Memory-Speed Virtual Distributed Storage

Bin Fan (Founding Engineer @ Alluxio Inc)

TEAM

- Team consists of Alluxio creators and top committers

- Invested by ANDREESSEN HOROWITZ
- Committed to Alluxio Open Source
- http://www.alluxio.com

About Us

- **Bin Fan**
- **Alluxio PMC & Maintainer**
- **Founding Engineer @ Alluxio, Inc.**
- **Worked at Google, MSR**
- **CMU Parallel Data Lab**
- **binfan@alluxio.com**

Outline

- **Industry Trend & Alluxio overview**
- Features and Use Cases
  1. Off-heap memory to alleviate resource pressure
  2. Fast data sharing between jobs
  3. Accelerate access to remote storage
  4. Unified namespace
- Summary

PROBLEM

01 Emerging Compute Frameworks

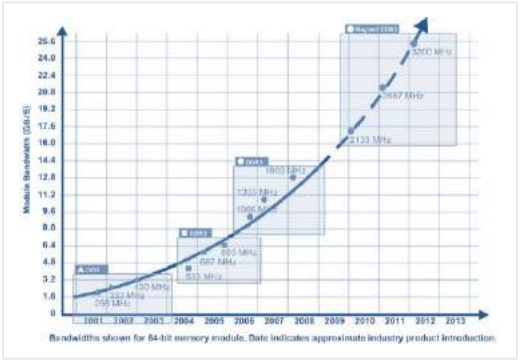
02 Exponential Data Growth

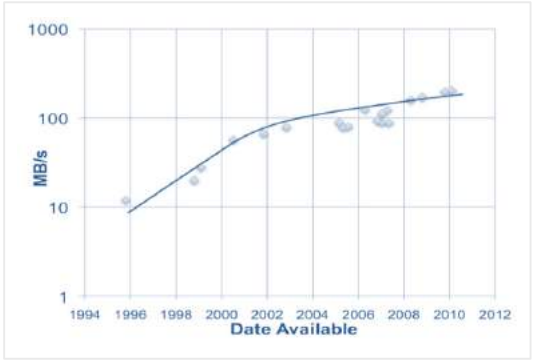
03 Increasingly Distributed Data

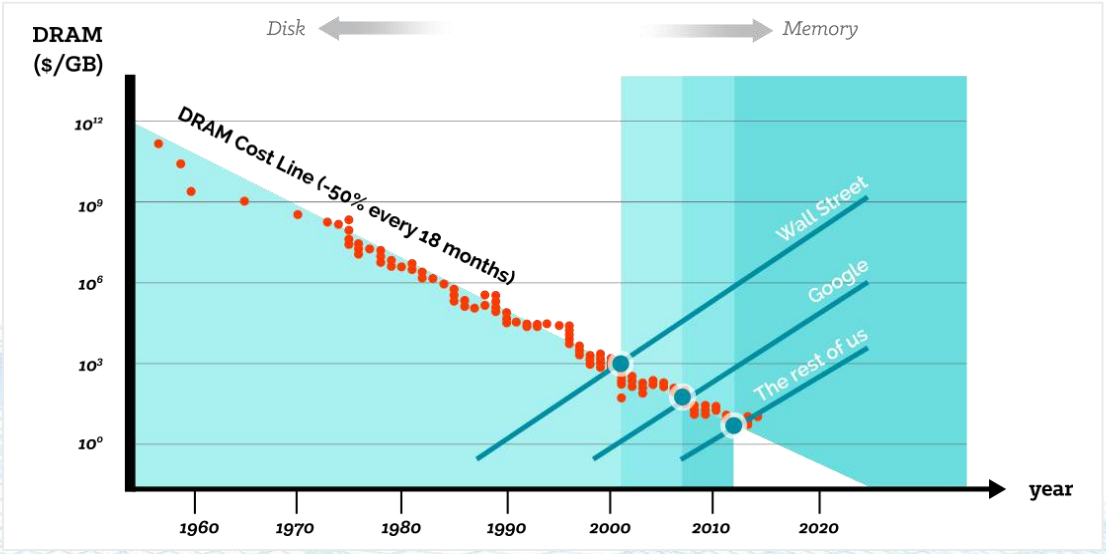
04 More Complex Data Movement

Resulting in a Proliferation of Disconnected Solutions

HARDWARE TRENDS







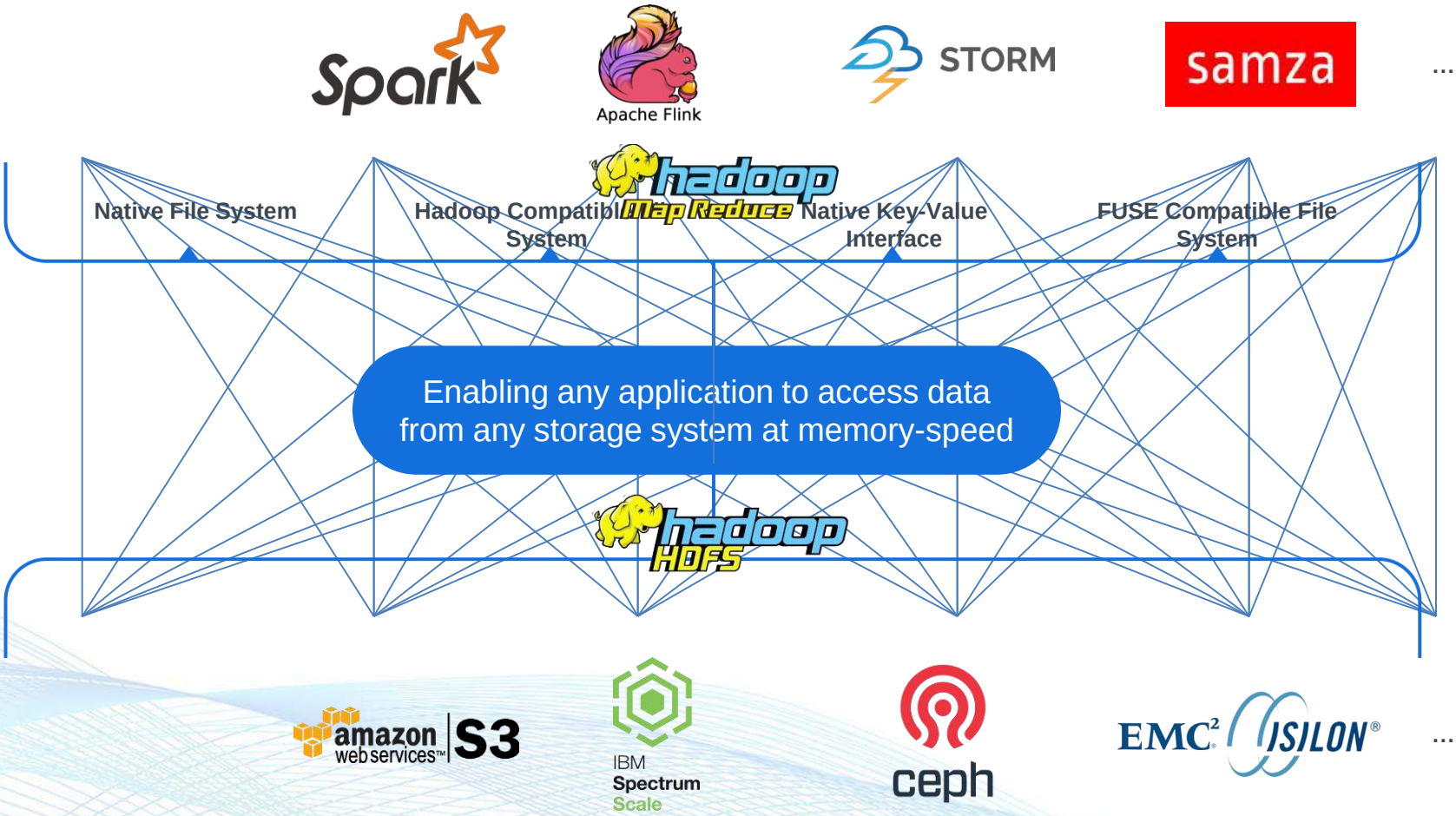
RAM throughput increasing exponentially

Disk throughput increasing slowly

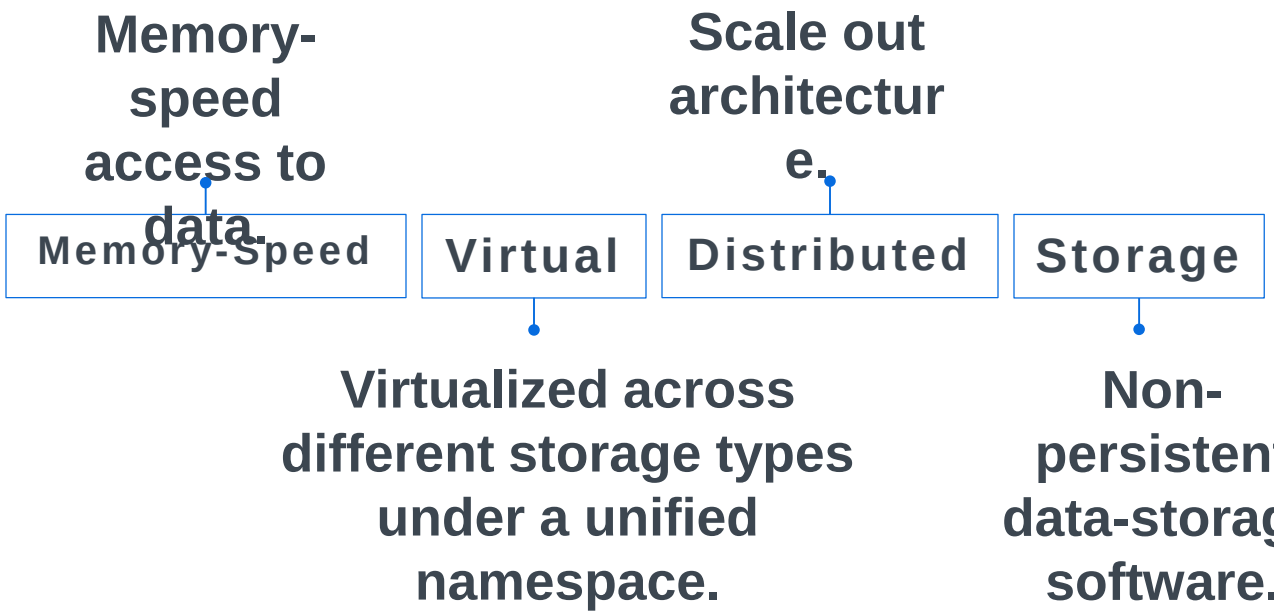
Memory prices halving every 18 months

Memory Locality is the Key!

