TACHYON

A Reliable Memory-Centric Distributed Storage System

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

September, 2014, ISTC @ Portland

http://www.istc-cc.cmu.edu/

amplab UC Berkeley

Intel Science & Technology Center for Cloud Computing

Outline

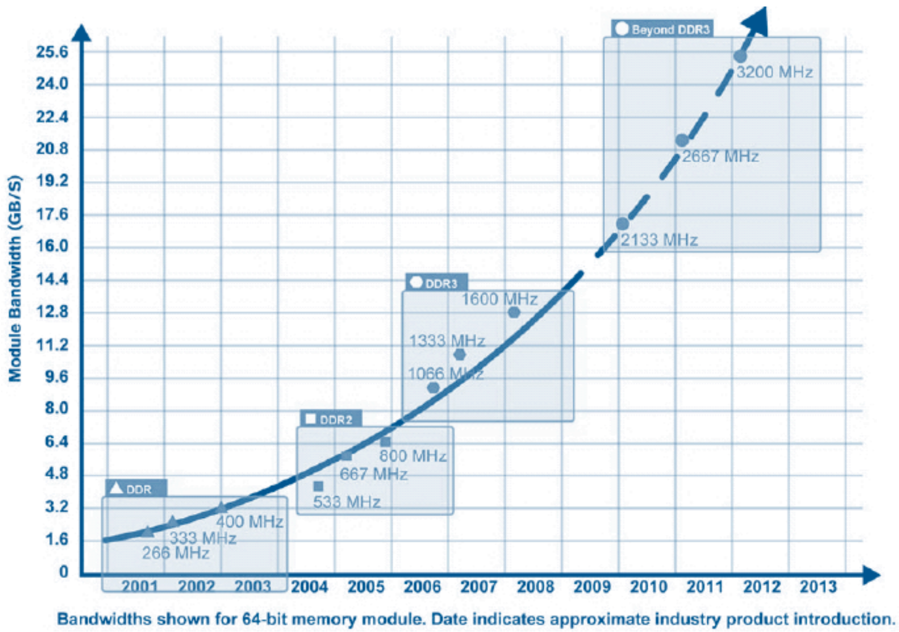
- Overview
  - Feature 1: Memory Centric Storage Architecture
  - Feature 2: Lineage in Storage
- Challenges
- Open Source
- Future

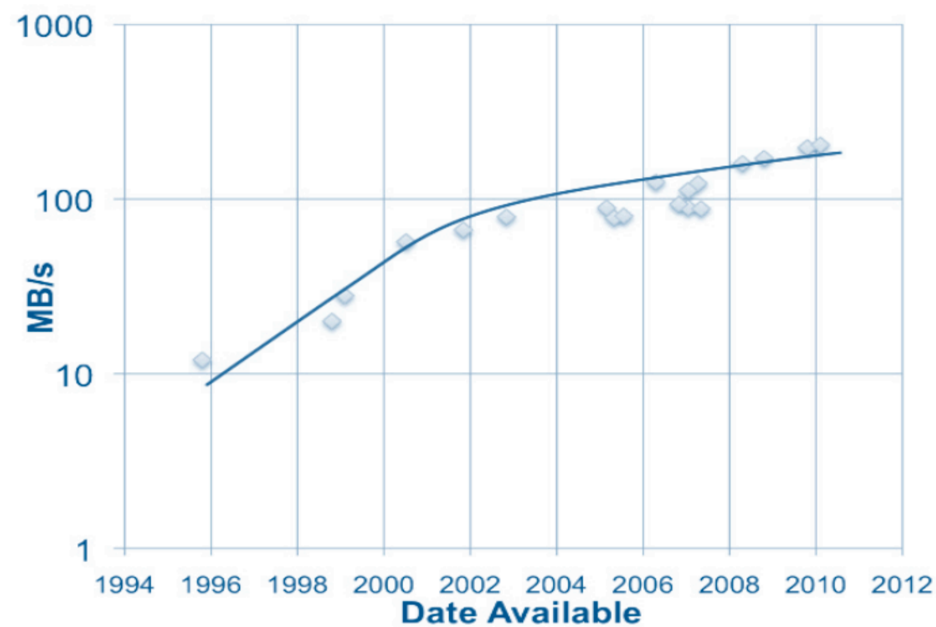
Outline

- **Overview**
  - Feature 1: Memory Centric Storage Architecture
  - Feature 2: Lineage in Storage
- Challenges
- Open Source
- Future

Memory is King

- RAM throughput increasing exponentially
- Disk throughput increasing slowly

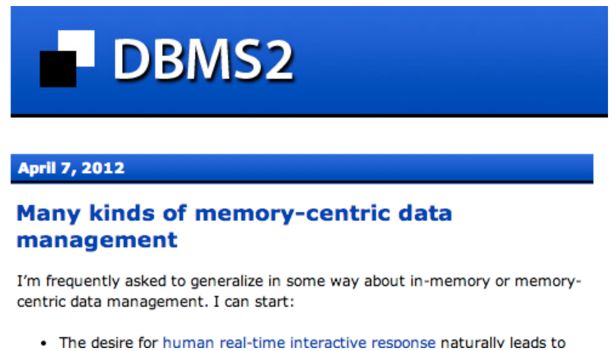




Memory-locality key to interactive response time

Realized by many…

- Frameworks already leverage memory


DBMS2

April 7, 2012

Many kinds of memory-centric data management

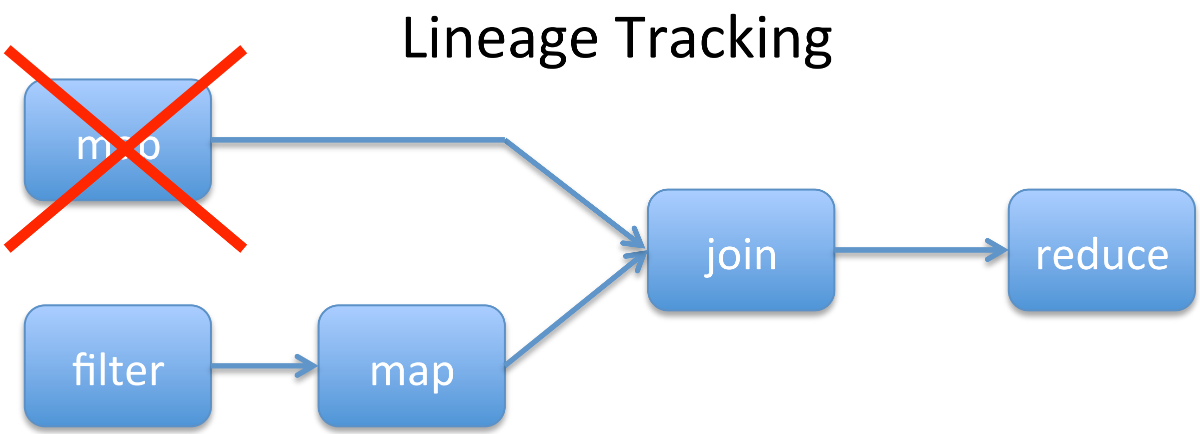
I'm frequently asked to generalize in some way about in-memory or memory-centric data management. I can start:

- The desire for human real-time interactive response naturally leads to

Problem solved?

An Example: Spark

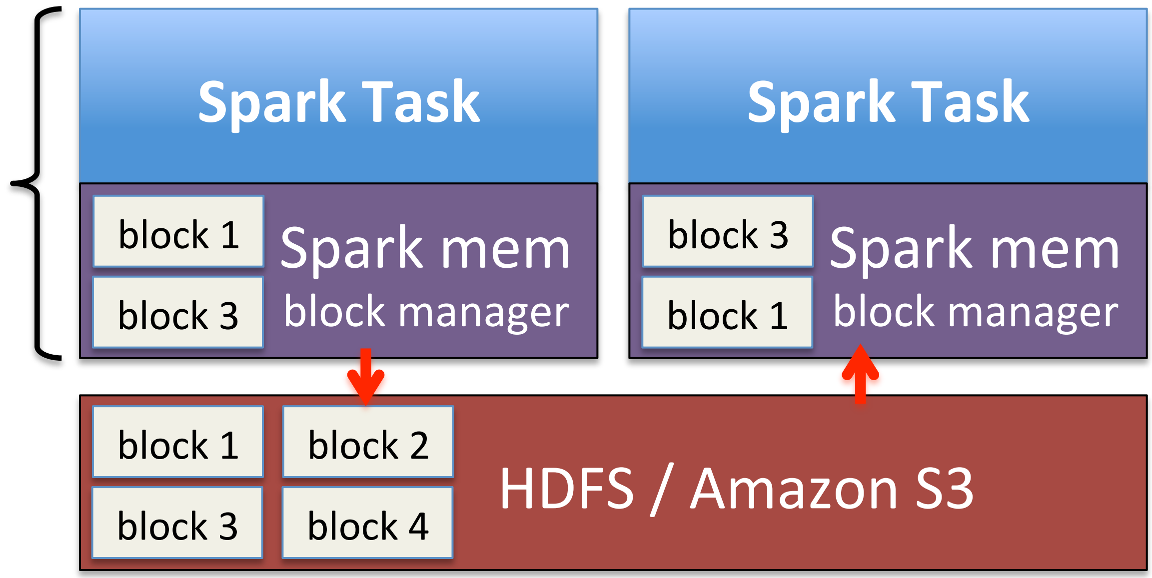
- Fast in-memory data processing framework
  - Keep one in-memory copy inside JVM
  - Track lineage of operations used to derive data
  - Upon failure, use lineage to recompute data



