Interactive Data Analytics with Spark on *Tachyon* in Baidu

Bin Fan (Tachyon Nexus)
binfan@tachyonnexus.com

Xiang Wen (Baidu)
wenxiang@baidu.com

Dec 02 2015 @ Strata + Hadoop World, Singapore

Who Are We?

- Bin Fan
- Tachyon Project Contributor
- Software Engineer at Tachyon Nexus
- Xiang Wen
- From Baidu Big Data Department
- Senior Software Engineer at Baidu

TACHYON NEXUS

- Team consists of Tachyon creators, top contributors
- Series A ($7.5 million) from Andreessen Horowitz
- Committed to Tachyon Open Source Project
- www.tachyonnexus.com

Agenda

- **Tachyon Basics & New Features**
- Motivation
- Building an interactive data service
  - Spark + Tachyon
- Future Works

History of Tachyon

- Started at UC Berkeley AMPLab
  - From summer 2012
- Open sourced
  - April 2013 (two and half years ago)
  - Apache License 2.0
  - Latest Release: Version 0.8.2 (November 2015)

One of the Fastest Growing
Big Data Open Source Project

http://tachyon-project.org/community/

> 170 Contributors (v0.8)

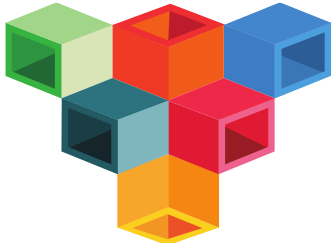
3x increment over the last year

TACHYON
NEXUS

One of the Fastest Growing Big Data Open Source Project

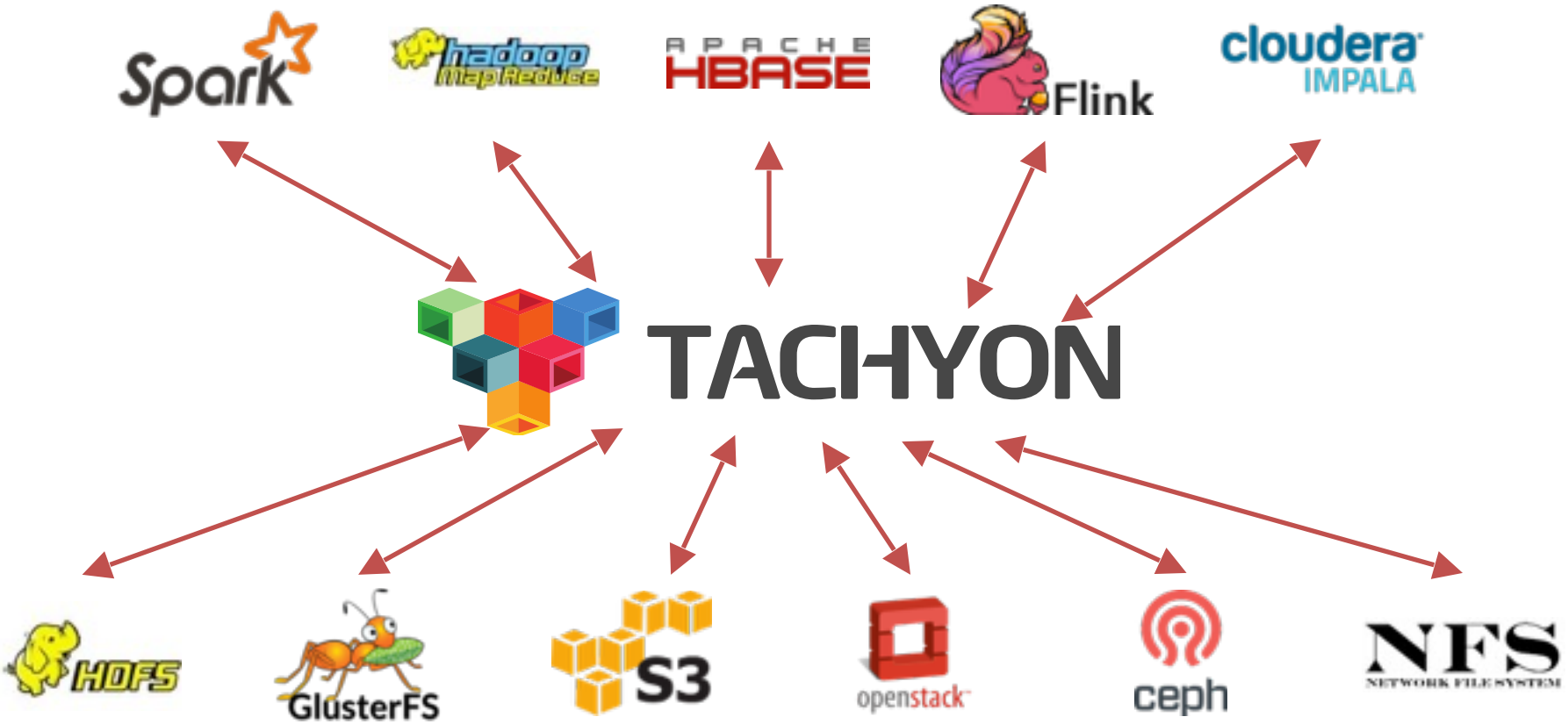
> 50 Organizations

TACHYON
NEXUS

What is  TACHYON

An Open Source Memory-Centric Distributed Storage System

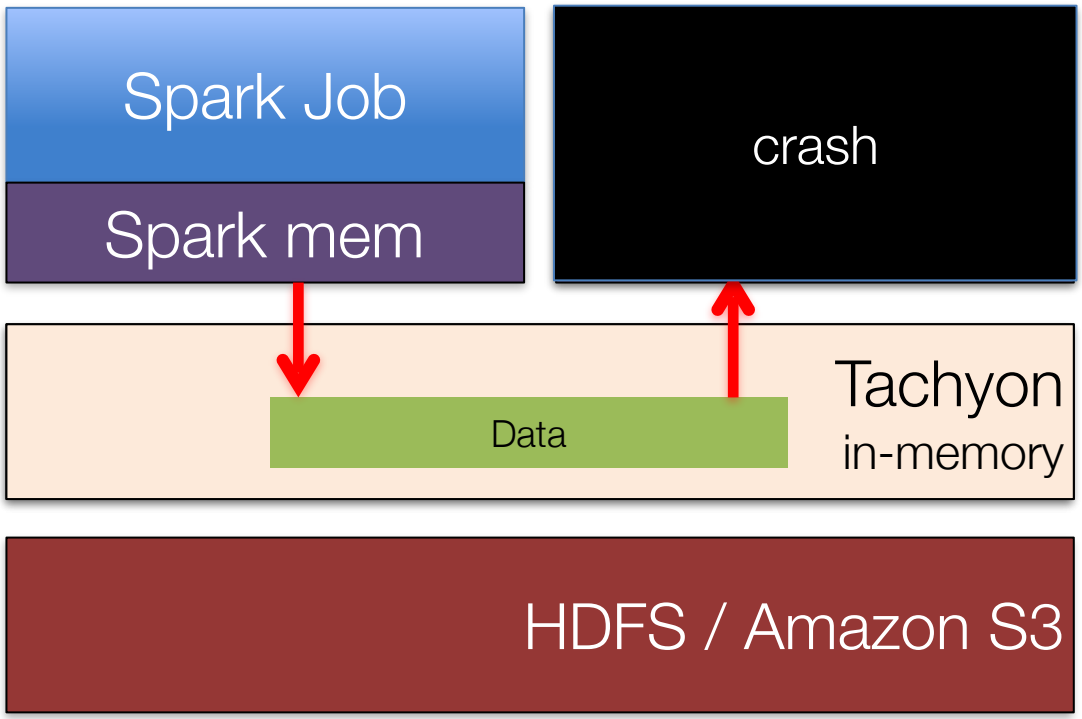
Tachyon Stack



Why Use Tachyon?

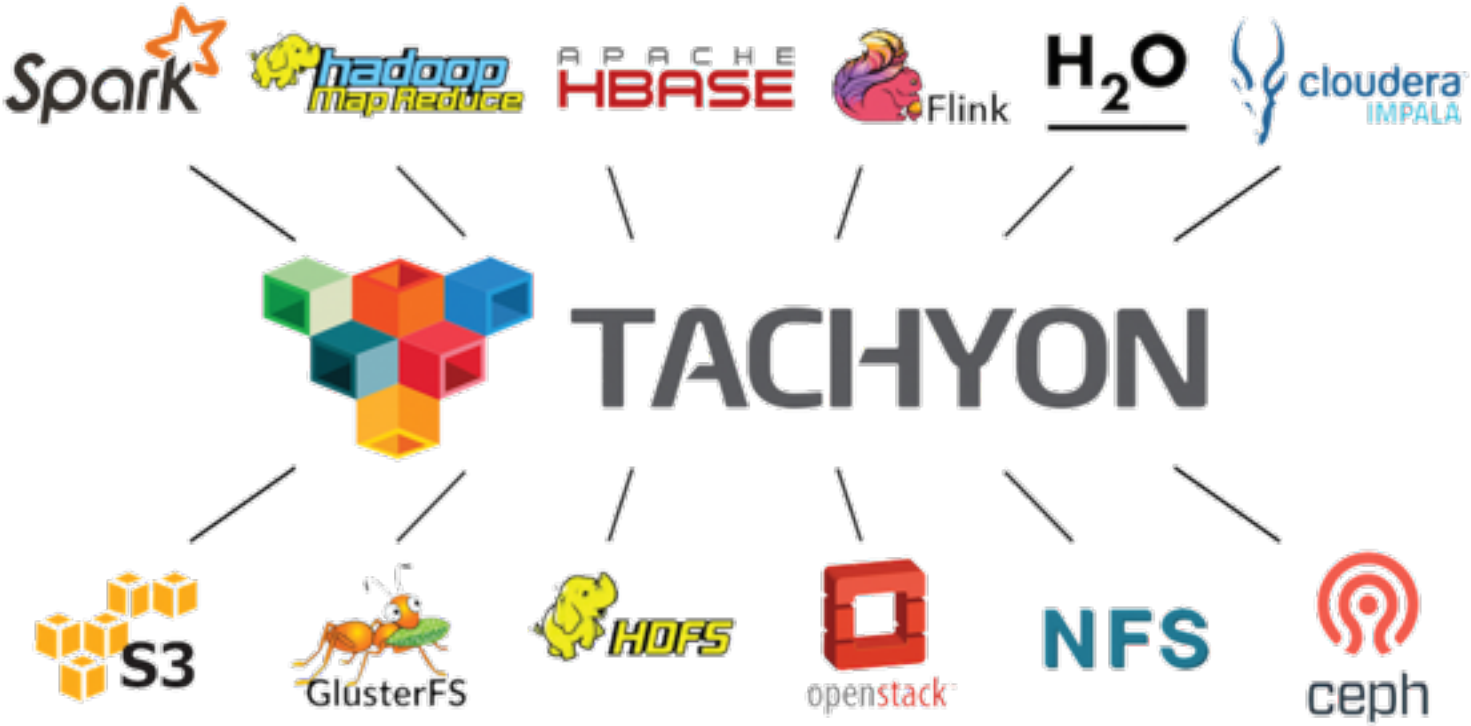
Memory-speed data sharing across jobs and frameworks