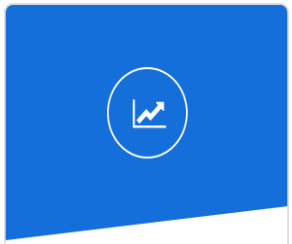
BIG DATA ECOSYSTEM WITH ALLUXIO



ATTRIBUTES

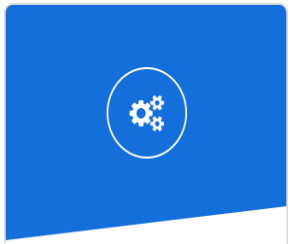


BENEFITS



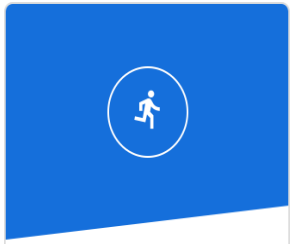
Performance

High performance data access



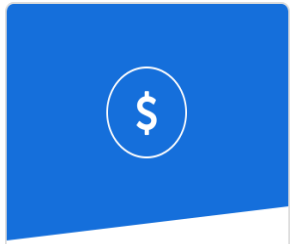
Flexibility

New workflows across existing data and access data in any storage system



Agility

Work with the compute and storage frameworks of your choice



Cost

Grow compute and storage systems independently

CASE STUDY

Accelerate Access to Remote Storage

Baidu File System

Spark

- 200+ nodes deployment
- 2+ petabytes of storage
- Mix of memory + HDD
- 1000-worker cluster

> **The performance was amazing. With Spark SQL alone, it took 100-150 seconds to finish a query; using Alluxio, where data may hit local or remote Alluxio nodes, it took 10-15 seconds.**
>
> - Shaoshan Liu, Baidu

RESULTS

- Data queries are now 30x faster with Alluxio
- 1000-worker Alluxio cluster run stably, providing over 50TB of RAM space
- By using Alluxio, batch queries usually lasting over 15 minutes were transformed into an interactive query taking less than 30 seconds

CASE STUDY

Share Data Across Jobs at Memory-Speed

Relational Database

Spark

- Reducing time from hours to seconds
- Solving issues of Teradata as under storage

Thanks to Alluxio, we now have the raw data immediately available at every iteration and we can skip the costs of loading in terms of time waiting, network traffic, and RDBMS activity.

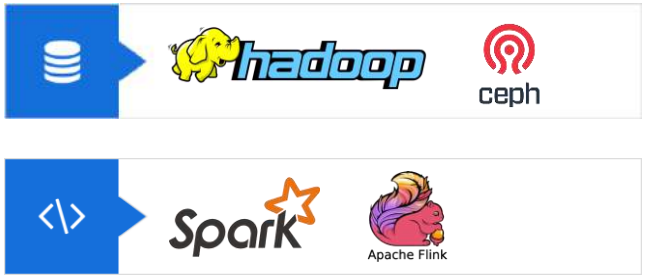
- Henry Powell, Barclays

RESULTS

- Barclays workflow iteration time decreased from hours to seconds
- Alluxio enabled workflows that were impossible before
- By keeping data only in memory, the I/O cost of loading and storing in Alluxio is now on the order of seconds

CASE STUDY

Transparently Manage Data Across Different Storage Systems



- 200+ nodes deployment
- 6 billion logs (4.5 TB) daily
- Mix of Memory + HDD

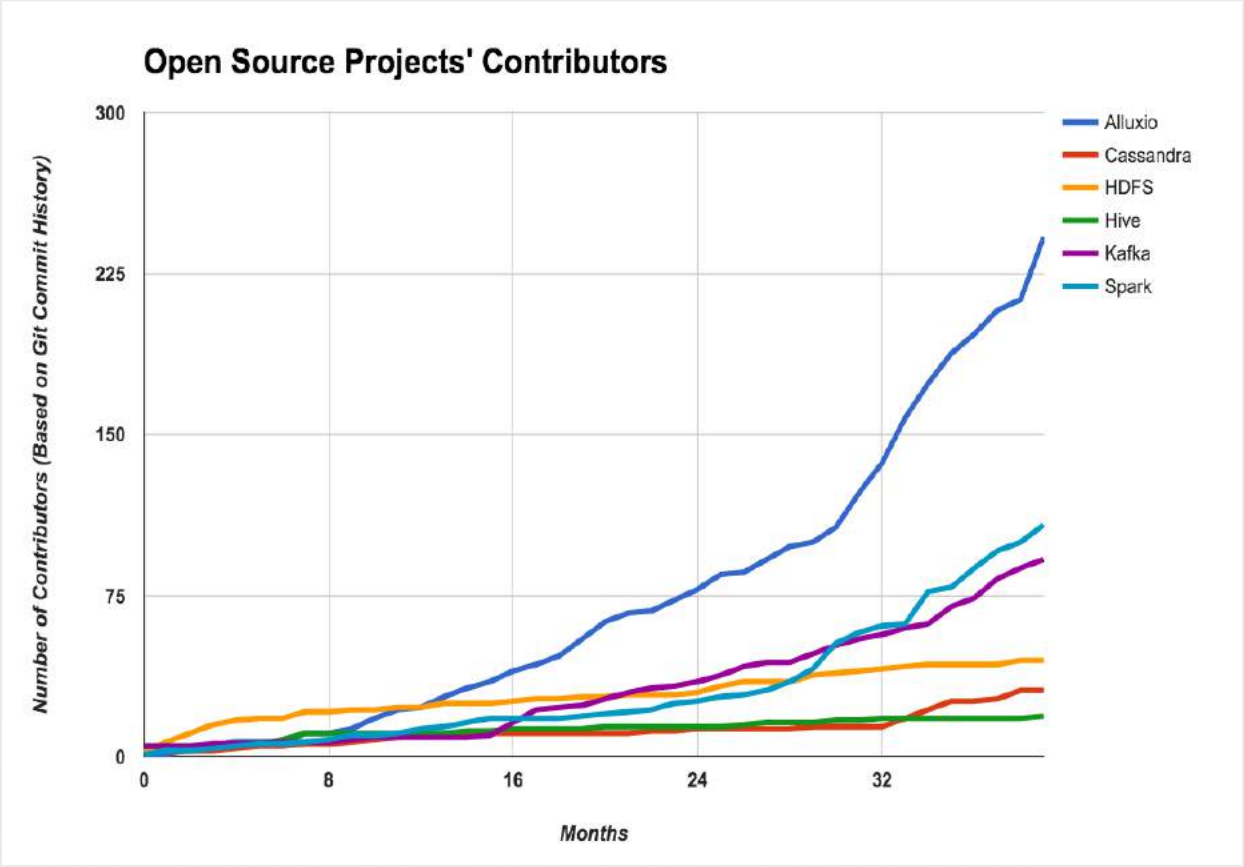
We've been running Alluxio in production for over 9 months, resulting in 15x speedup on average, and 300x speedup at peak service times.

- Xueyan Li, Qunar

RESULTS

- Alluxio's unified namespace enables different applications and frameworks to easily interact with their data from different storage systems
- Improved the performance of their system with 15x – 300x speedups
- Tiered storage feature manages various storage resources including memory, SSD and disk

OPEN SOURCE ALLUXIO



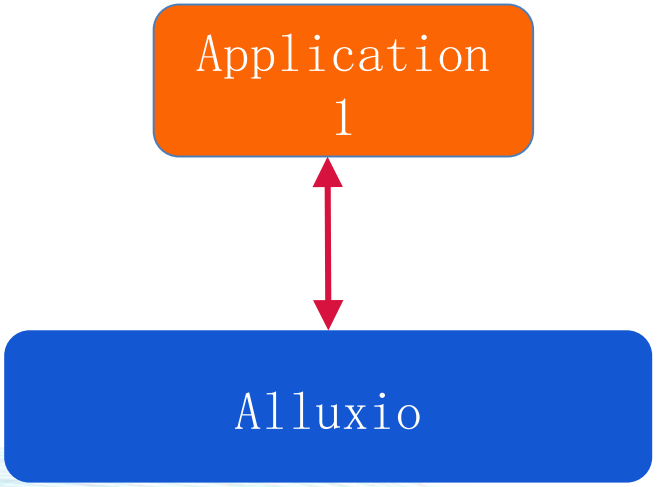
The fastest growing open-source project in the big data ecosystem

Currently over 300 contributors from over 100 organizations

Outline

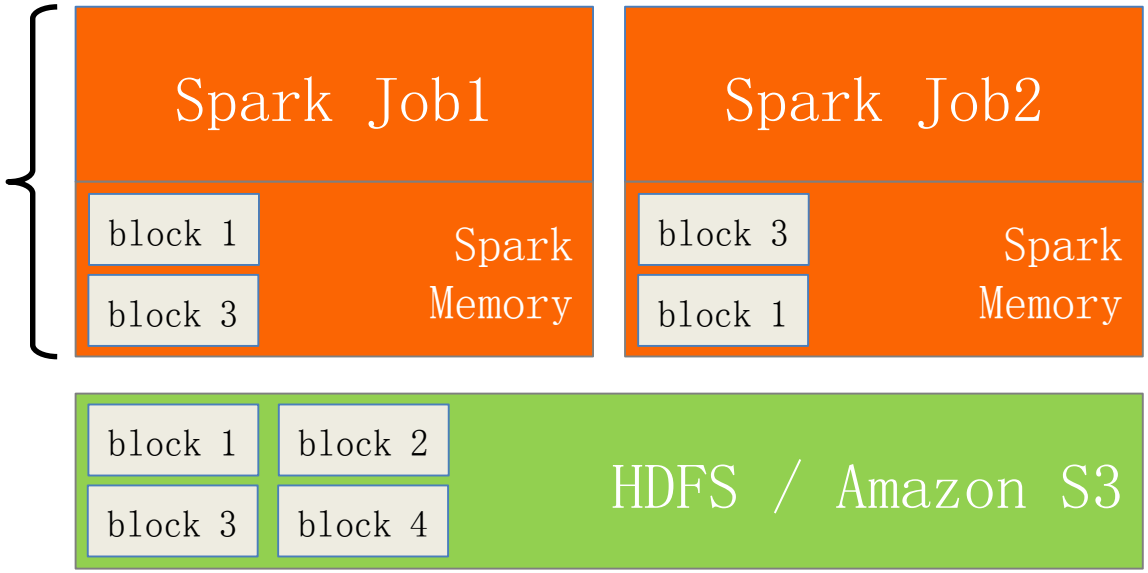
- Industry Trend & Alluxio overview
- **Features & Use Cases**
  1. Off-heap memory to alleviate resource pressure
  2. Fast data sharing between jobs
  3. Accelerate access to remote storage
  4. Unified namespace
- Summary