Issue 1

Data Sharing is the bottleneck in analytics pipeline: Slow writes to disk

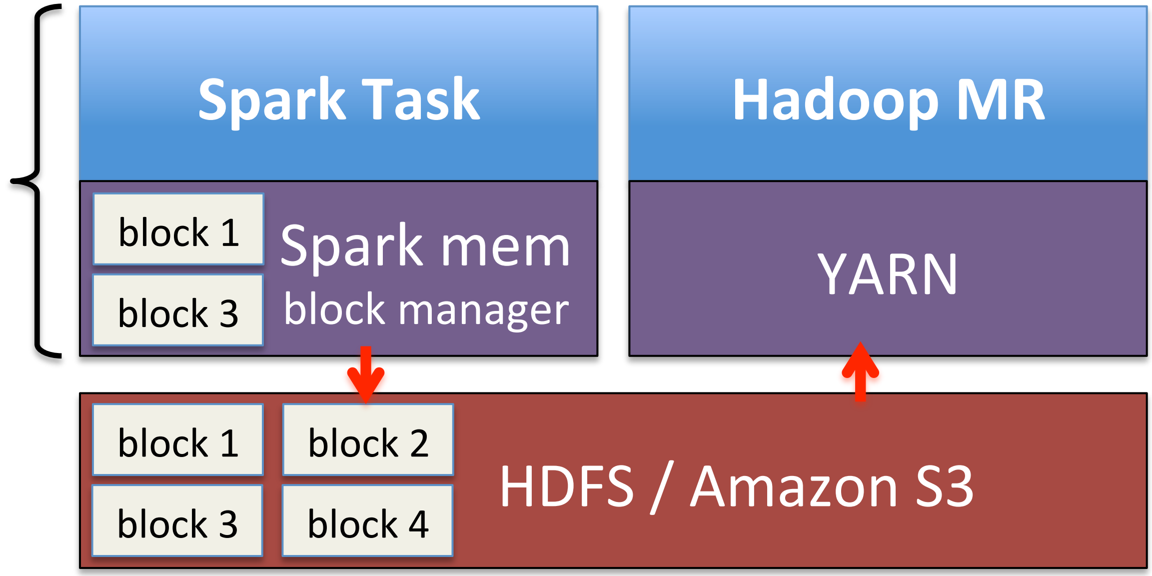
storage engine & execution engine same process (slow writes)



Issue 1

Data Sharing is the bottleneck in analytics pipeline:
Slow writes to disk

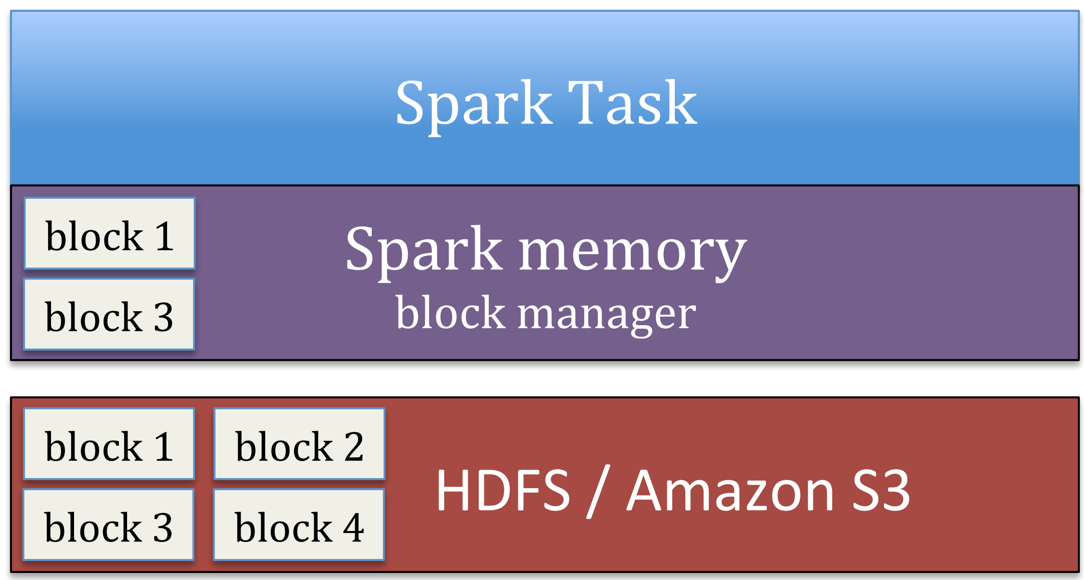
storage engine &
execution engine
same process
(slow writes)



Issue 2

Cache loss when process crashes.

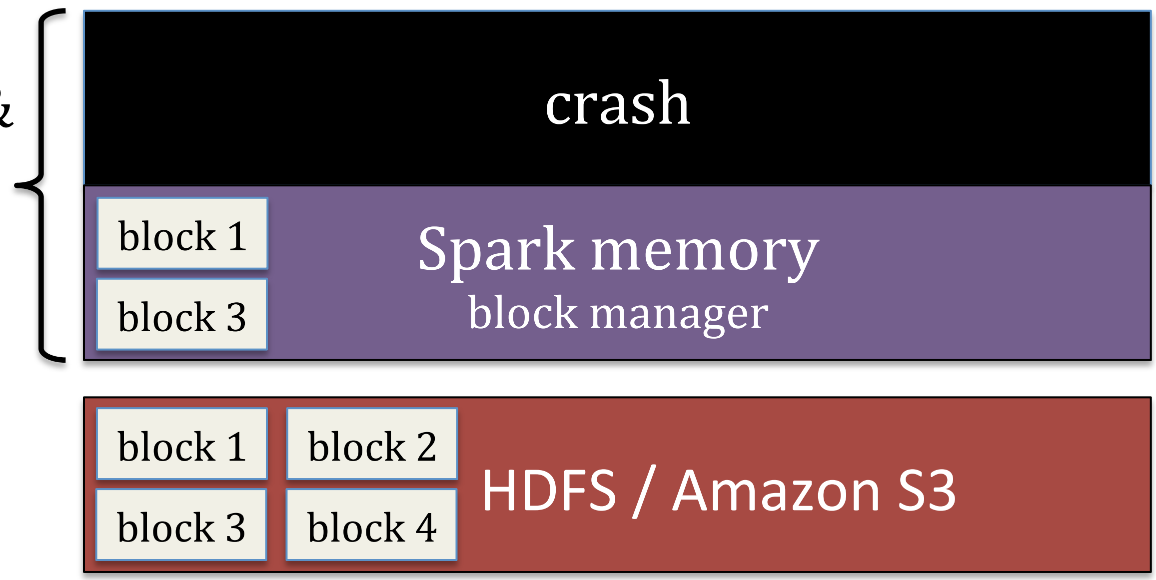
execution engine &
storage engine
same process



Issue 2

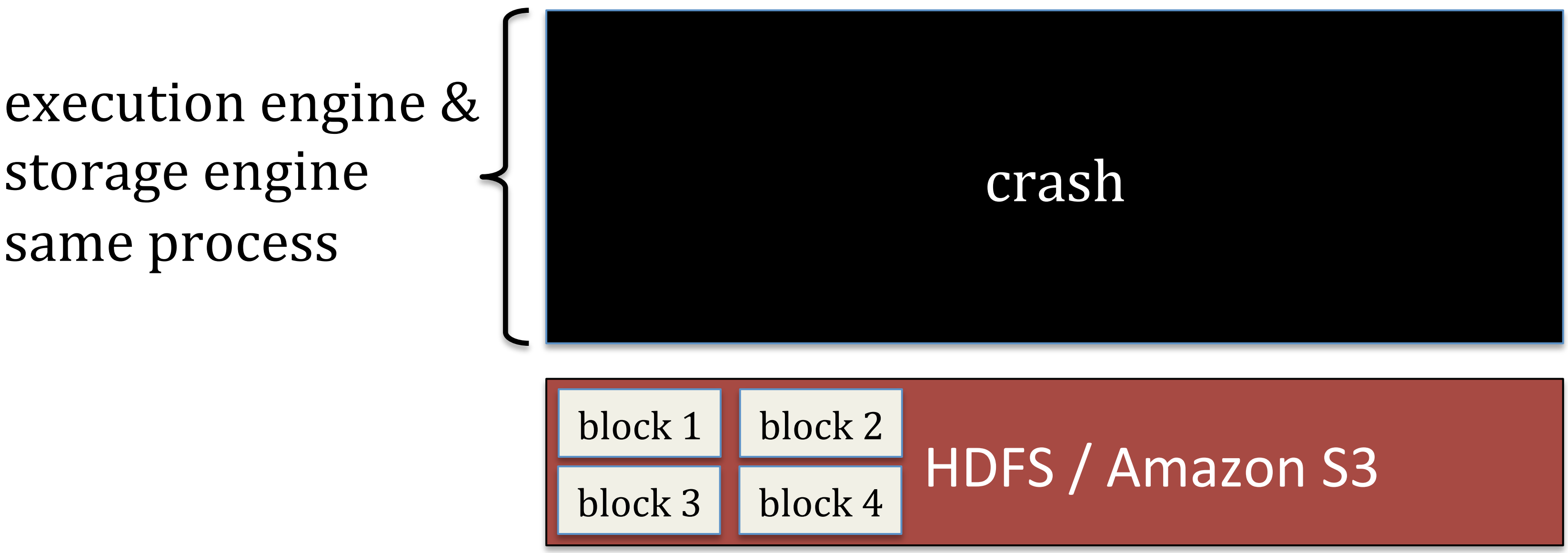
Cache loss when process crashes.

execution engine &
storage engine
same process



Issue 2

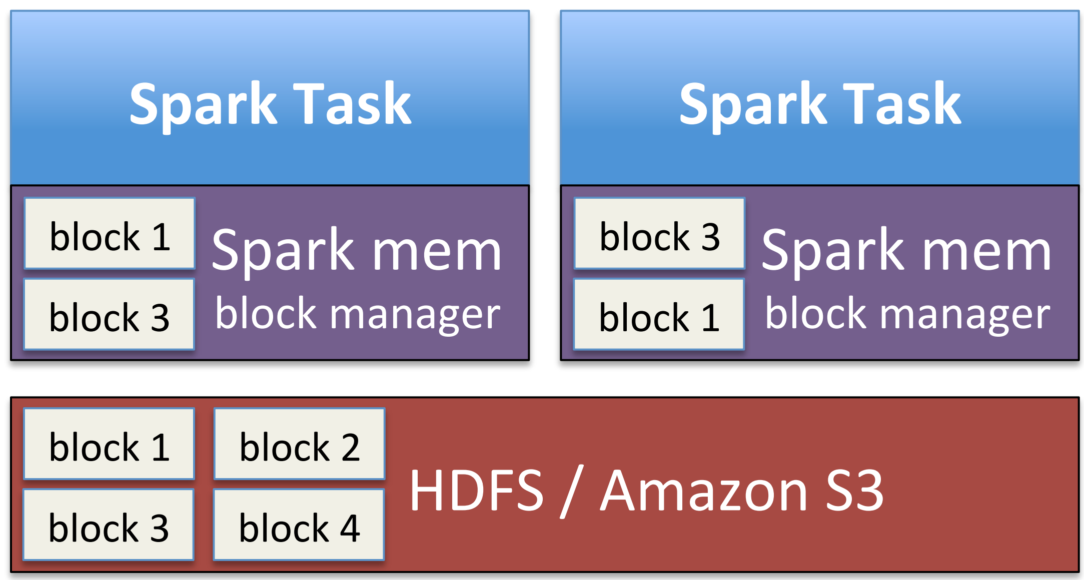
Cache loss when process crashes.