Data survive in memory after computation crashes



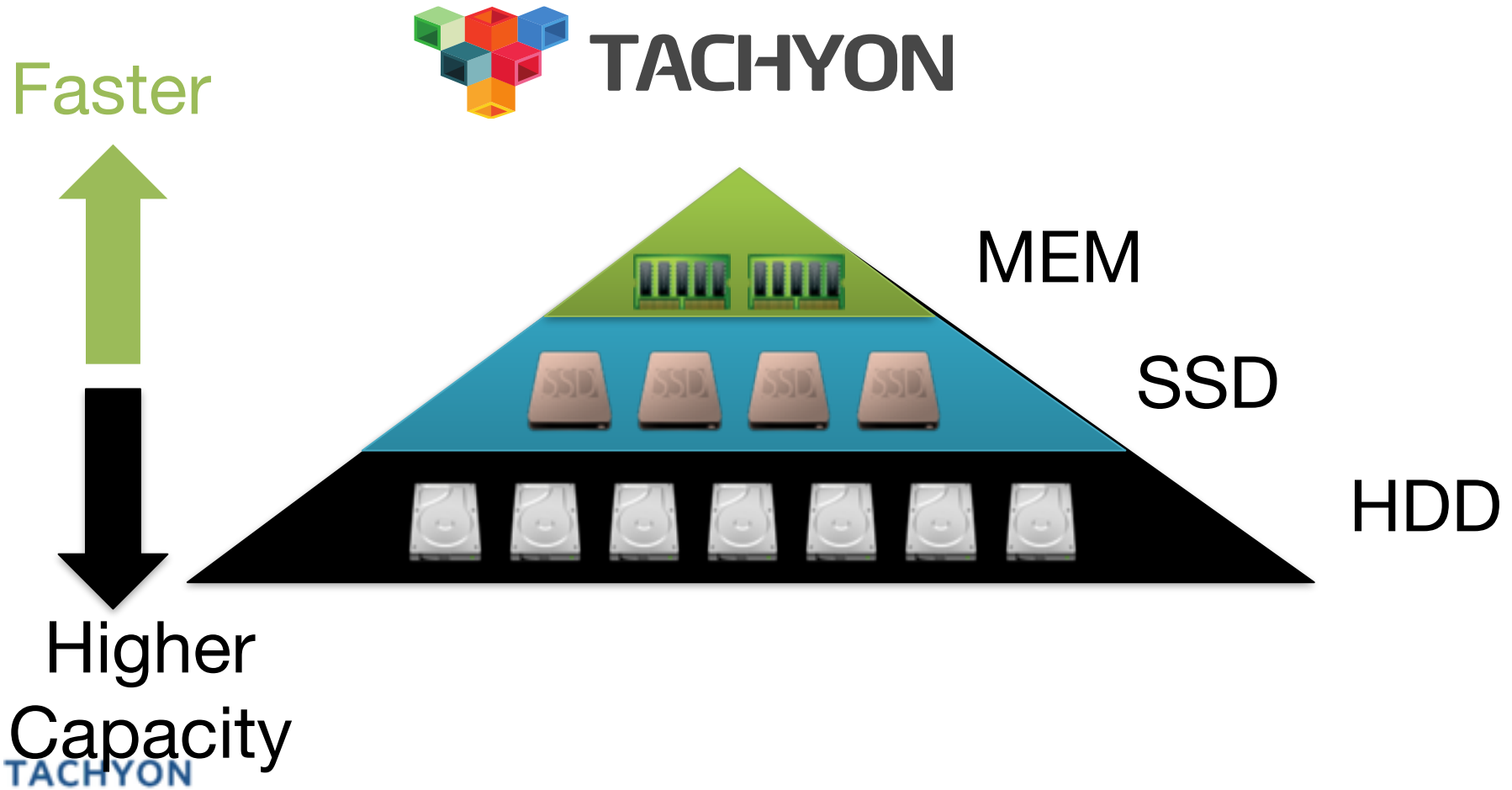
Off-heap storage, no GC

Enable Faster Innovation in Storage Layer



What if data size exceeds memory capacity?

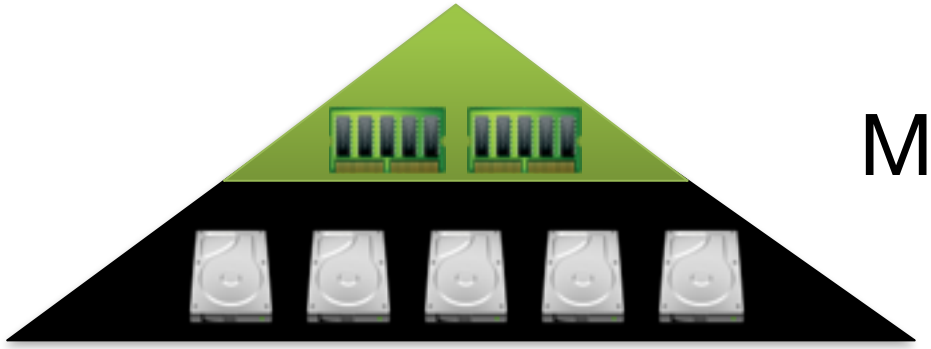
Tiered Storage: Tachyon Manages More Than DRAM