CASE 1:
Off-heap Memory Storage to Alleviate Resource Pressure



Consolidating Memory

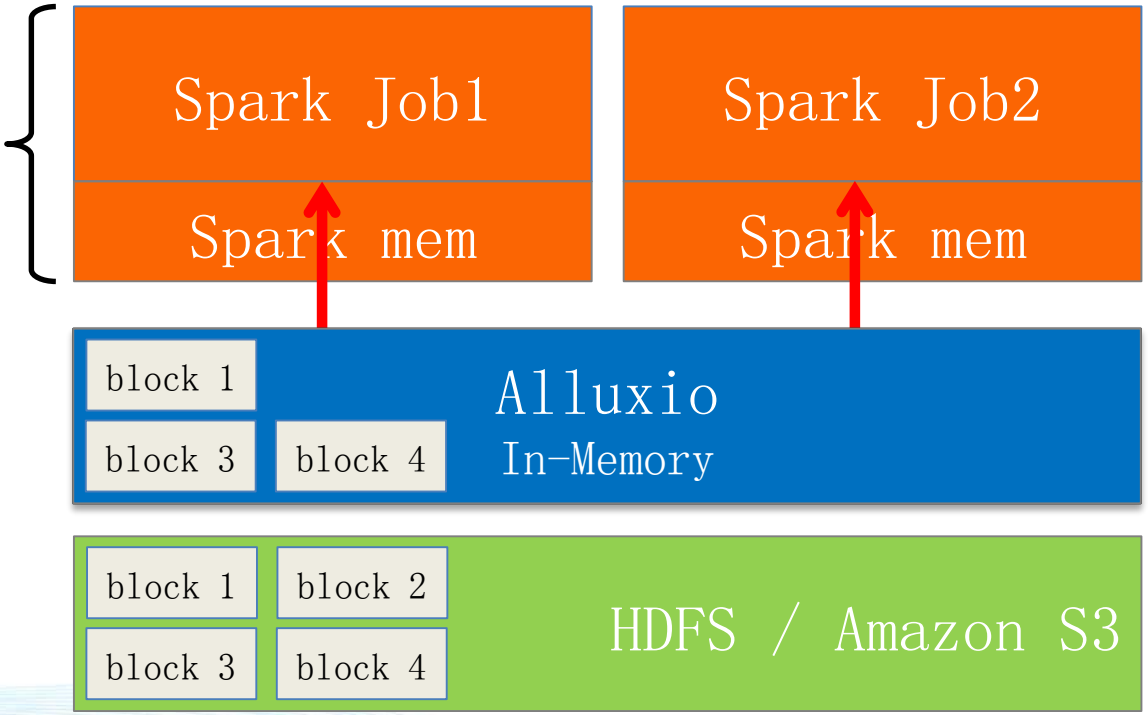
storage engine & execution engine same process



Data duplicated at memory-level

Consolidating Memory

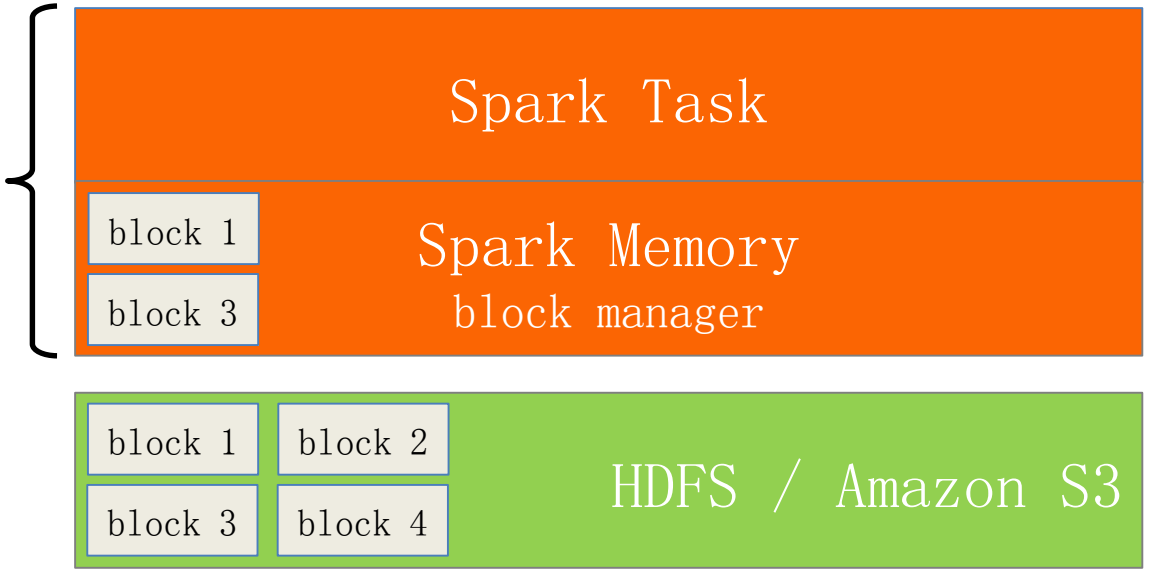
storage engine & execution engine same process



Data not duplicated at memory-level

Data Resilience during Crashes

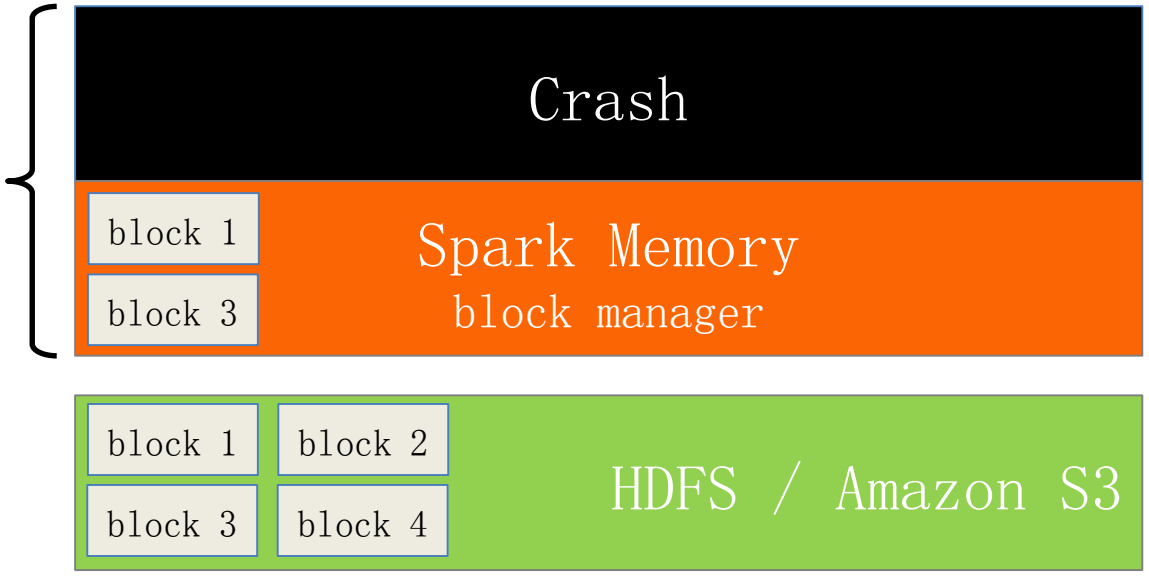
storage engine & execution engine same process



Process crash requires network and/or disk I/O to re-read the data

Data Resilience during Crashes

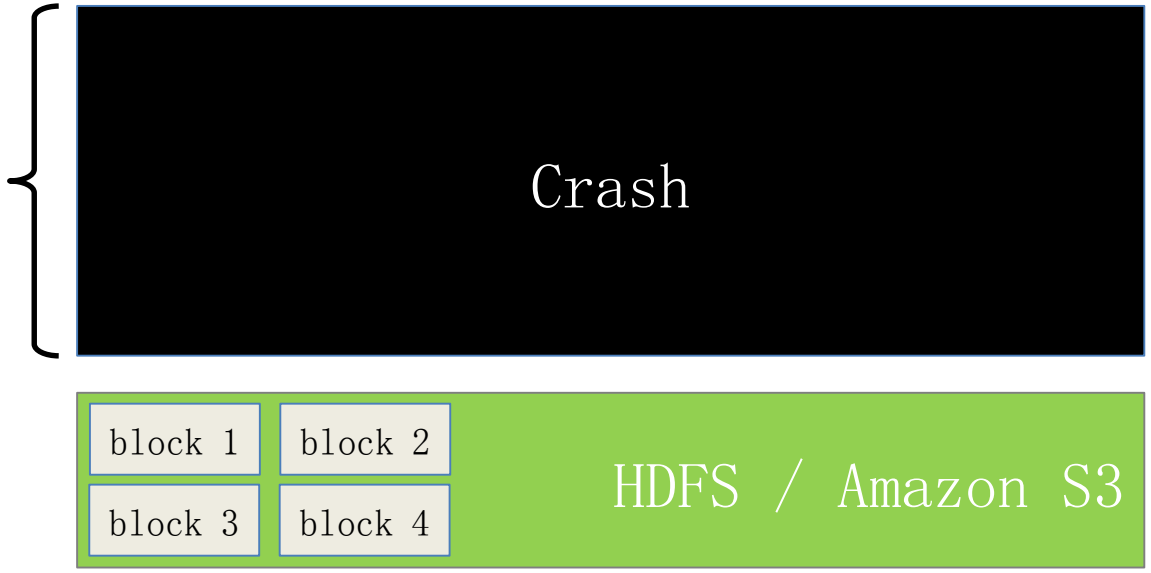
storage engine &
execution engine
same process