Issue 3

In-memory Data Duplication & Java Garbage Collection

execution engine &
storage engine
same process
(duplication & GC)



Tachyon

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

Solution Overview

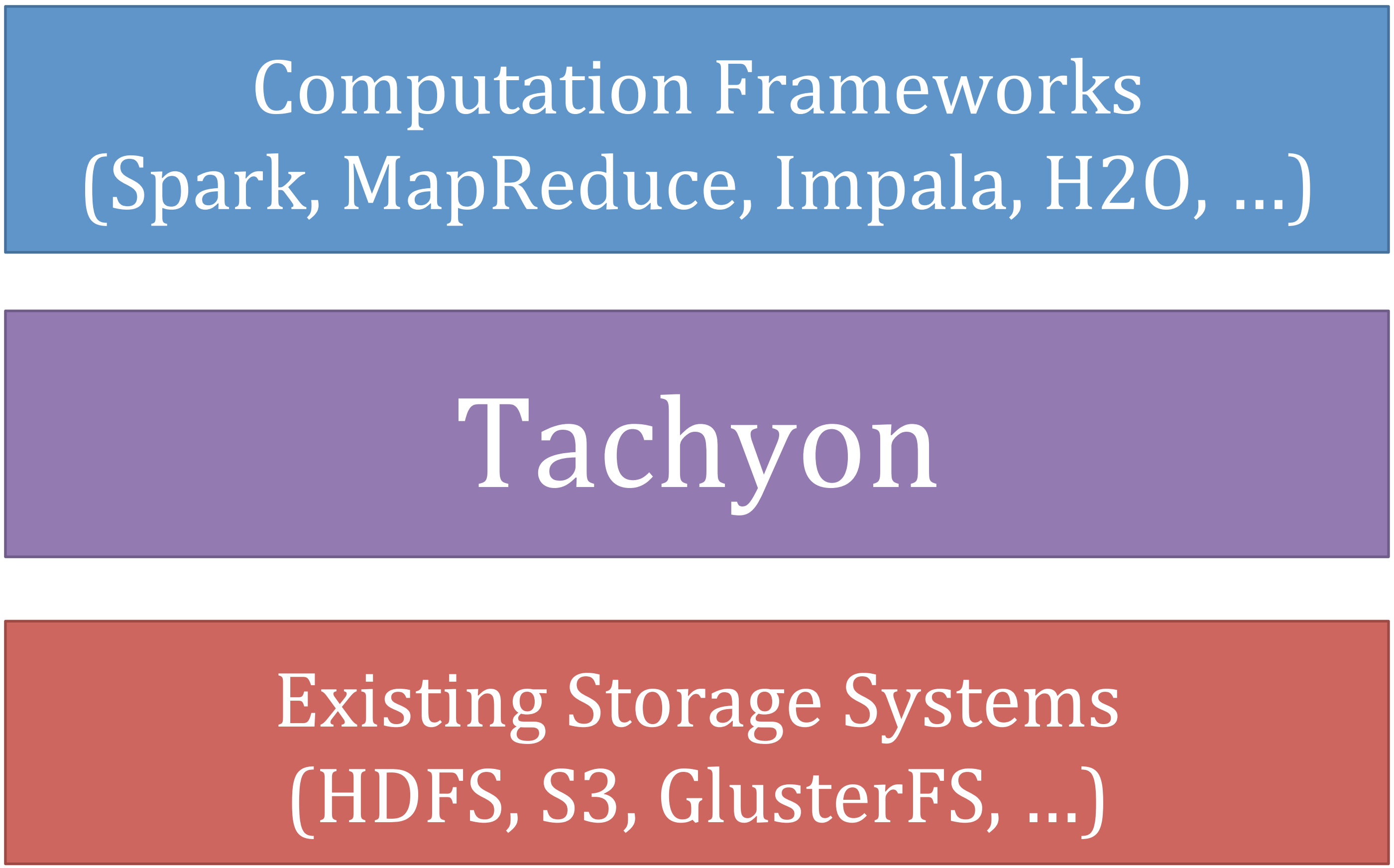
Basic idea

- Feature 1: memory-centric storage architecture
- Feature 2: push lineage down to storage layer

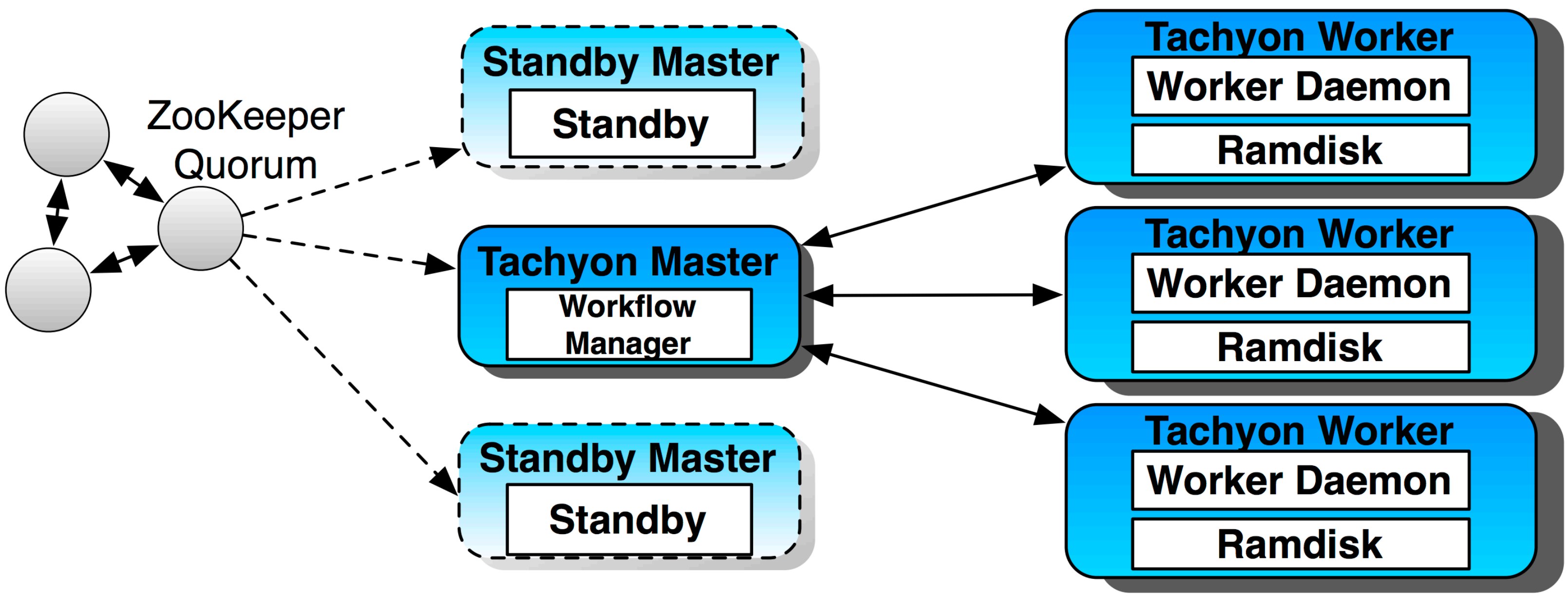
Facts

- One data copy in memory
- Recomputation for fault-tolerance

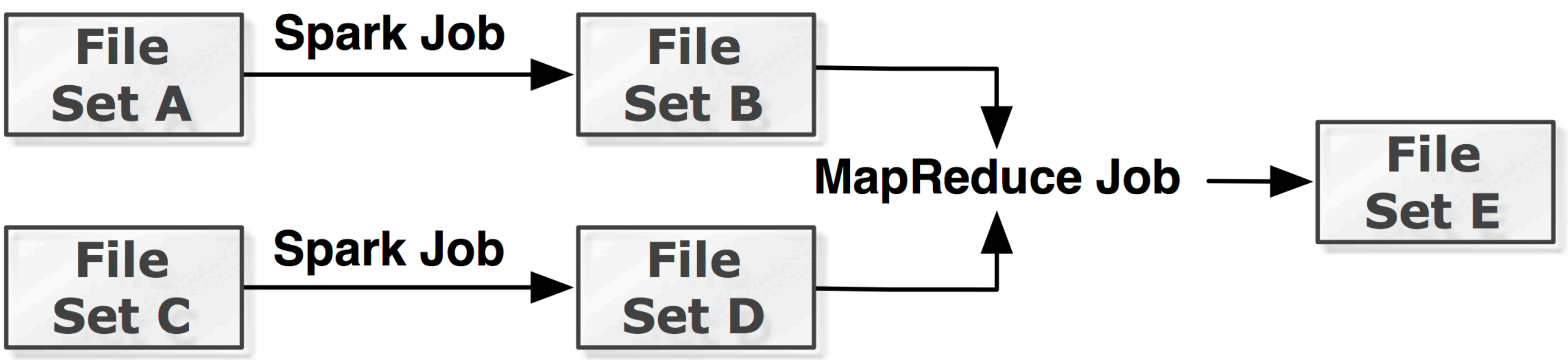
Stack


Computation Frameworks
(Spark, MapReduce, Impala, H2O, …)
Tachyon
Existing Storage Systems
(HDFS, S3, GlusterFS, …)