TACHYON NEXUS

Configurable Storage Tiers

MEM only



MEM + HDD

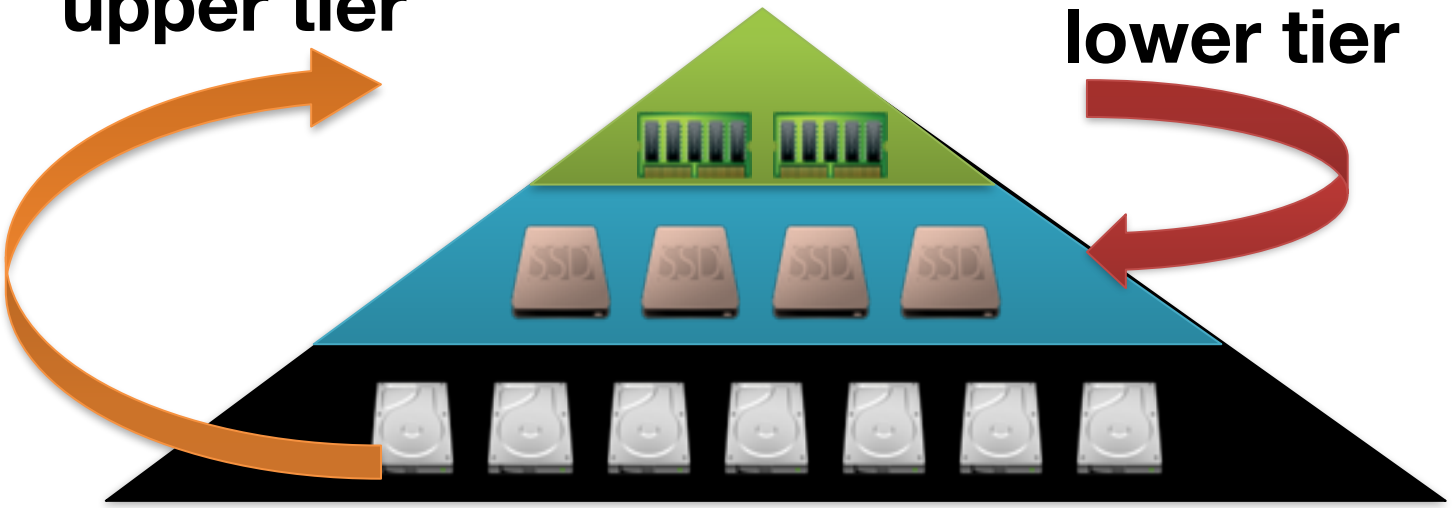
SSD only

TACHYON
NEXUS

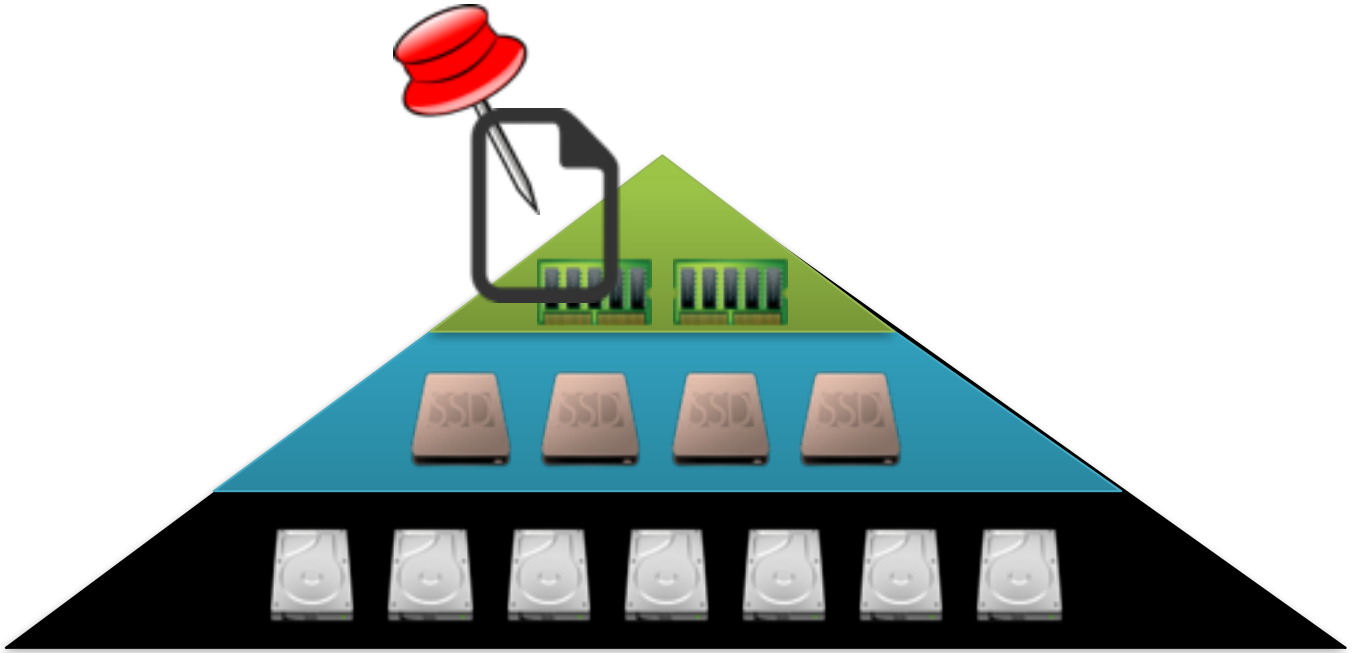
Pluggable Data Management Policy

Promote hot data to upper tier

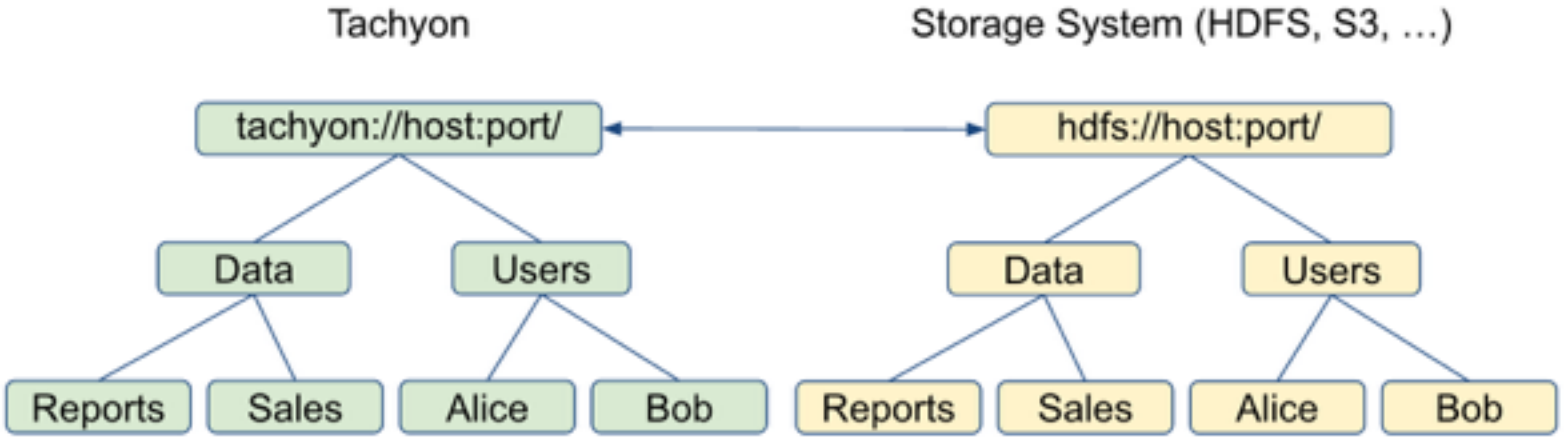
Evict stale data to lower tier



Pin Data in Memory

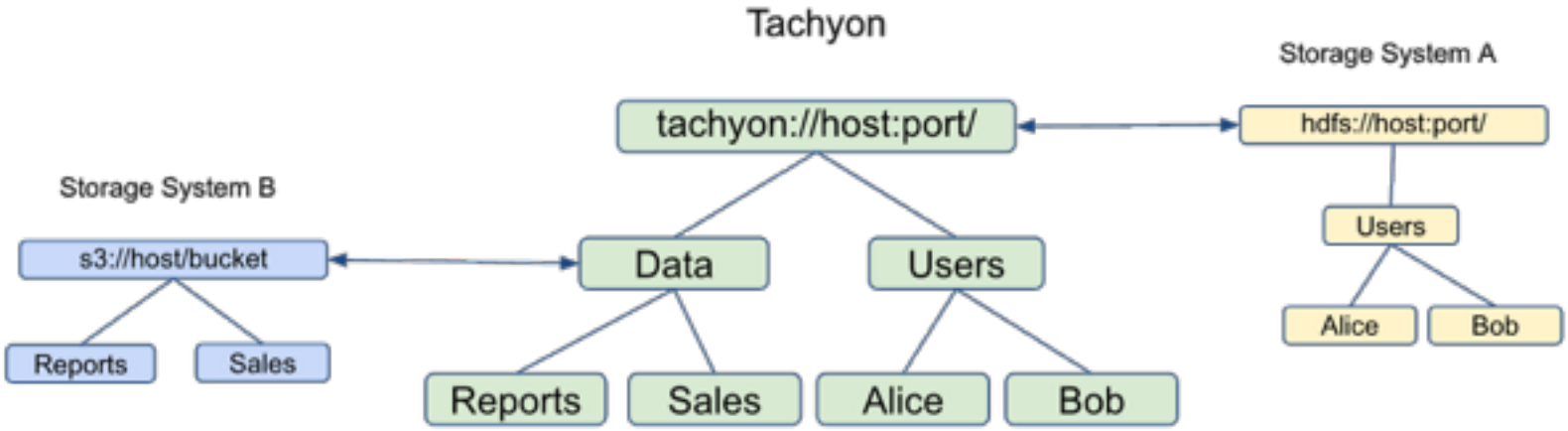


Transparent Naming



- Creation, renaming, and deletion of persisted Tachyon objects mapped to storage layer
- Tachyon paths are preserved in storage layer

Unified Namespace Across Under Storage Systems



- Unified namespace for multiple data sources
- Sharing of data across storage systems
- API for on-the-fly mounting / unmounting

More Features

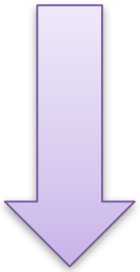
- Remote Write Support
- Easy deployment with **Mesos** and **YARN**
- Initial Security Support
- One Command Cluster Deployment
- Metrics Reporting for Clients, Workers, and Master