Process crash requires network and/or disk I/O to re-read the data

Data Resilience during Crashes

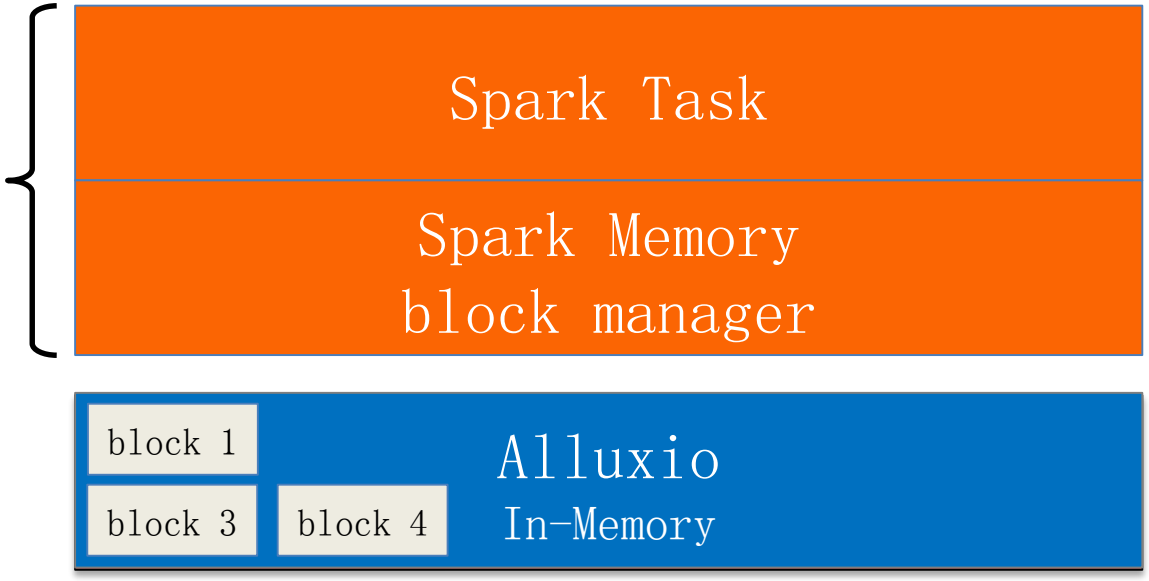
storage engine &
execution engine
same process



Process crash requires network and/or disk I/O to re-read the data

Data Resilience during Crashes

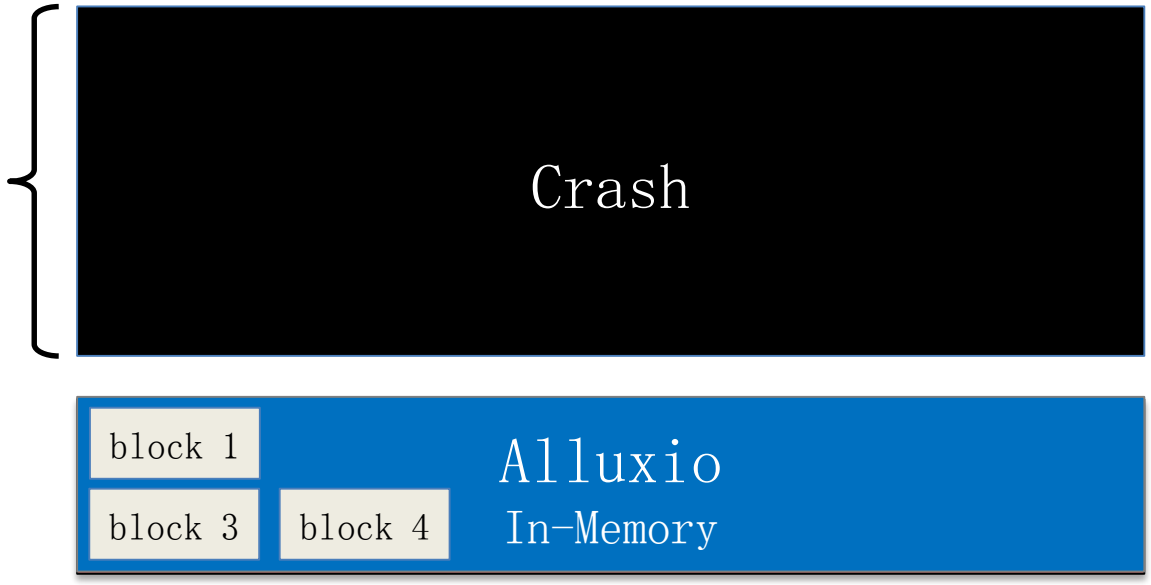
storage engine & execution engine same process



Process crash only needs memory I/O to re-read the data

Data Resilience during Crashes

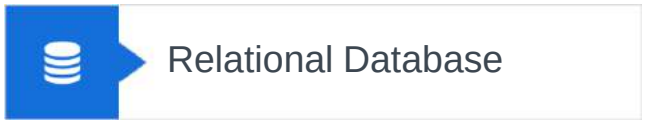
storage engine &
execution engine
same process



Process crash only needs memory I/O to re-read the data

CASE STUDY

Share Data Across Jobs at Memory Speed



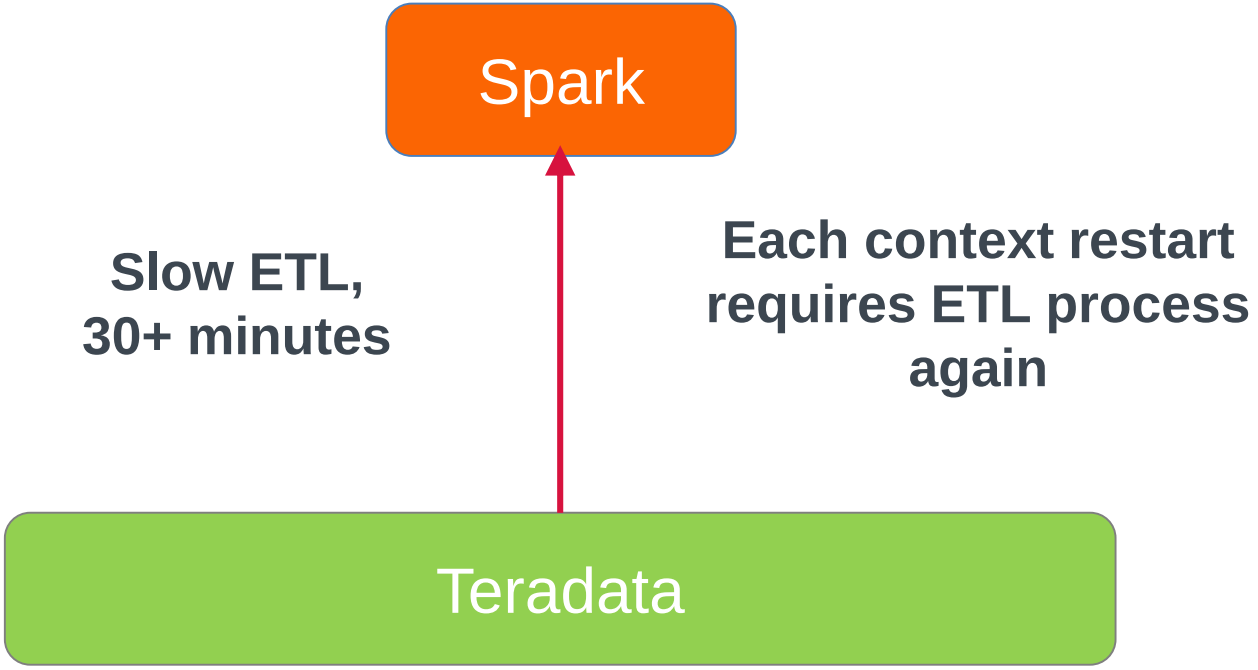
- Reducing time from hours to seconds
- Solving issues of Teradata as under storage

> **Thanks to Alluxio, we now have the raw data immediately available at every iteration and we can skip the costs of loading in terms of time waiting, network traffic, and RDBMS activity.**
>
> - Henry Powell, Barclays

RESULTS

- Barclays workflow iteration time decreased from hours to seconds
- Alluxio enabled workflows that were impossible before
- By keeping data only in memory, the I/O cost of loading and storing in Alluxio is now on the order of seconds