Memory-Centric Storage Architecture



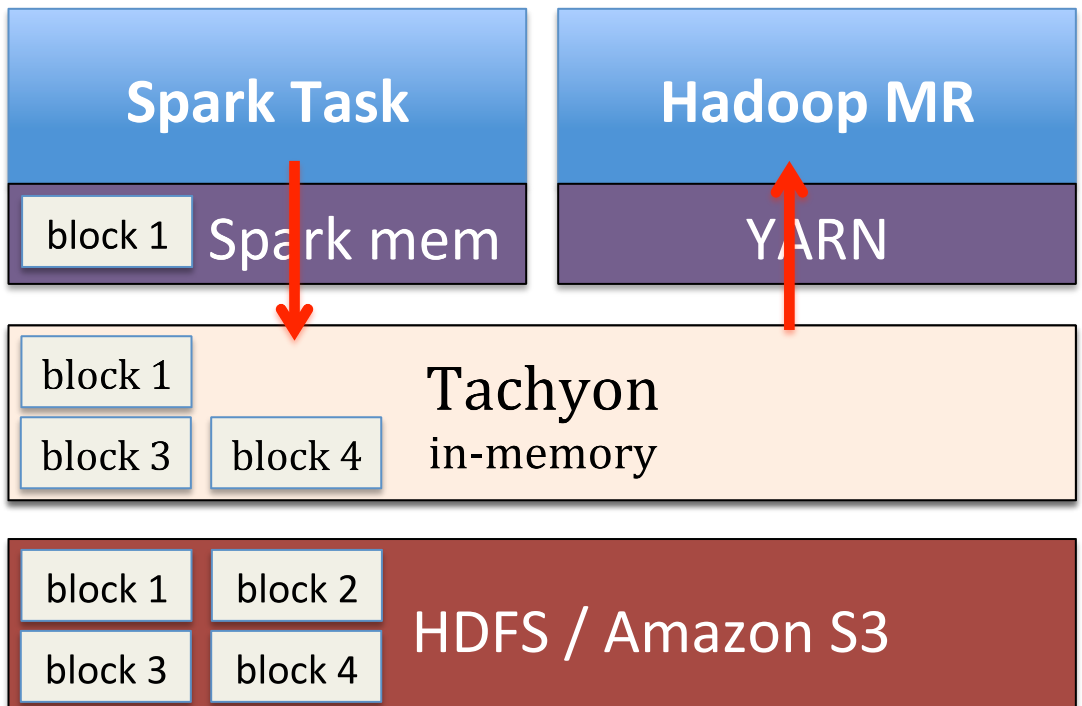
Lineage in Storage



Issue 1 revisited

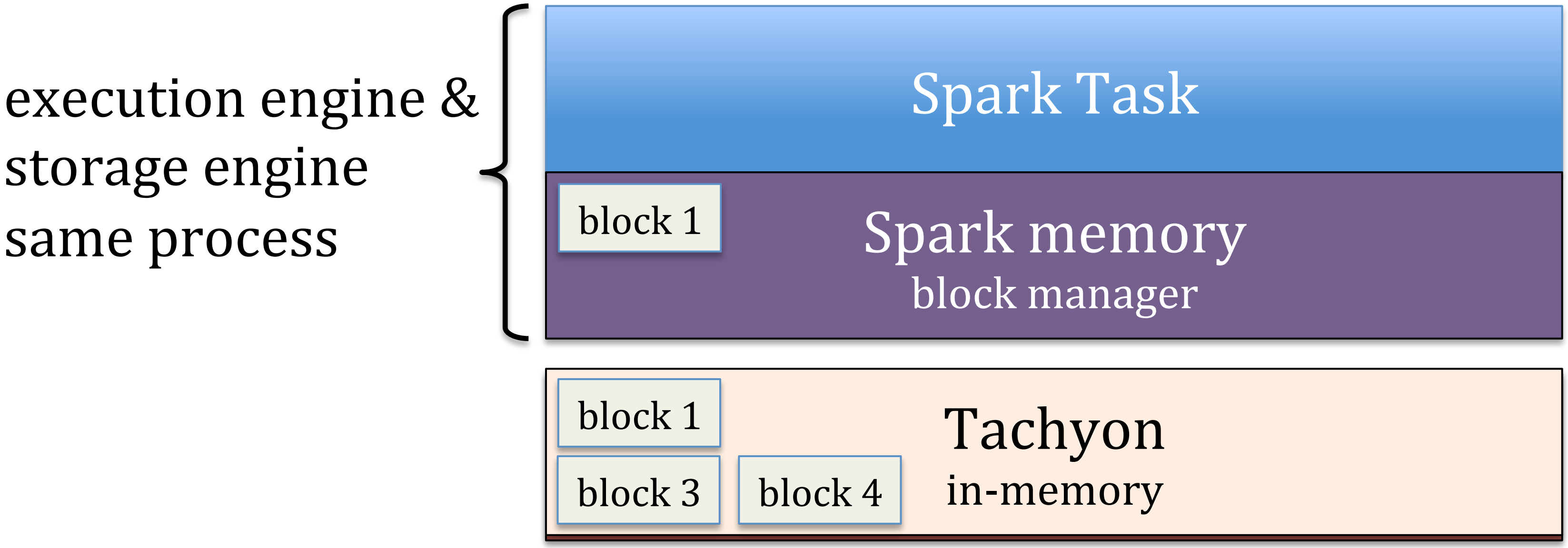
Memory-speed data sharing among jobs in different frameworks

execution engine &
storage engine
same process
(fast writes)



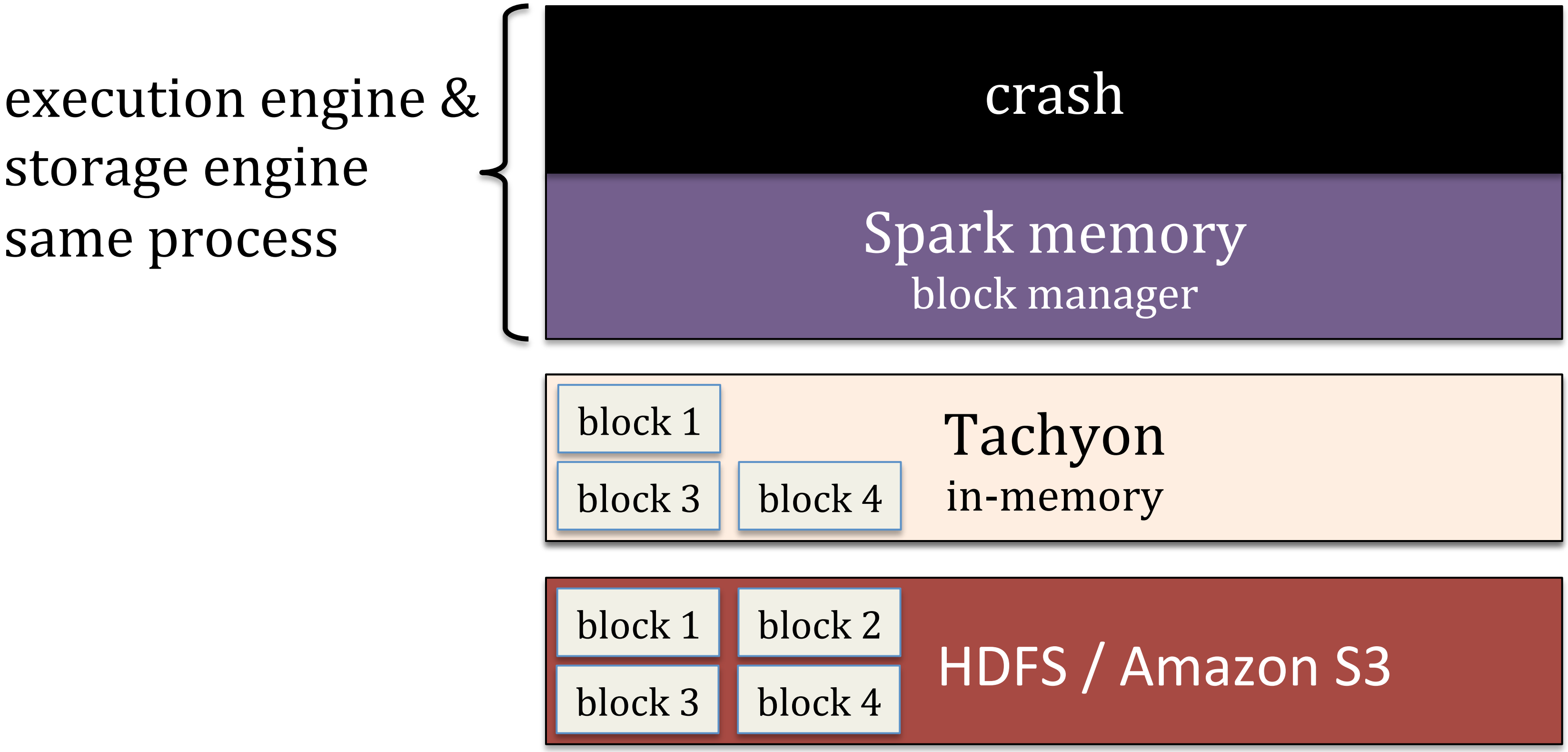
Issue 2 revisited

Keep in-memory data safe, even when a job crashes.



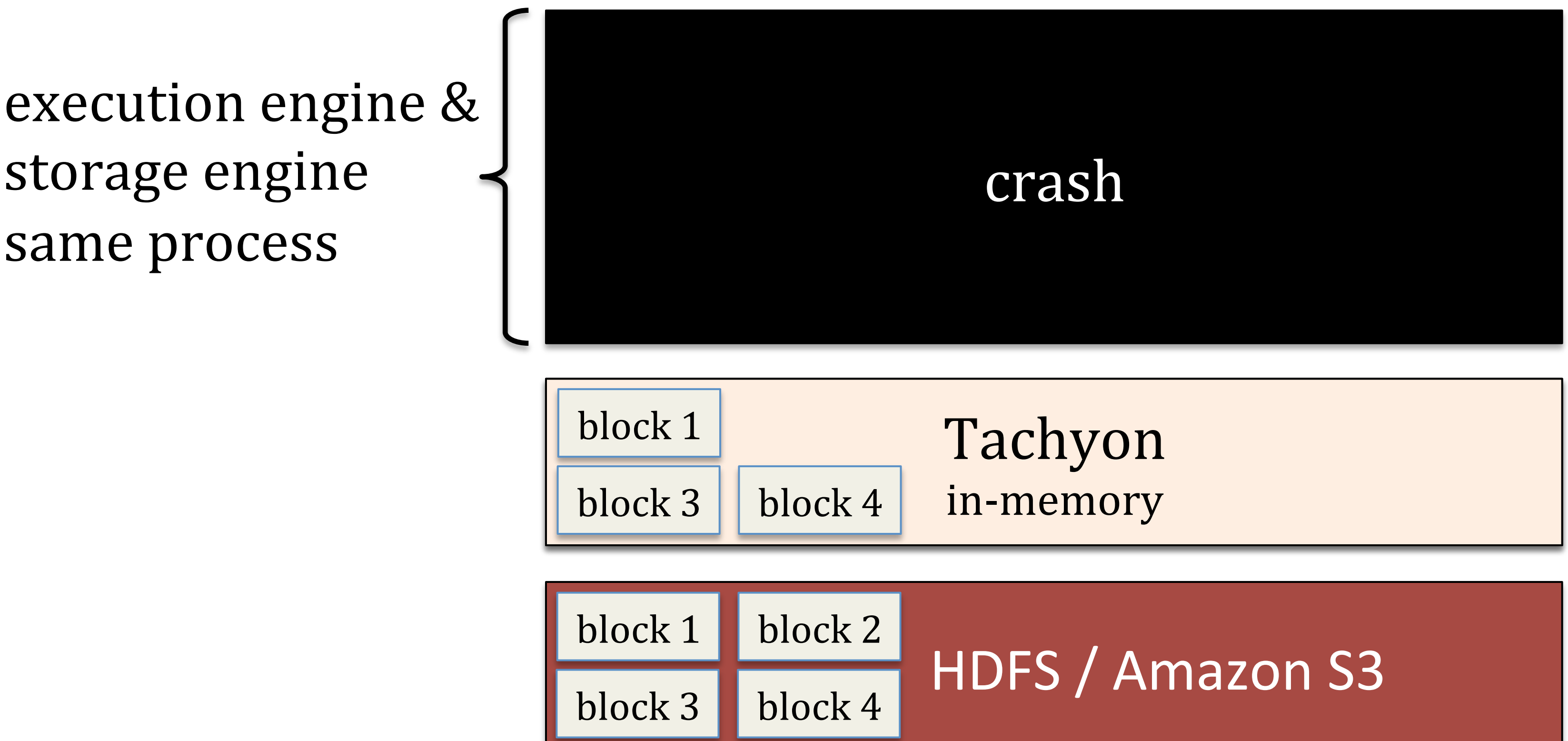
Issue 2 revisited

Keep in-memory data safe, even when a job crashes.