Rich Choice of Under Storage Supports

TACHYON
NEXUS

How Easy to Use Tachyon in Spark

```
scala> val file = sc.textFile(“hdfs://foo”)
```



```
scala> val file = sc.textFile(“tachyon://foo”)
```

Use Case: a SAAS Company

- Framework: Impala
- Under Storage: S3
- Storage Media: MEM + SSD
- 15x Performance Improvement

Use Case: a Biotechnology Company

- Framework: Spark & MapReduce
- Under Storage: GlusterFS
- Storage Media: MEM and SSD

When Tachyon Meets Baidu

30X Acceleration of our Big Data Analytics Workloads

- 100s nodes in deployment, > 1 PB storage space

Tachyon Summary

Master Address:	
Started:	05-11-2015 10:30:40:438
Uptime:	4 day(s), 16 hour(s), 3 minute(s), and 50 second(s)
Version:	0.6.0-baidu
Running Workers:	104

Cluster Usage Summary

Workers Capacity:	1247.63 TB
Workers Free / Used:	1243.32 TB / 4407.93 GB
UnderFS Capacity:	UNKNOWN
UnderFS Free / Used:	UNKNOWN / UNKNOWN

Storage Usage Summary

Storage Alias	Space Capacity	Space Used	Space Usage
MEM	1668.00 GB	1619.06 GB	97%Used
HDD	1246.00 TB	2788.87 GB	100%Free

TACHYON NEXUS

Agenda

- Tachyon Basics & New Features
- **Motivation**
- Building an interactive data service
  - Spark + Tachyon
- Future Works

Background

Data Warehouse

Hours or Days

Logs

Hours

Online Data Services

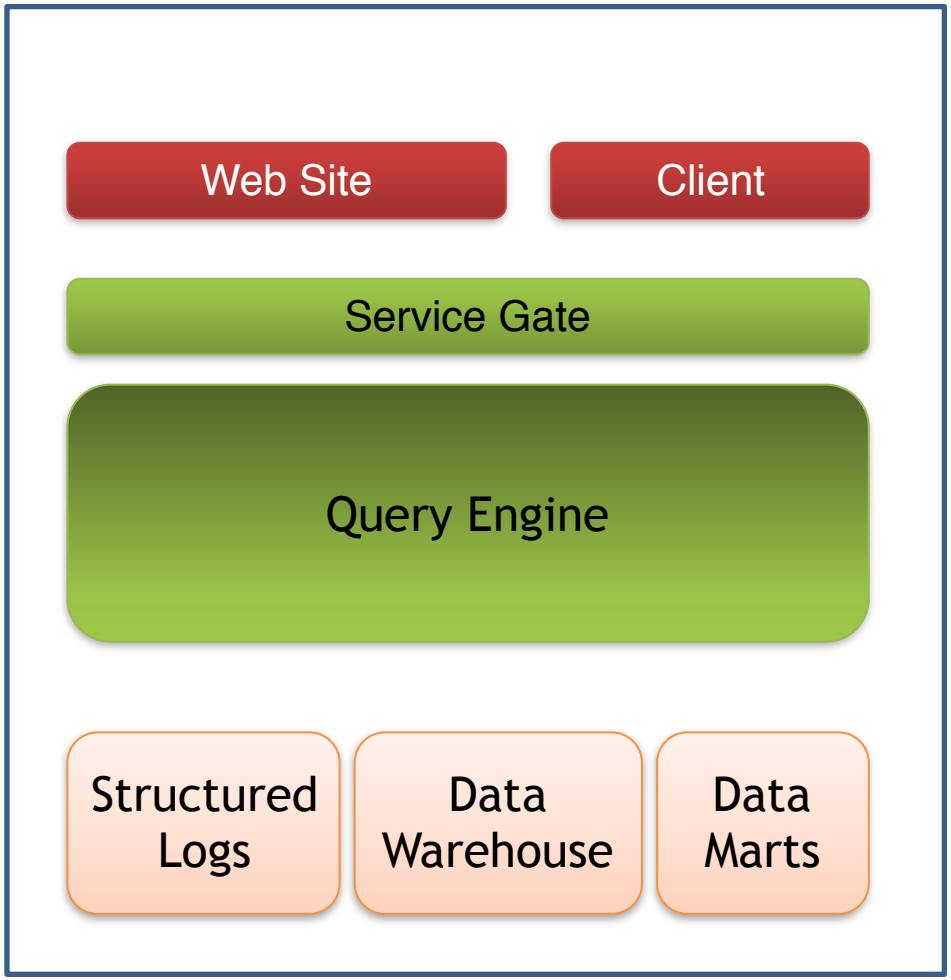
Frustrated data explorers

- Example:
  - John is a PM and he needs to keep track of the top user actions for a new feature
  - Based on the top actions of the day, he will perform additional analysis
  - But John is very frustrated that each query takes tens of minutes to finish

A dedicated service for data exploring

- Manages PBs of data
- Most queries within one minute

User Scenario



Have first try

```
select some_action, count(*)
from event_table
where event_day='20151123'
group by user_action
```

Try another way

```
select some_action, event_hour,  count(*)
from  event_table
where event_day='20151123'
group by user_action, event_hour
```

Not as expected!
See what happens in original log

```
select *
from event_log
where event_day='20151123' and
event_hour='01'
limit 10
```

Agenda

- Tachyon Basics & New Features
- Motivation
- **Building an interactive data service**
  - **Spark + Tachyon**
- Future Works

Choose Spark as compute solution

Service Gate

Hive

Map Reduce

4X Improvement but not good enough!