Barclays Previous Architecture

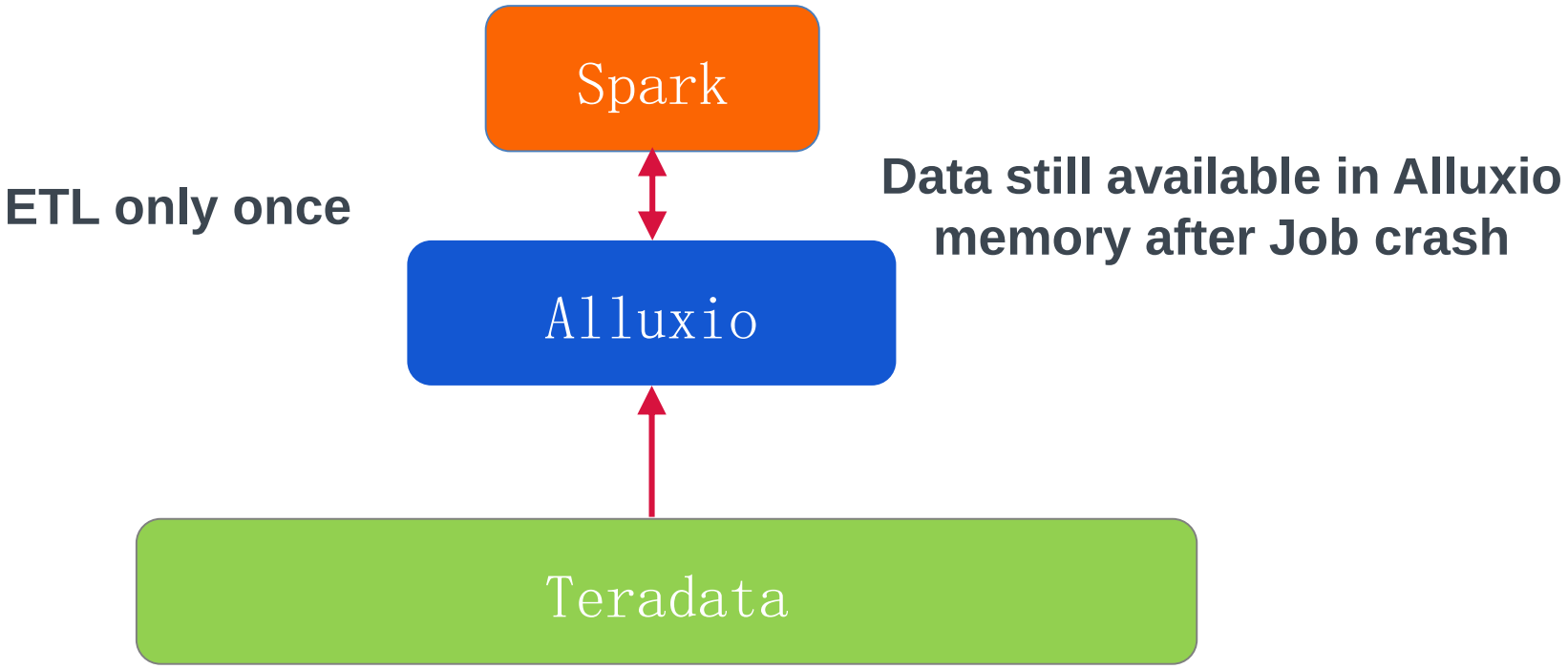


Issues With Existing Architecture

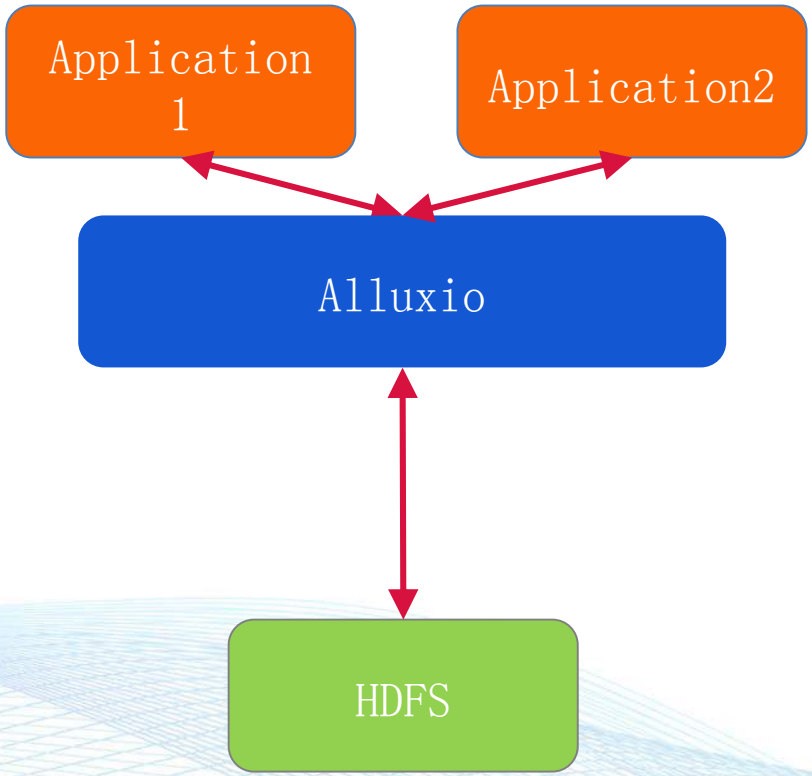
The main problem with our existing methodology is that the Spark cache is volatile across different jobs. Even though Spark provides a cache functionality, every time we restart the context, update the dependency jars or re-submit the job, the loaded data is dropped from the memory and the only way to restore it is to reload it from the central warehouse.

https://dzone.com/articles/Accelerate-In-Memory-Processing-with-Spark-from-Hours-to-Seconds-With-Tachyon

Barclays Architecture with Alluxio

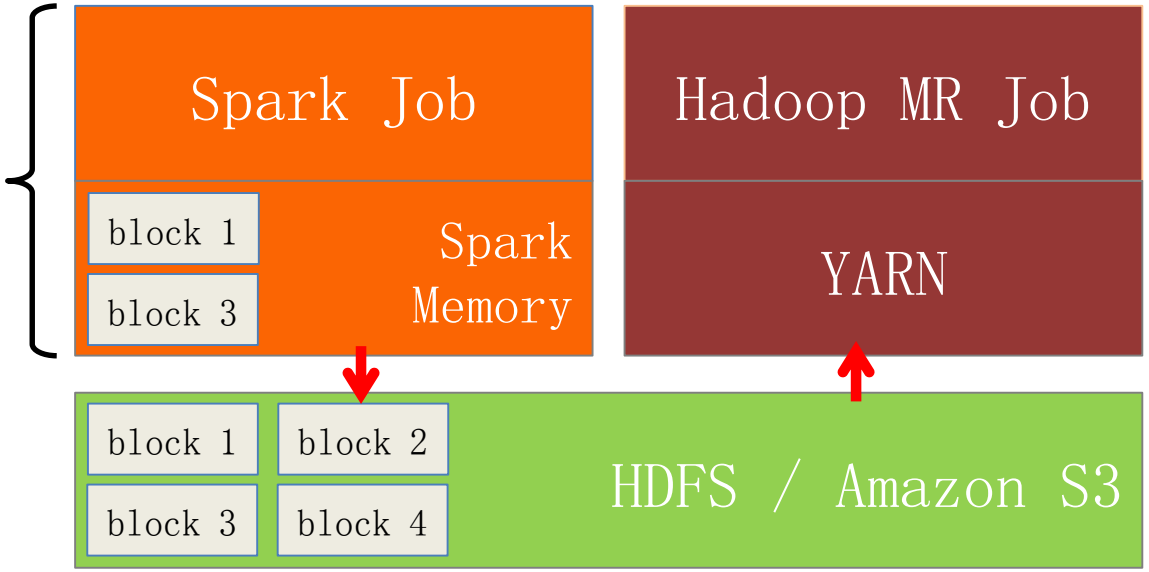


CASE 2:
Fast Data sharing between jobs



Data Sharing Between Frameworks

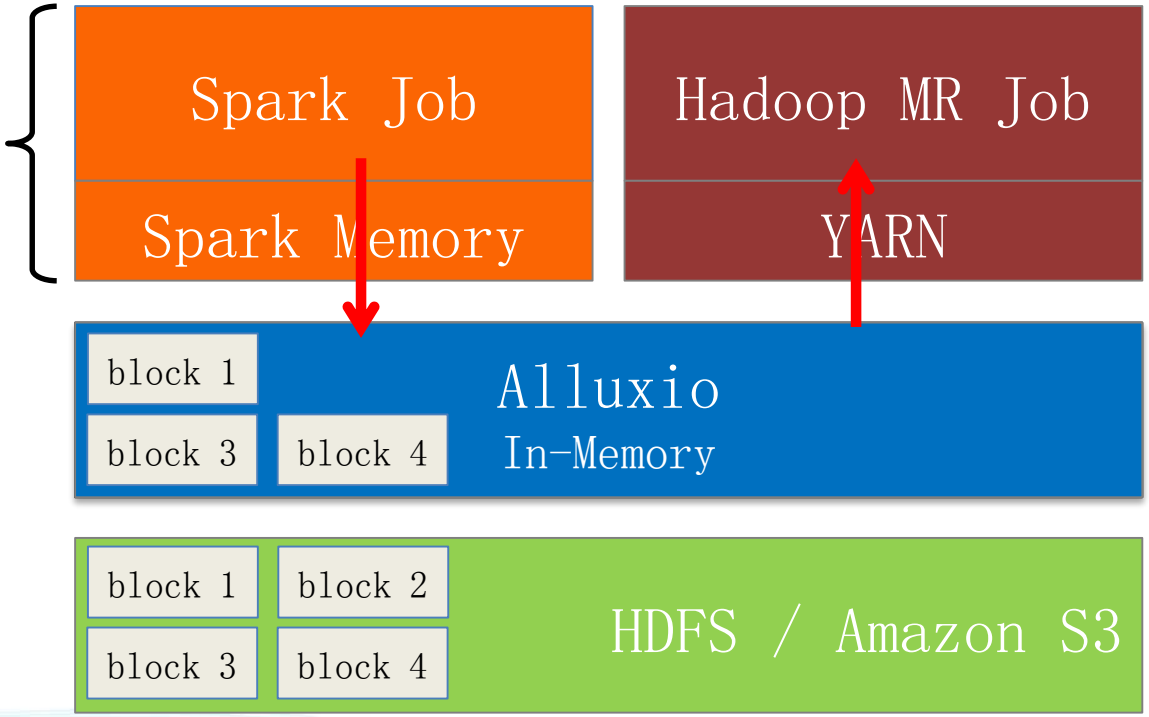
storage engine & execution engine same process



Inter-process sharing slowed down by network and/or disk I/O

Data Sharing Between Frameworks

storage engine & execution engine same process



Inter-process sharing can happen at memory speed

CASE STUDY

Sharing Across Different Storage Compute Frameworks

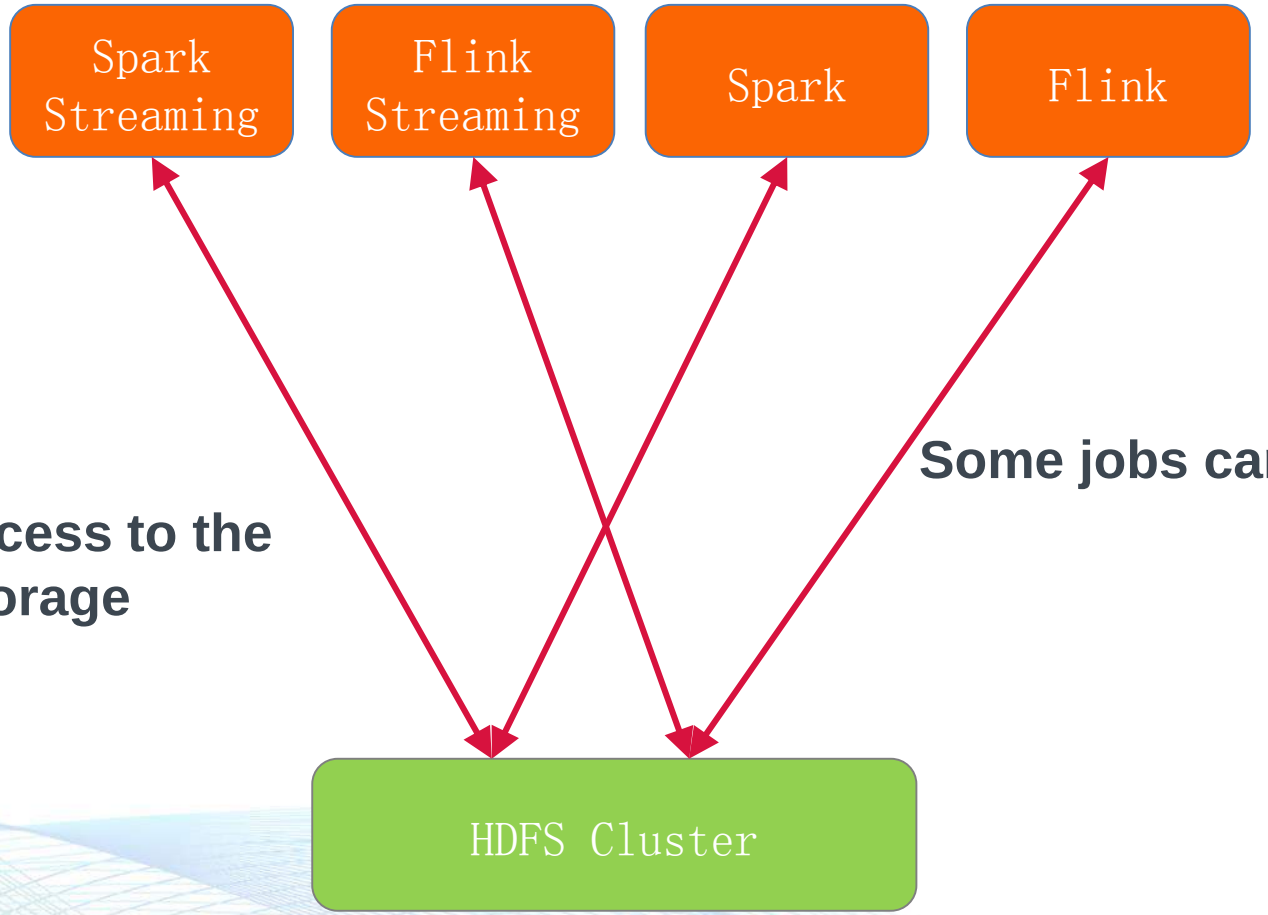
- 6 billion logs (4.5 TB) daily
- Mix of Memory + HDD