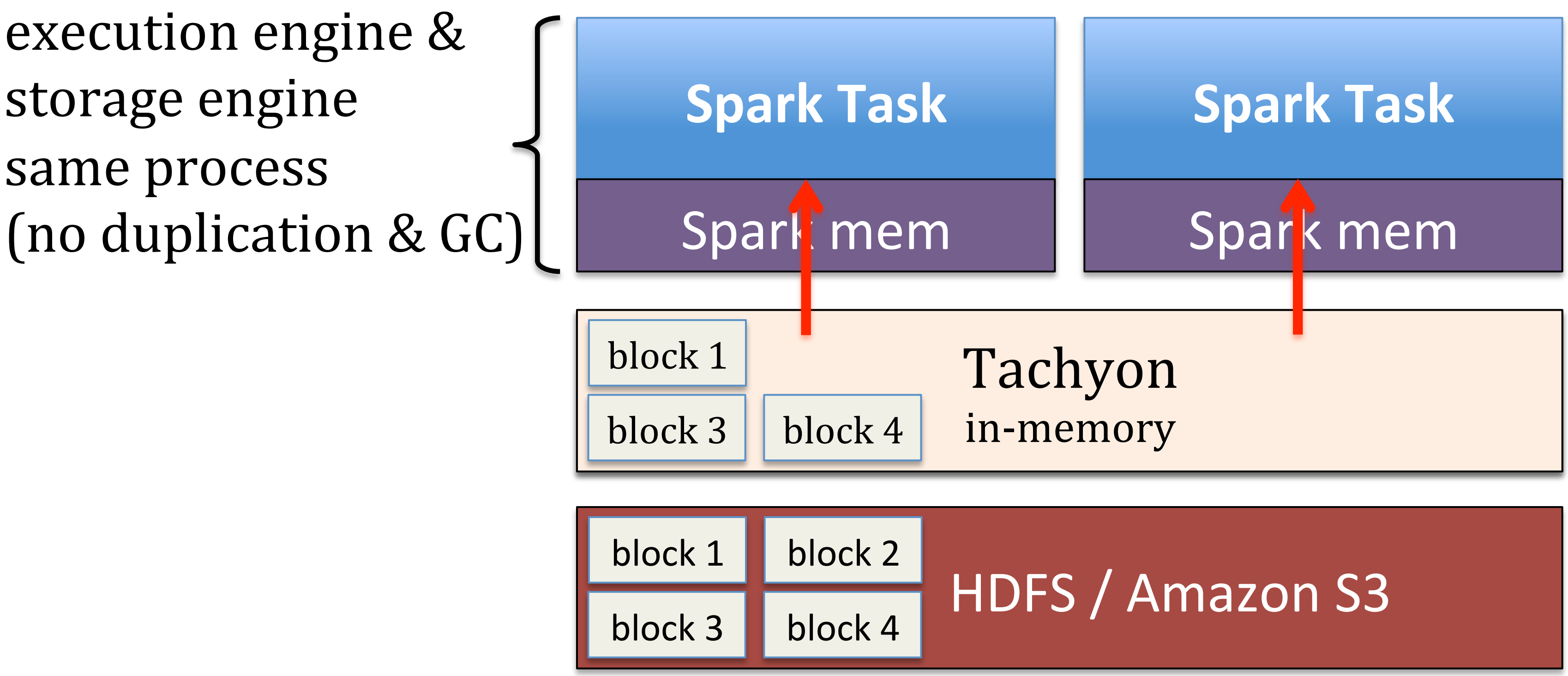
Issue 2 revisited

Keep in-memory data safe, even when a job crashes.



Issue 3 revisited

No in-memory data duplication, much less GC



Outline

- Overview
  - Feature 1: Memory Centric Storage Architecture
  - Feature 2: Lineage in Storage
- **Challenges**
- Open Source
- Future

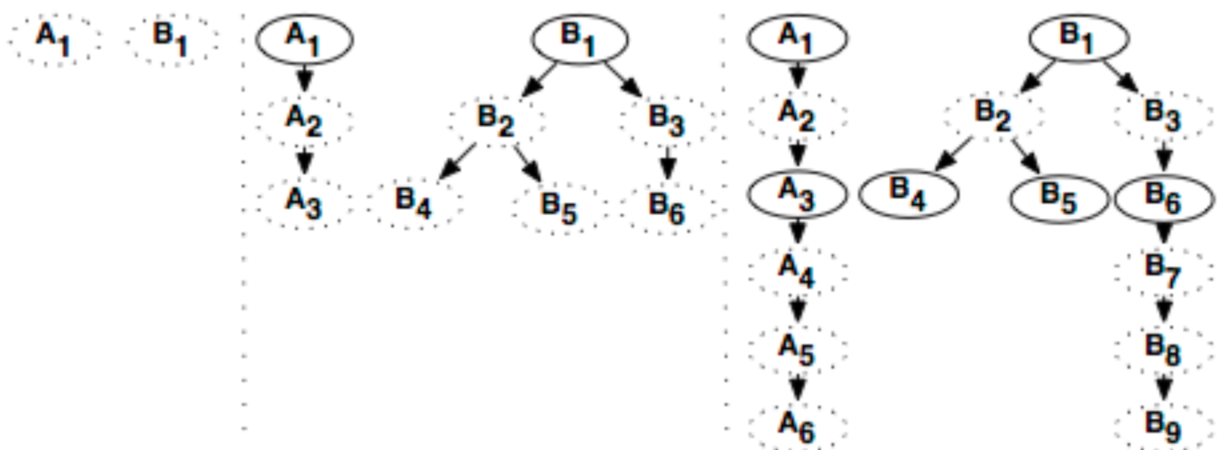
Question 1: How long to get missing data back?

That server contains the data computed last month!

Lineage enables Asynchronous Checkpointing

Edge Algorithm

- Checkpoint leaves
- Checkpoint hot files
- Bounded Recovery Cost

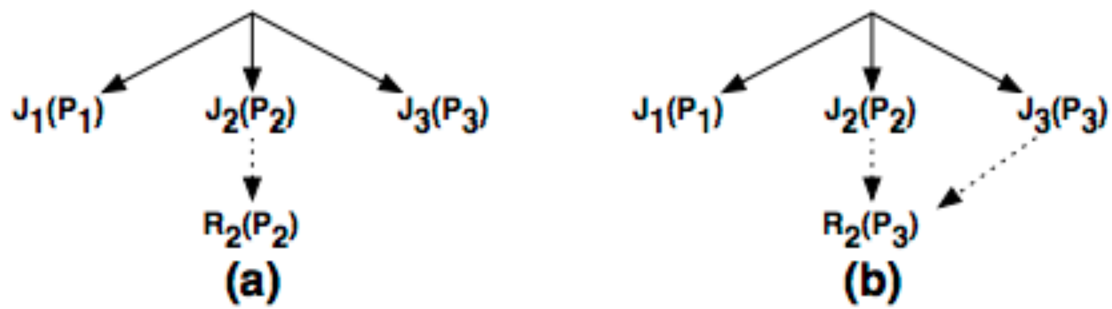


Question 2: How to allocate recomputation resource?

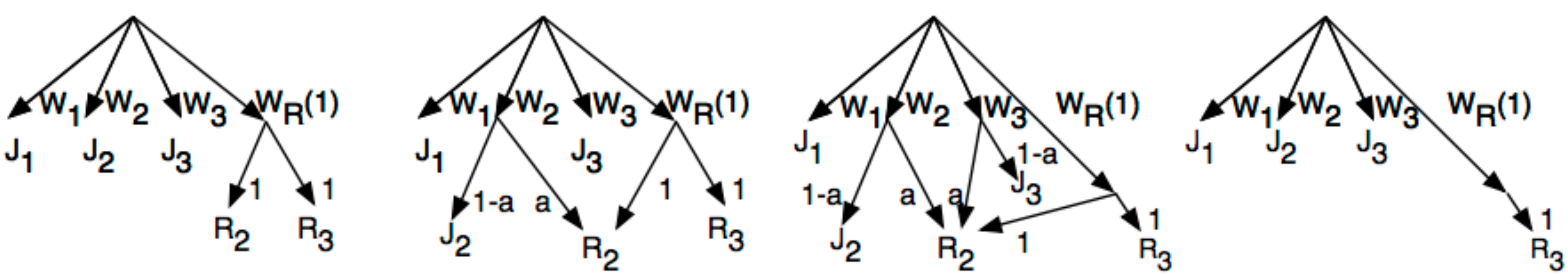
Would recomputation slow down my high priority jobs?
Priority Inversion?