Service Gate

Meta Store

HiveQL | UDFs | SerDes

Spark SQL

Apache Spark

Compute Center

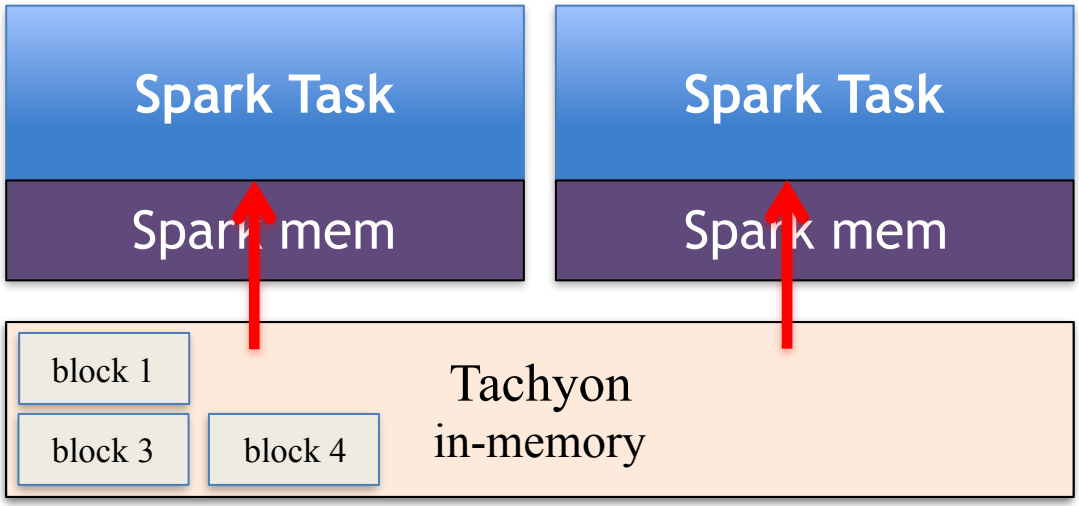
Data Center

Data Warehouse

BFS

Choose Tachyon as storage solution

Compute Center



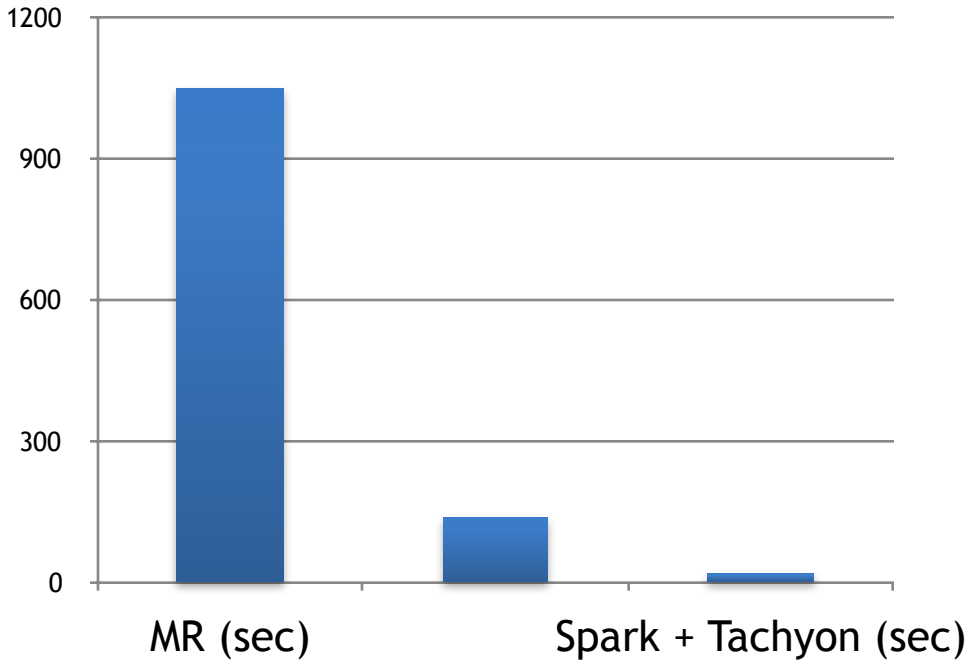
- Read from remote data center: ~ 100 ~ 150 seconds
- Read from Tachyon cluster local node: 10 ~ 15 sec
- Read from Tachyon machine local node: ~ 5 sec

Tachyon Brings 30X Speed-up !

Data Center

Baidu File System (BFS)

Overall Performance



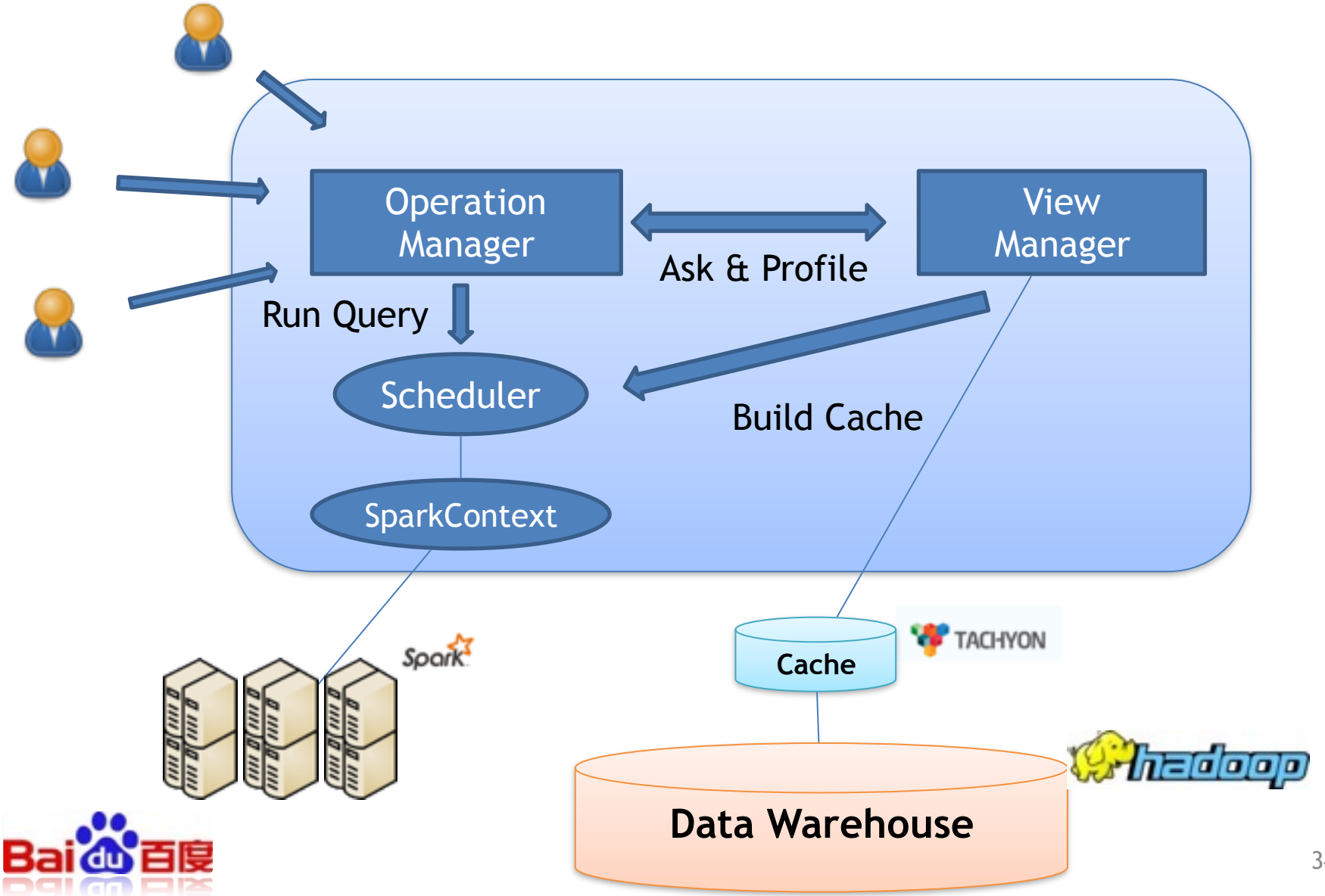
Setup:

1. Use MR to query 6 TB of data
2. Use Spark to query 6 TB of data
3. Use Spark + Tachyon to query 6 TB of data

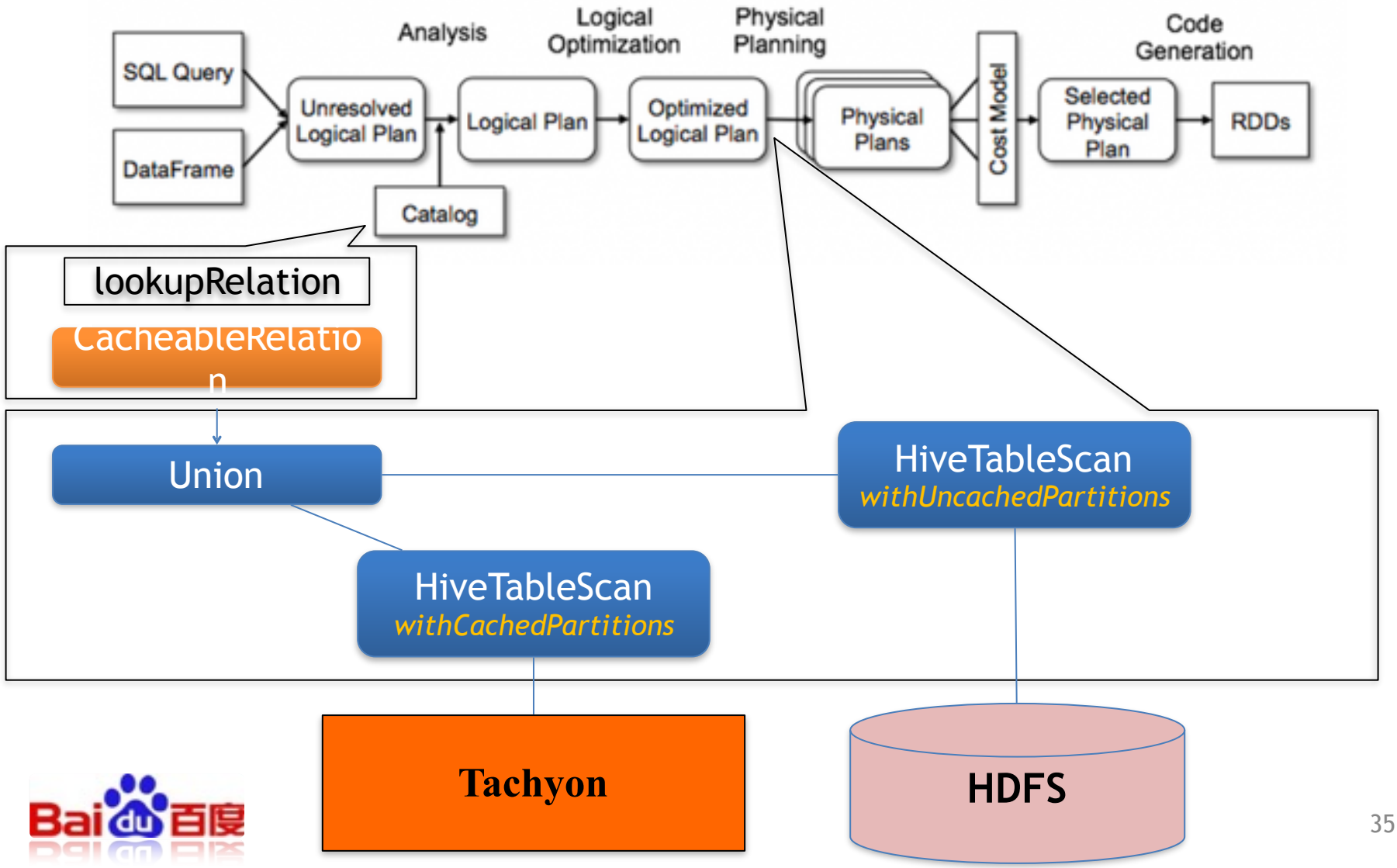
Results:

1. Spark + Tachyon achieves 50-fold speedup compared to MR

Architecture



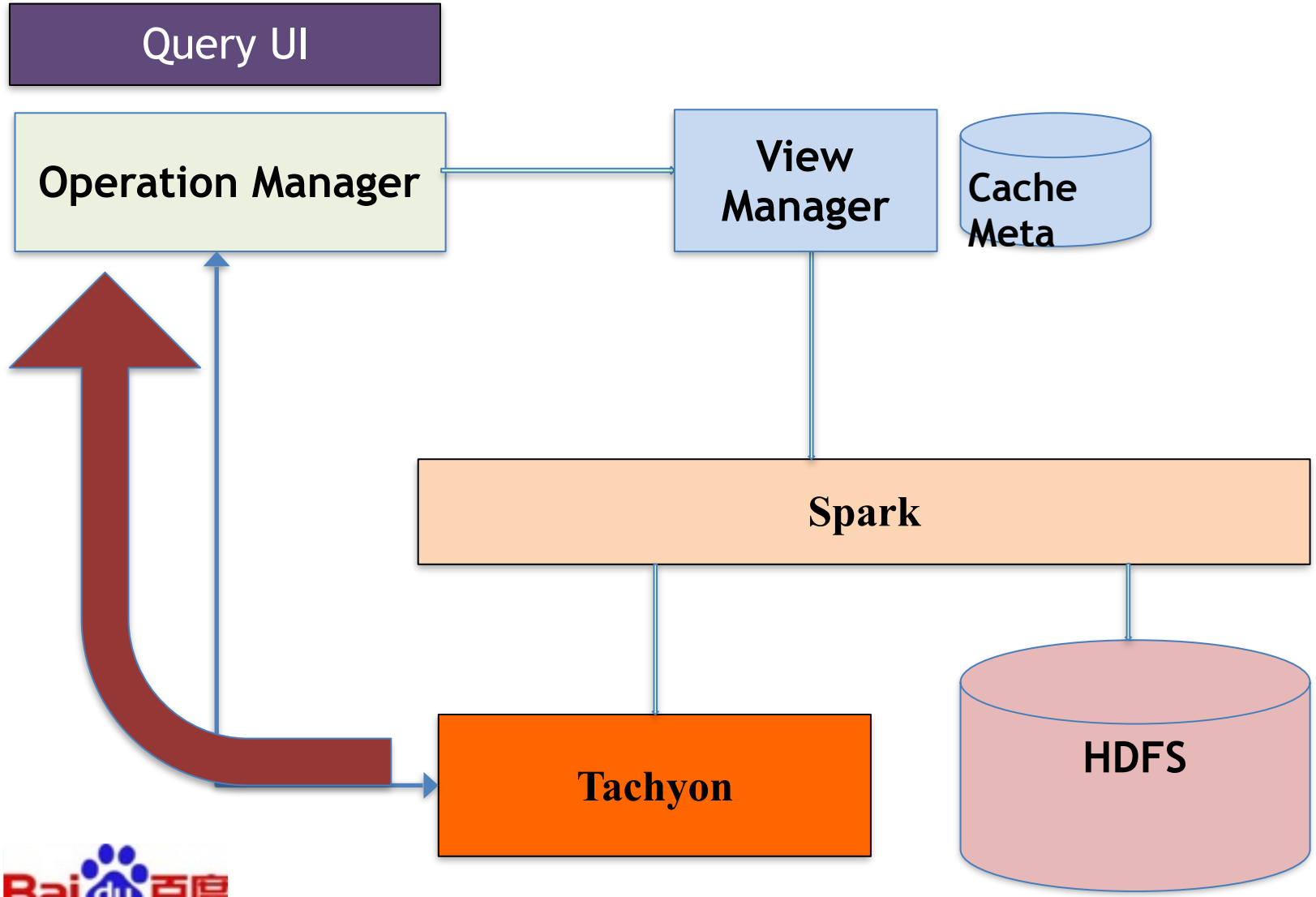
Catalyst helps to be ‘transparent’