> **We've been running Alluxio in production for over 9 months, resulting in 15x speedup on average, and 300x speedup at peak service times.**
>
> - Xueyan Li, Qunar

RESULTS

- Multiple computing frameworks can share data at memory speed with Alluxio
- Improved the performance of their system with 15x – 300x speedups
- Alluxio provides various user-friendly APIs, which simplifies the transition to Alluxio.

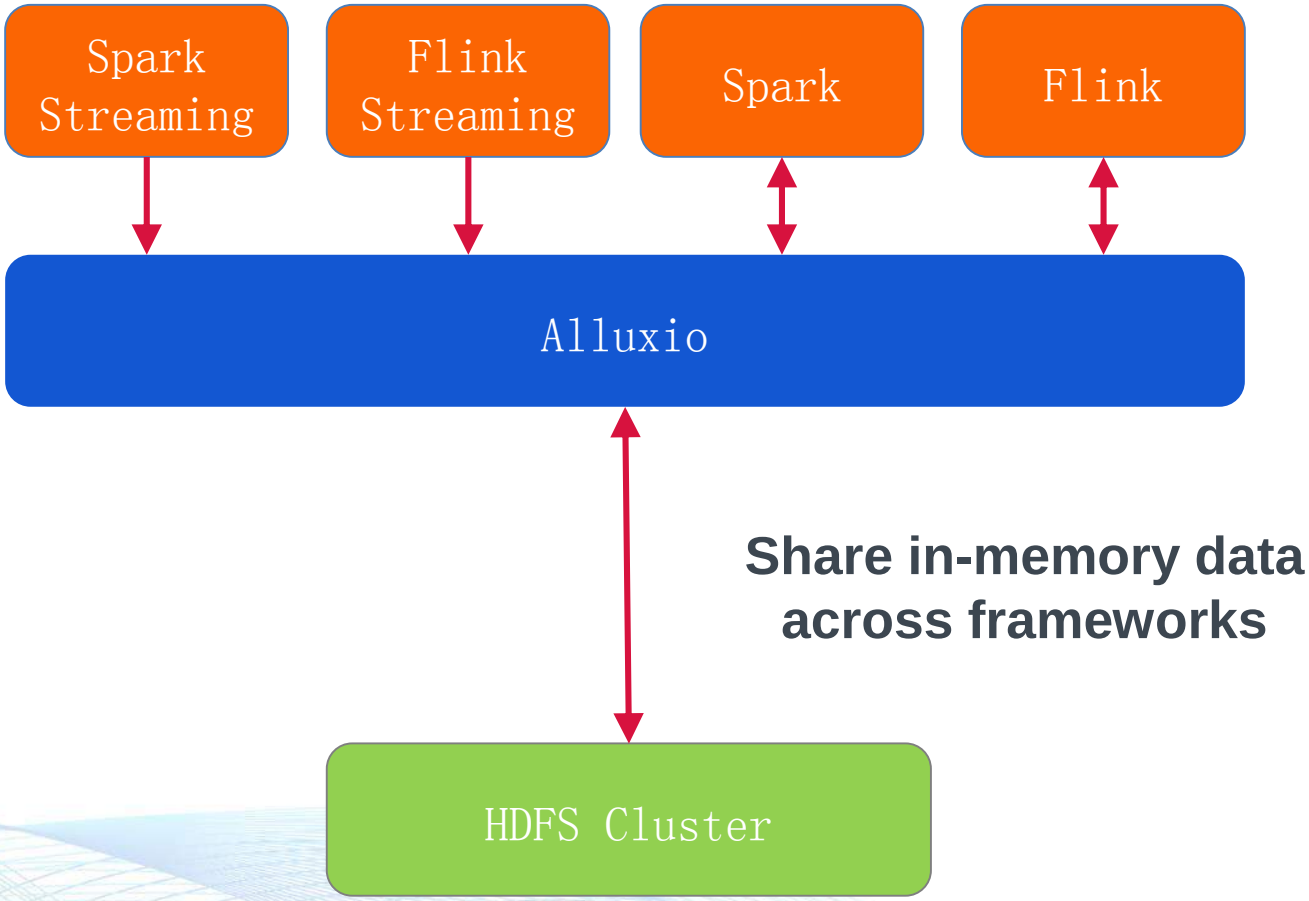
Qunar Previous Architecture



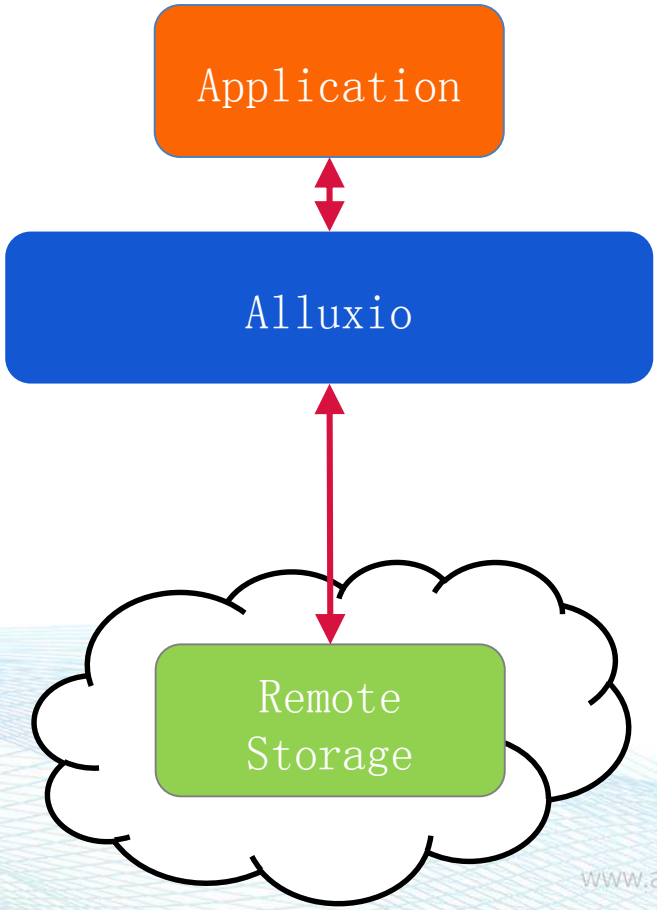
Slow access to the storage

Some jobs cannot finish

Qunar Architecture with Alluxio



CASE 3:
Accelerate access to remote storag



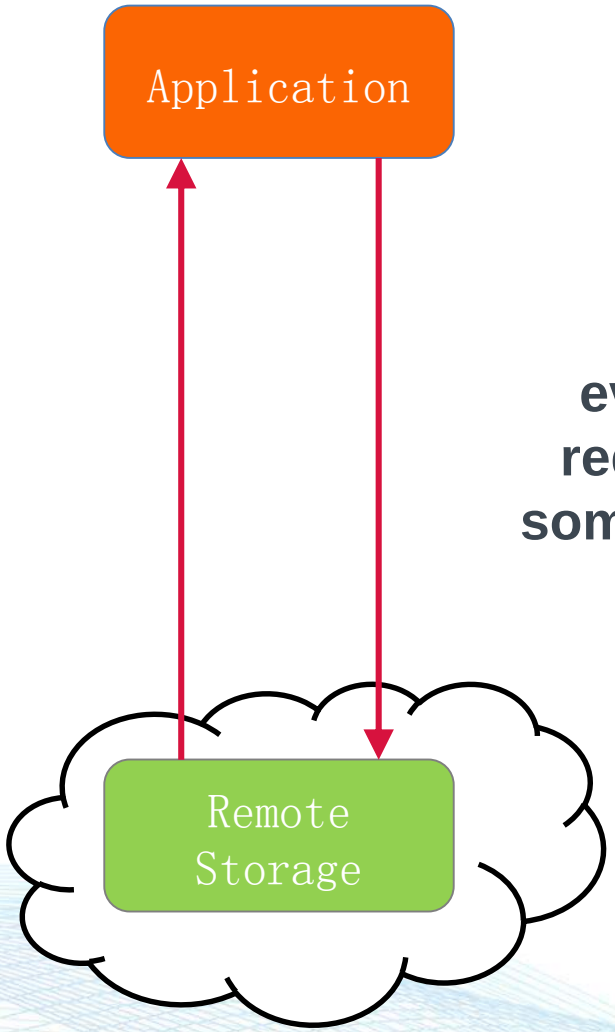
ALLUXIO

Decoupling Compute from Storage

- Different compute and storage hardware requirements
- Scale compute and storage resources independently
- Traditional filers/SANs and cost effective object stores (Amazon S3, Google GCS, Microsoft Azure Blob Store) are inherently decoupled

Accessing data requires remote I/O!