Recomputation Resource Allocation

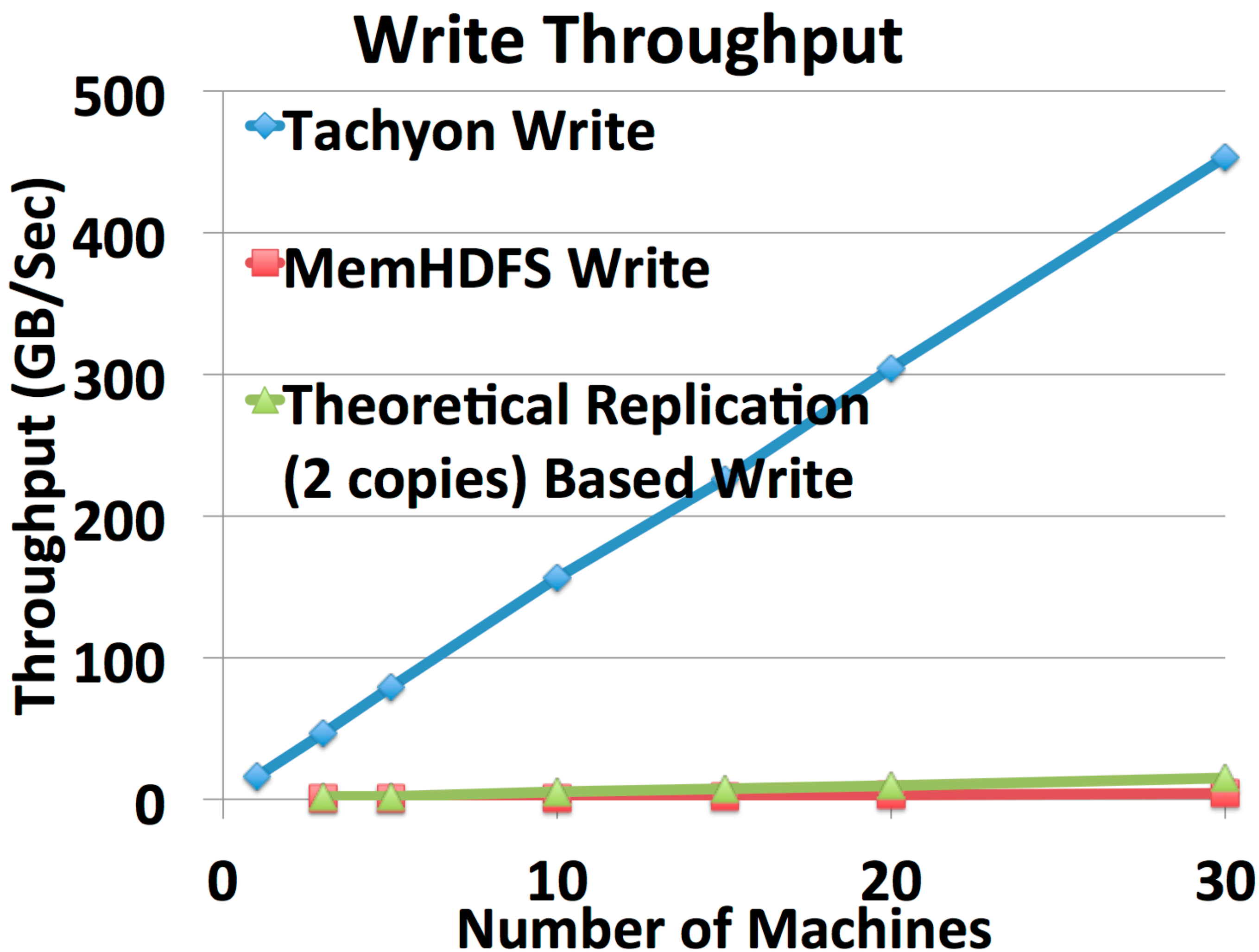
- Priority Based Scheduler



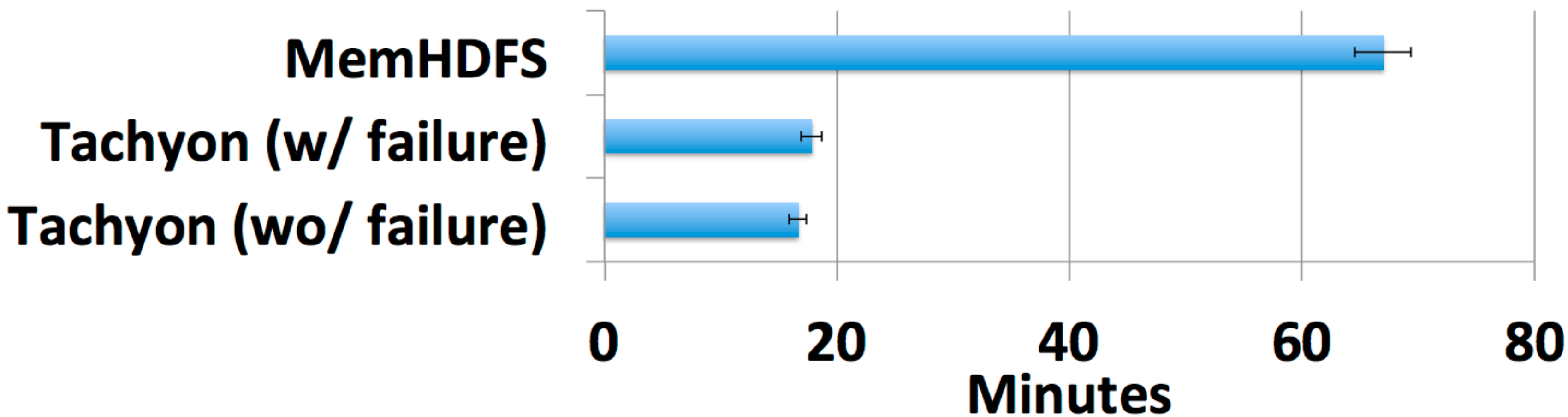
- Fair Sharing Based Scheduler



Comparison with in Memory HDFS



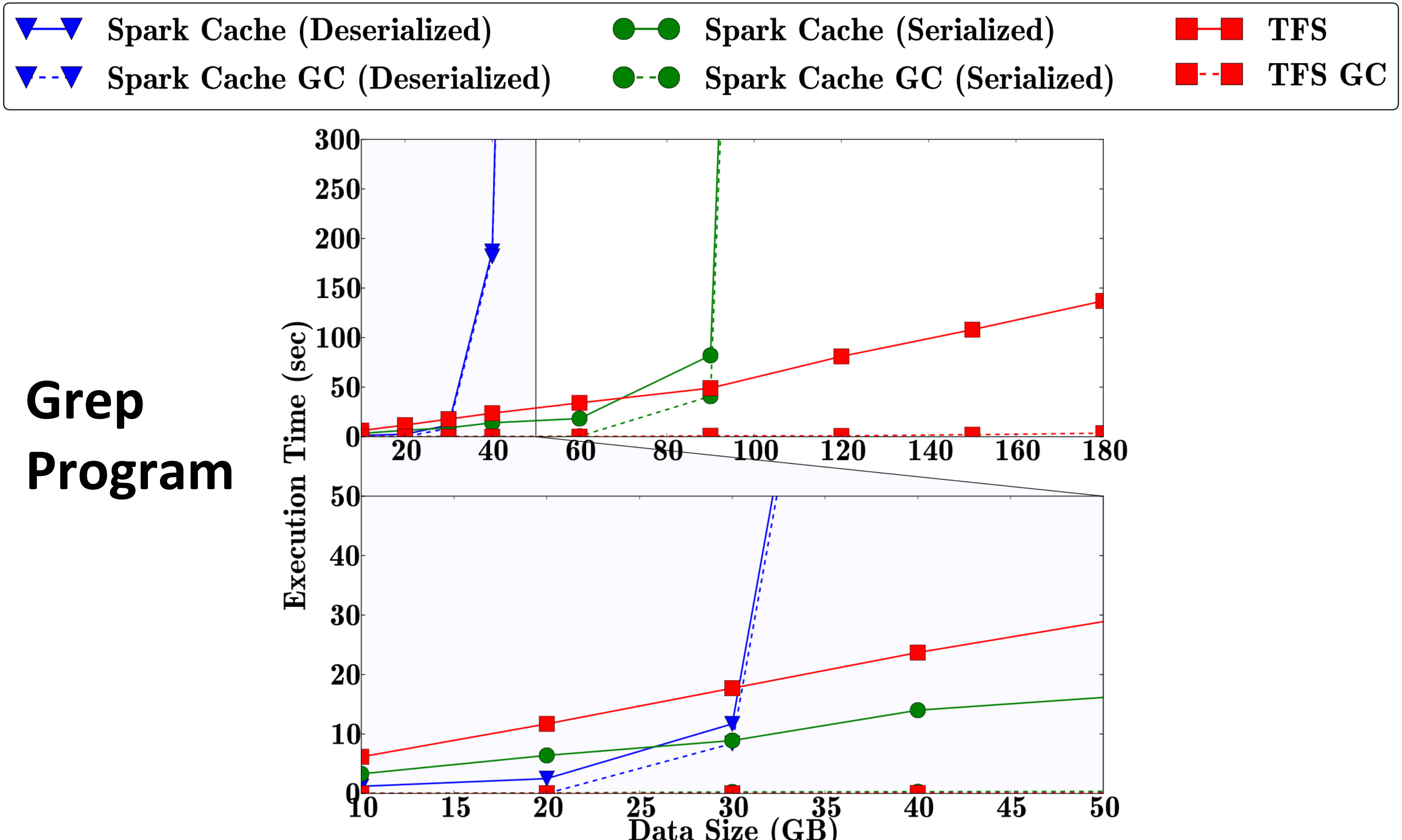
Workflow Improvement



Performance comparison for realistic workflow. The workflow ran 4x faster on Tachyon than on MemHDFS. In case of node failure, applications in Tachyon still finishes 3.8x faster.

