy community with 15+ companies contributing

Thanks!

Questions?

- *More Information:*
  - *http://tachyon-project.org*
  - *https://github.com/amplab/tachyon*
- *Email:*
  - *haoyuan@cs.berkeley.edu*