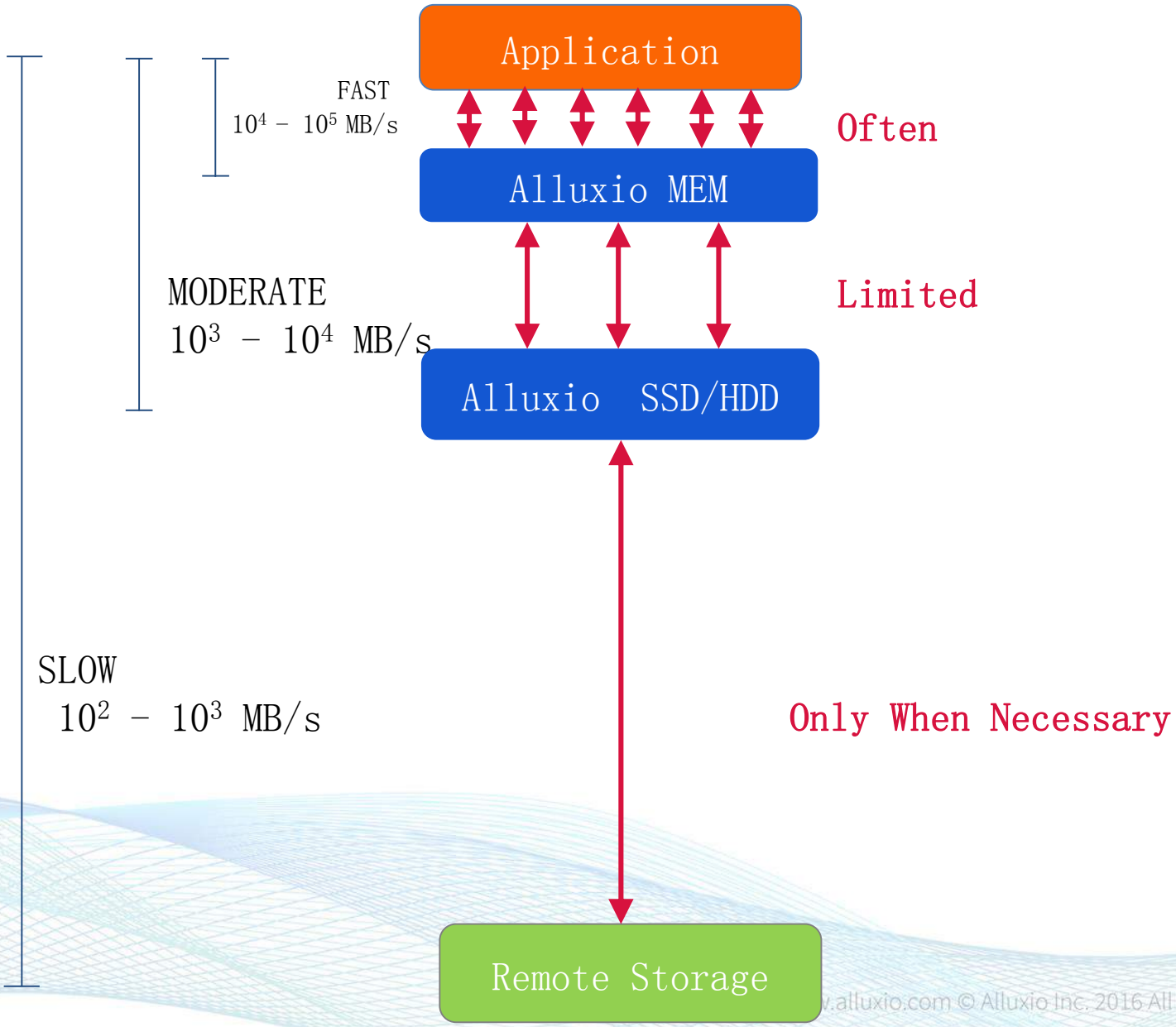
Remote I/O problems



high latency, network throughput

every data operation requires data transfer, sometimes over the WAN

Visualizing the Stack with Alluxio



CASE STUDY

Accelerate Access to Remote Storage

Baidu File System

Spark

- 200+ nodes deployment
- 2+ petabytes of storage
- Mix of memory + HDD
- 1000-worker cluster

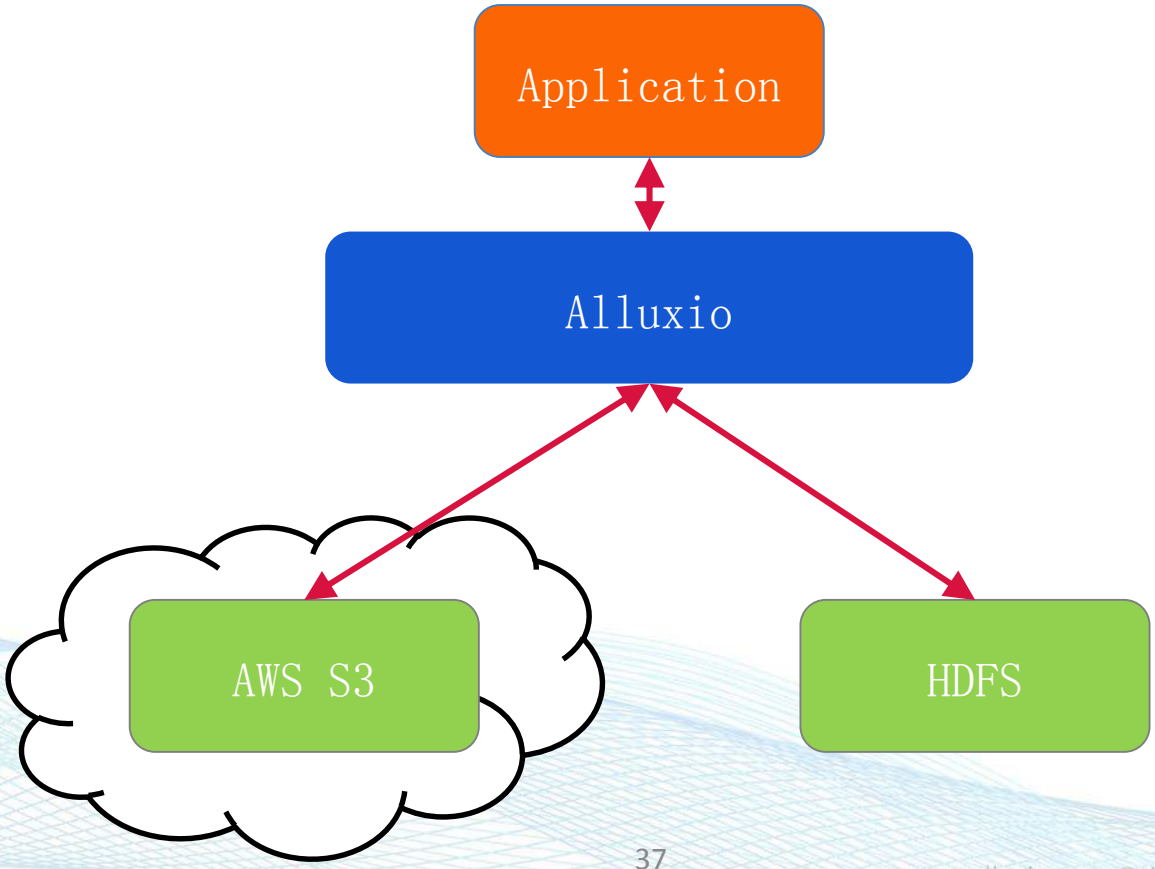
The performance was amazing. With Spark SQL alone, it took 100-150 seconds to finish a query; using Alluxio, where data may hit local or remote Alluxio nodes, it took 10-15 seconds.

- Shaoshan Liu, Baidu

RESULTS

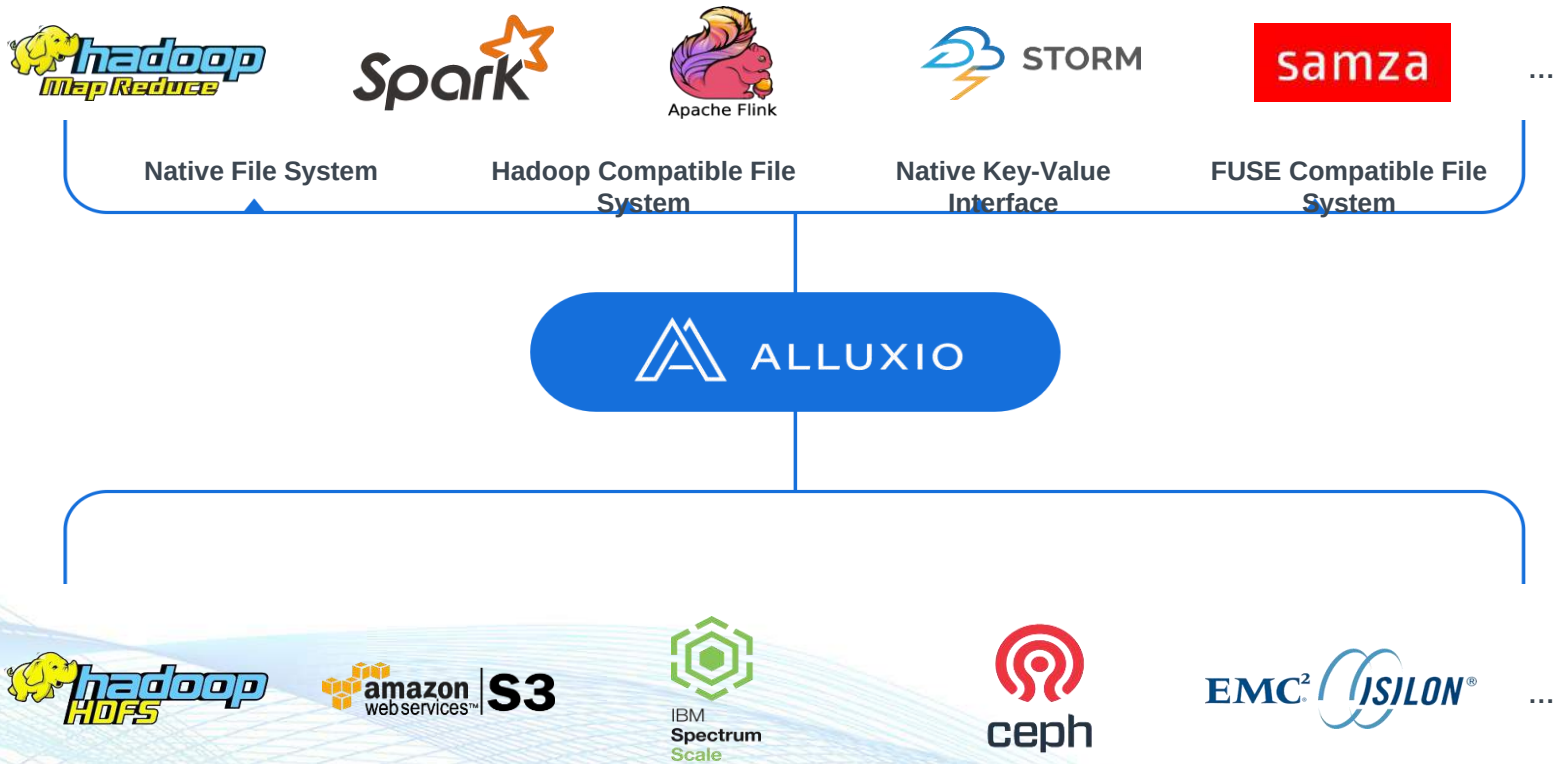
- Data queries are now 30x faster with Alluxio
- 1000-worker Alluxio cluster run stably, providing over 50TB of RAM space
- By using Alluxio, batch queries usually lasting over 15 minutes were transformed into an interactive query taking less than 30 seconds