Further Improve Spark's Performance

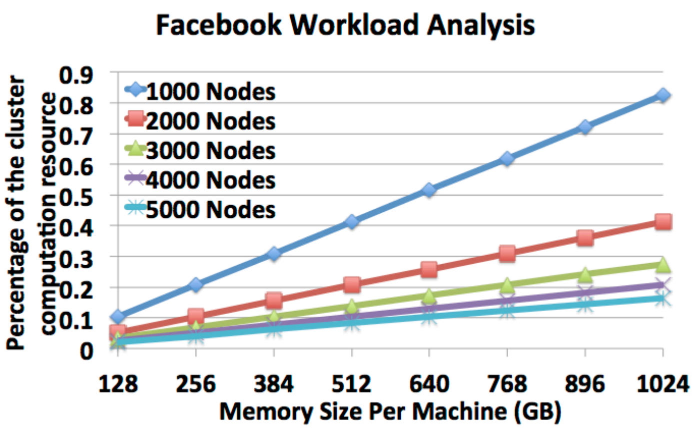
Grep Program

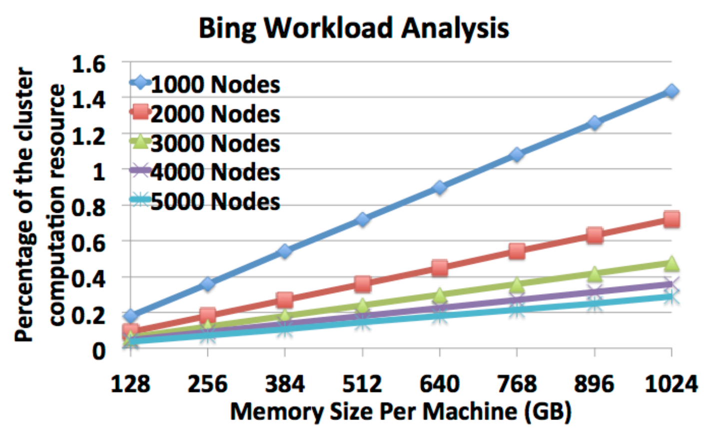


Recomputation Resource Consumption

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
  - Feature 1: Memory Centric Storage Architecture
  - Feature 2: Lineage in Storage
- Challenges
- **Open Source**
- Future

TACHYON Open Source Status

- Apache License 2.0, Version 0.5.0 (July 2014)

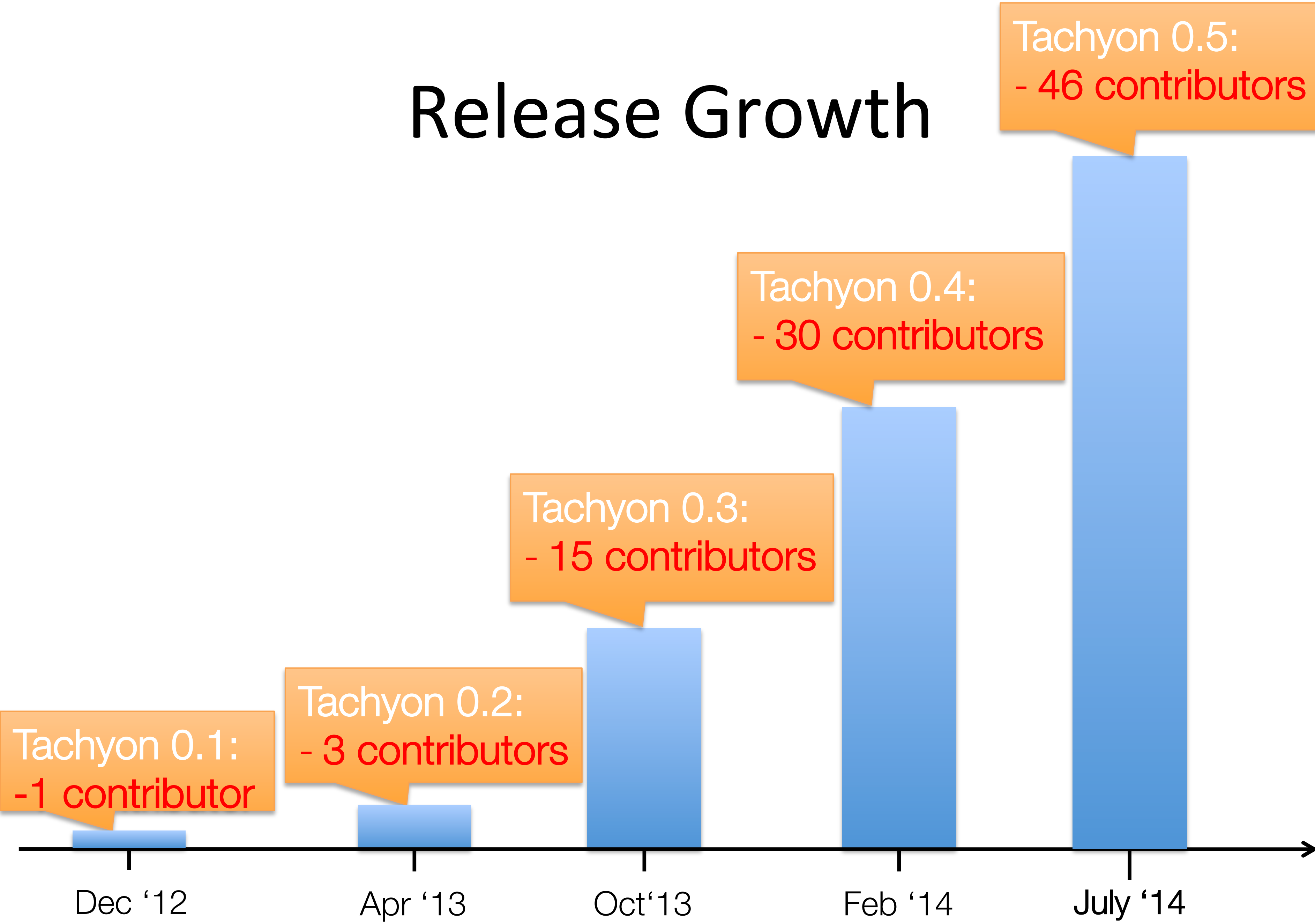
  PUBLIC amplab / tachyon Unstar 859

- Deployed at tens of companies

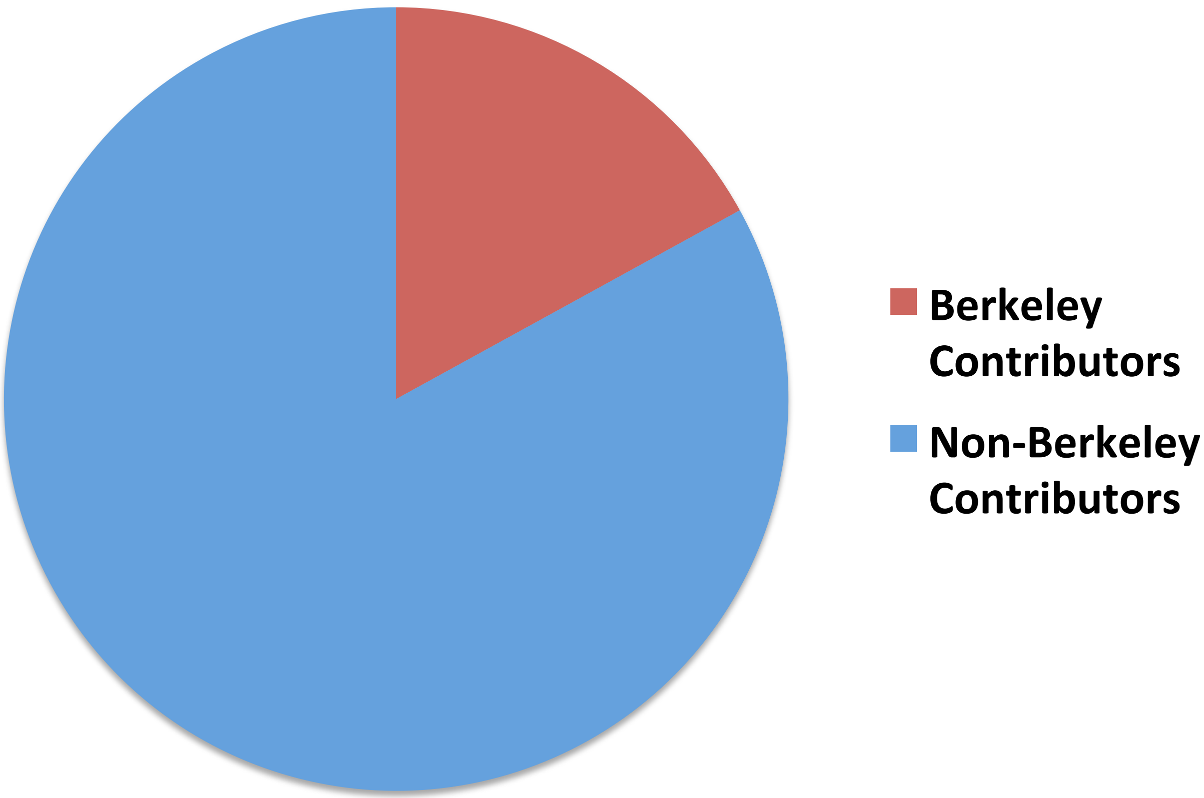
- 15+ Companies Contributing

- No code change for Spark and MapReduce applications.

Release Growth



Open Community



Thanks to our Code Contributors!

Aaron Davidson
Achal Soni
Ali Ghodsi
Andrew Ash
Anurag Khandelwal
Aslan Bekirov
Bill Zhao
Brad Childs
Calvin Jia
Chao Chen
Cheng Chang
Cheng Hao
Colin Patrick McCabe

David Capwell
David Zhu
Du Li
Fei Wang
Gerald Zhang
Grace Huang
Haoyuan Li
Henry Saputra
Hobin Yoon
Huamin Chen
Jey Kottalam
Joseph Tang
Juan Zhou

Lukasz Jastrzebski
Manu Goyal
Mark Hamstra
Mingfei Shi
Mubarak Seyed
Nick Lanham
Orcun Simsek
Pengfei Xuan
Qianhao Dong
Qifan Pu
Raymond Liu
Reynold Xin
Robert Metzger

Rong Gu
Sean Zhong
Seonghwan Moon
Shivaram Venkataraman
Srinivas Parayya
Tao Wang
Thu Kyaw
Timothy St. Clair
Vamsi Chitters
Xi Liu
Xiang Zhong
Xiaomin Zhang
Zhao Zhang