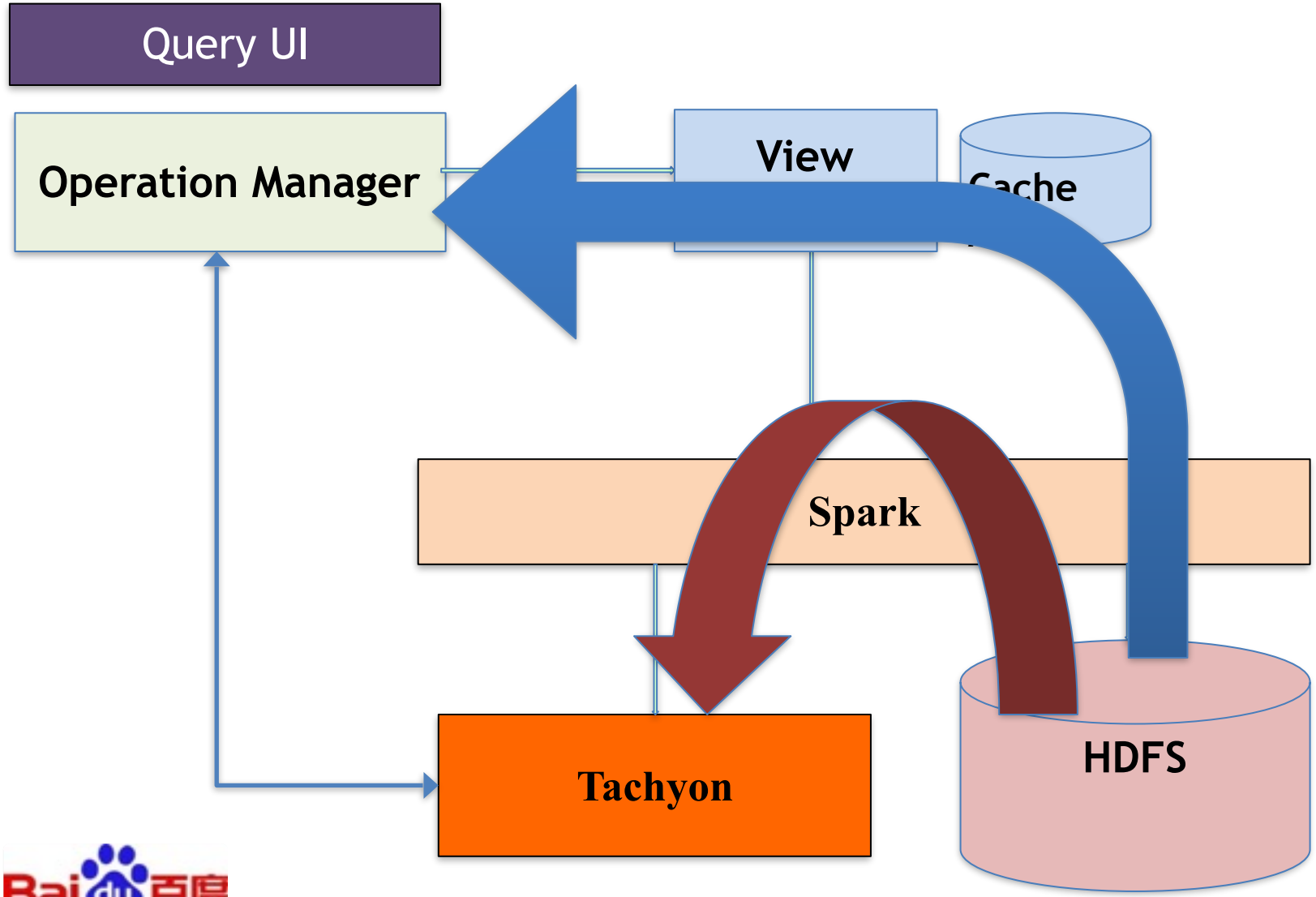
Cache Policy

- Prefetch
  - Fetch the views daily in advance when system is idle
  - The views fetched are based on the pattern of the past query history profiling, e.g. 3 months query logs
- On Demand caches
  - Fetch the views at runtime when system is serving regular queries.
  - Using machine learning to generate policy file monthly for views/tables
  - When a query is accessing some views, and parts of views match our pre-generated policy, those views will be cached at that time.

Hot Query: Cache Hit



Cold Query: Cache Miss



Daily Stats with Cache

- Daily
  - Table
    - Queries: 100 – 300
    - Hit Rate: ~40%
  - Partition
    - Queries: 80K – 120K
    - Hit Rate: ~40 – 50%
- Performance with Cache
  - avg 2 - 3 time faster than without Cache

Agenda

- Tachyon Basics & New Features
- Motivation
- Building an interactive data service
  - Spark + Tachyon
- **Future Works**

Improve Caching System

- As Extended Meta Service
  - Improve legacy schema/input-format
  - Load block meta into cache layer
  - Index / Materialized View
- Cost Based Caching/Optimizing
  - Better performance, hit rate & execution
  - Lower storage needs for cache layer

More User Scenario

- If John is data scientist
  - Need a way to construct dataset conveniently
  - Usually have many tries with same dataset
- An interactive system should help a lot
  - Spark is an ideal solution

Hardware assisted big data infrastructure

- Hardware
  - GPU
  - FPGA
- Applications
  - Accelerate common SQL and ML operators
  - TableScan && InputFormat && Serde
- Lower down the cost
  - 10 dollars for big data
  - 1 more dollar for interactive big data

Q&A