Tachyon is in Fedora 20

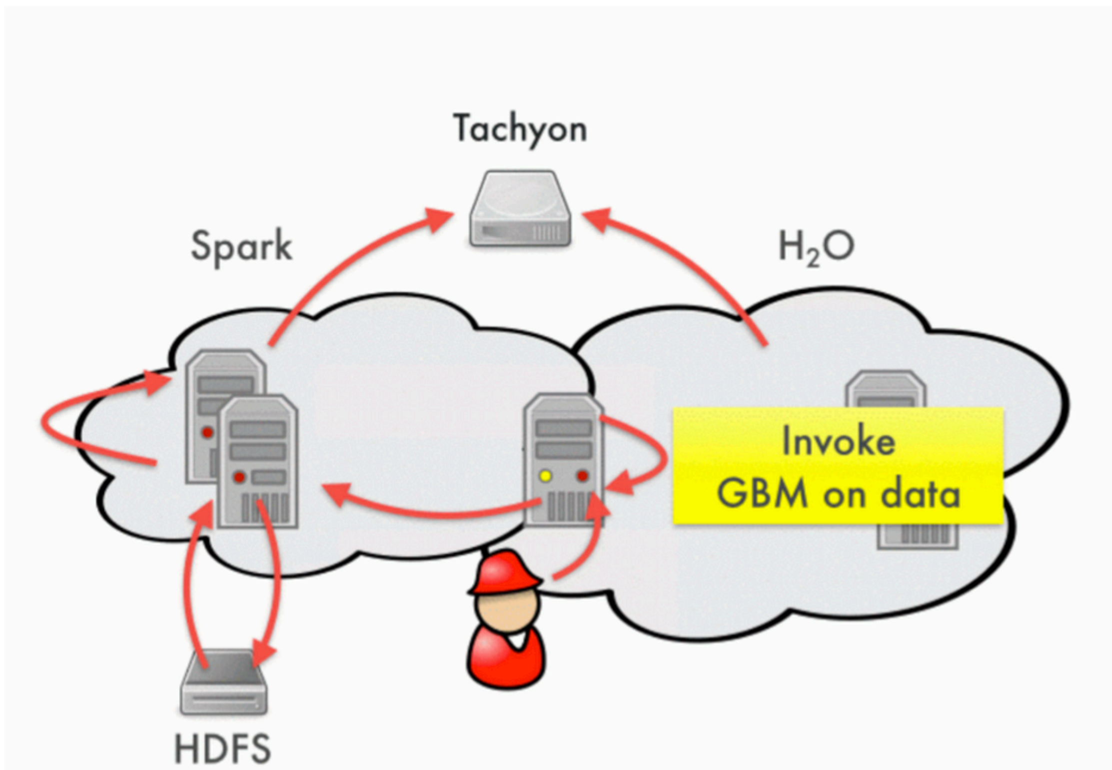
Thanks to Redhat!

Commercially supported by Atigeo™ and running in dozens of their customers' clusters

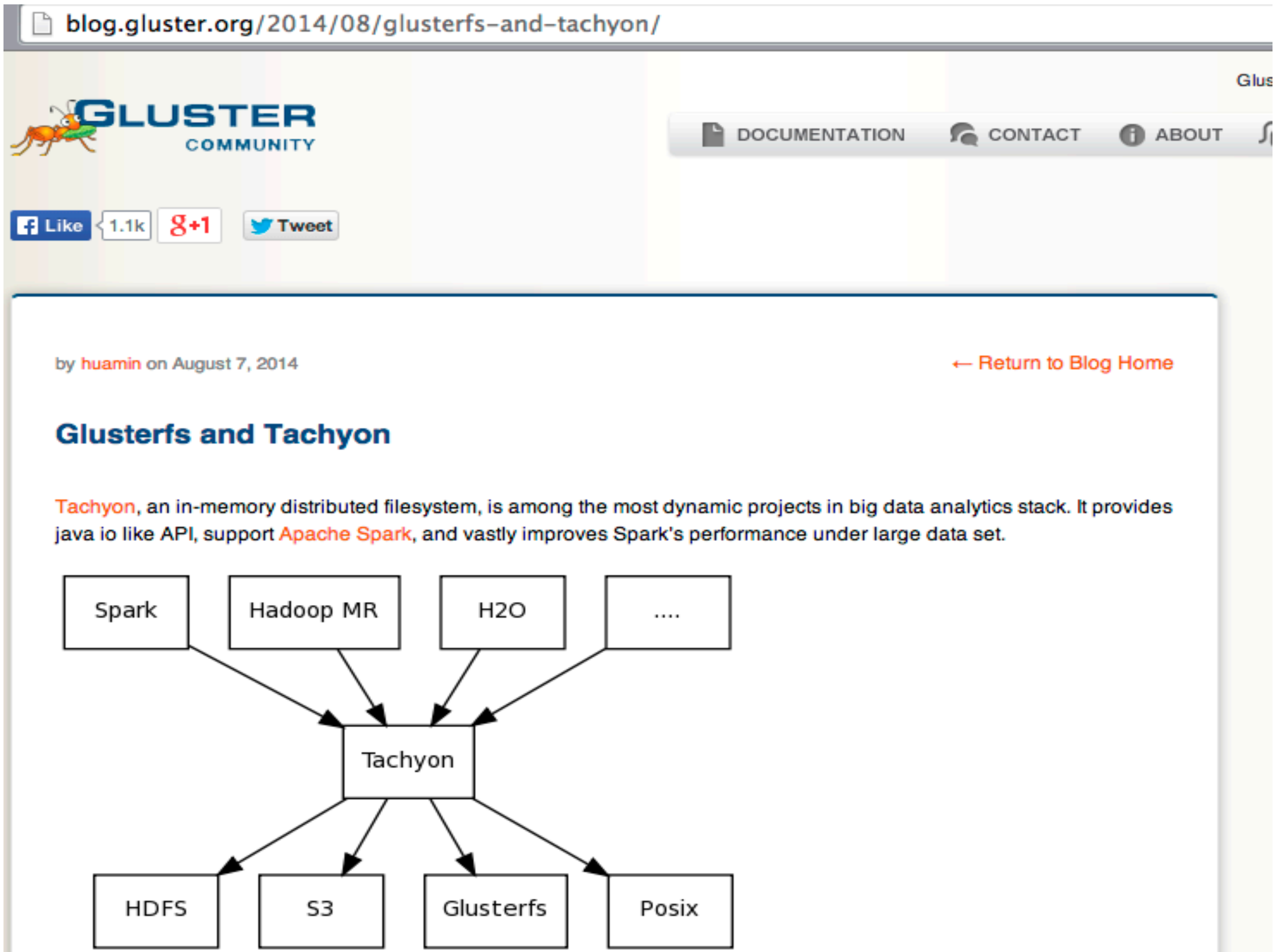
Tachyon is the Default Off-Heap Storage Solution for Spark

TACHYON in Spark H_2O

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.



Reaching wider communities: e.g. GlusterFS


blog.gluster.org/2014/08/glusterfs-and-tachyon/

GLUSTER COMMUNITY

DOCUMENTATION CONTACT ABOUT

Like 1.1k 8+1 Tweet

by huamin on August 7, 2014

← Return to Blog Home

Glusterfs and Tachyon

Tachyon, an in-memory distributed filesystem, is among the most dynamic projects in big data analytics stack. It provides java io like API, support Apache Spark, and vastly improves Spark's performance under large data set.

Under Filesystem Choices (Big Data, Cloud, HPC, Enterprise)

Outline

- Overview
  - Feature 1: Memory Centric Storage Architecture
  - Feature 2: Lineage in Storage
- Challenges
- Open Source
- **Future**

Short Term Roadmap (0.6 Release)

- Ceph Integration (Ceph Community)
- Hierarchical Local Storage (Intel)
- Performance Improvement (Yahoo)
- Multi-tenancy (AMPLab)
- Mesos Integration (Mesos Community)
- ***Many more*** from AMPLab and Industry Collaborators.

Features

- Memory Centric Storage Architecture
- Lineage in Storage (alpha)
- Hierarchical Local Storage
- Data Serving
- Scalable metadata management
- Different hardware
- More…
- Your Requirements?

Data Serving: An Example

Data Analytics Pipeline:

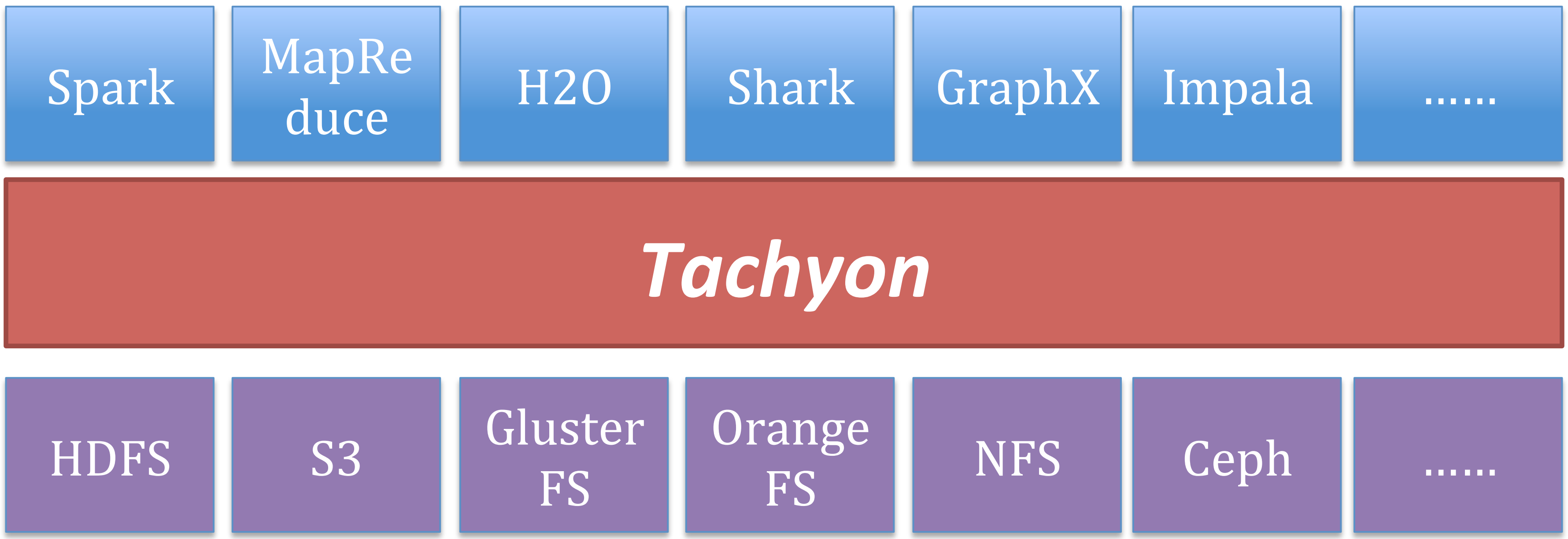
Query the results of batch jobs.

What do we need?

Sequential I/O + Random Access!

Tachyon Goal?

Better Assist Other Components



Welcome Collaboration!

Thanks!

Questions?

- *More Information:*
  - *Website: http://tachyon-project.org*
  - *Github: https://github.com/amplab/tachyon*
  - *Meetup: http://www.meetup.com/Tachyon*
- *Email: haoyuan@cs.berkeley.edu*