CASE 4: Unified Name Space



Unified Name Space

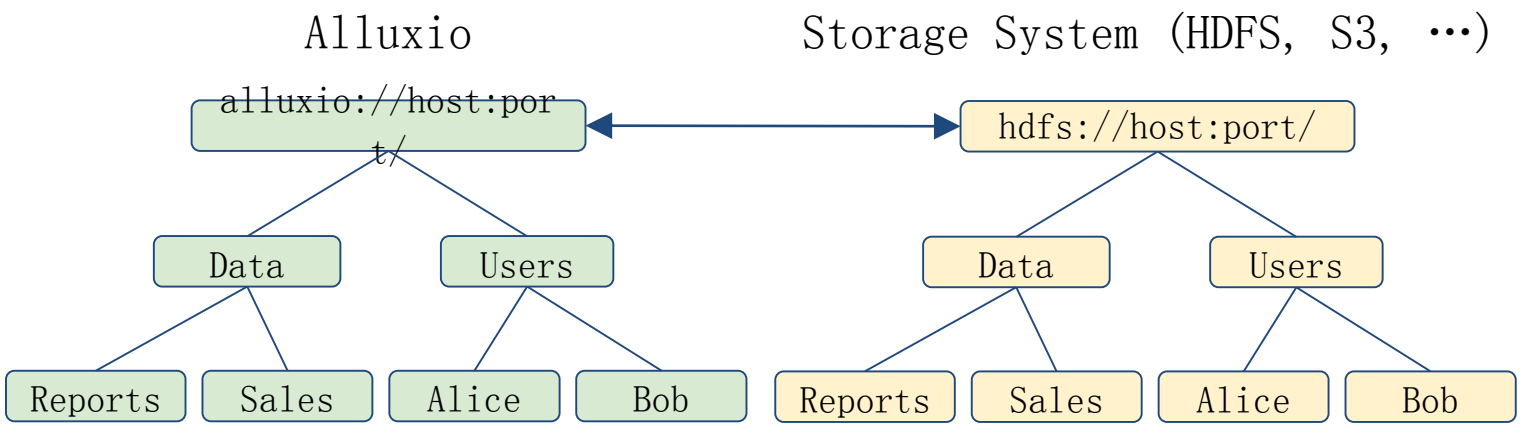
An abstraction that makes it possible for applications to access different storage systems through the same interface



Common Scenarios

- At large organizations, data spans many storage systems
- Application logic needs to integrate with different storage systems
- Legacy storage systems

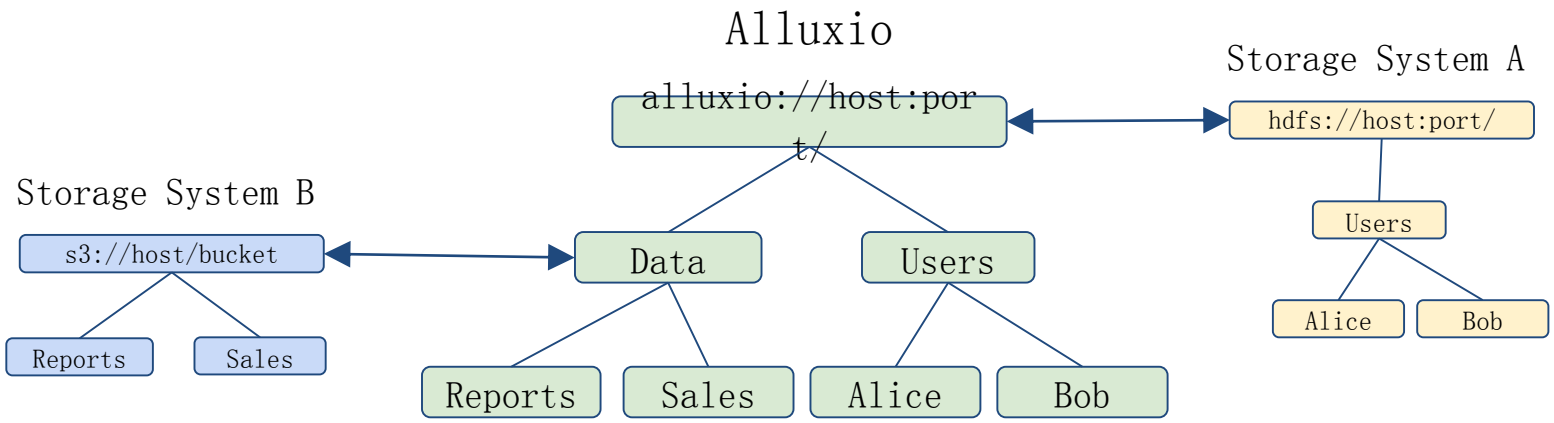
Transparent Naming



- **Operations over persisted Alluxio objects mapped transparently to underlying storage**

alluxio://Data/Sales -> hdfs://Data/Sales

Multiple Storage Systems



- **Unified namespace for multiple data sources**

  alluxio://Data/Sales -> s3://bucket/Sales

  alluxio://Users/Alice -> hdfs://Users/Alice

- **API for on-the-fly mounting / unmounting**

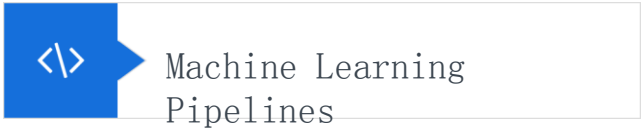
A Proliferation of Under Storage Systems

MORE…

CASE STUDY

An Internet Company

Transparently Manage Data Across Different Storage Systems



- Storage medium: MEM
- Clusters in different regions

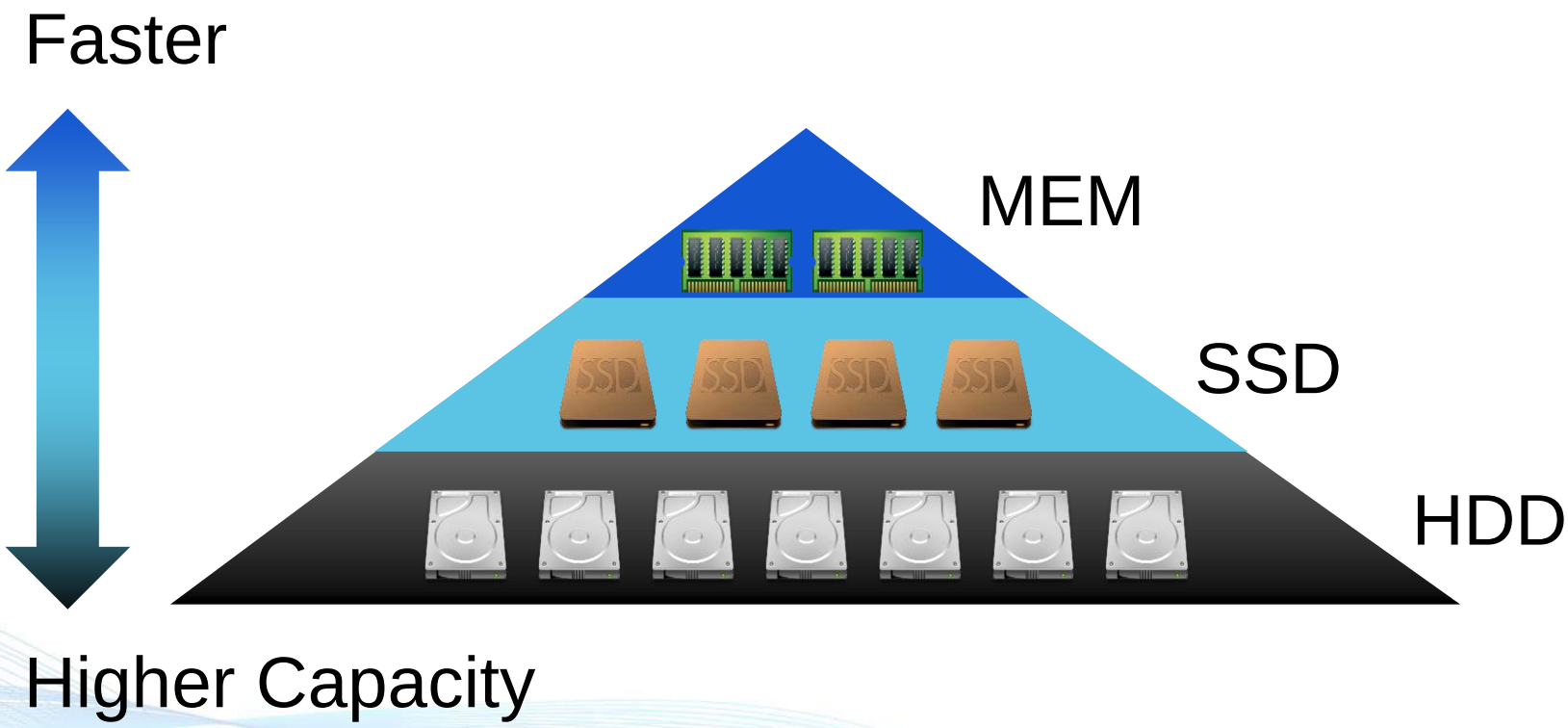
RESULTS

- Alluxio's unified namespace enables different applications and frameworks to easily interact with their data from different storage systems
- Global data centers across North America and Asia
- Tiered storage feature manages various storage resources including memory, SSD and disk

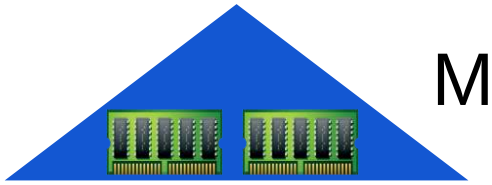
ALLUXIO

What if memory capacity is still not enough?

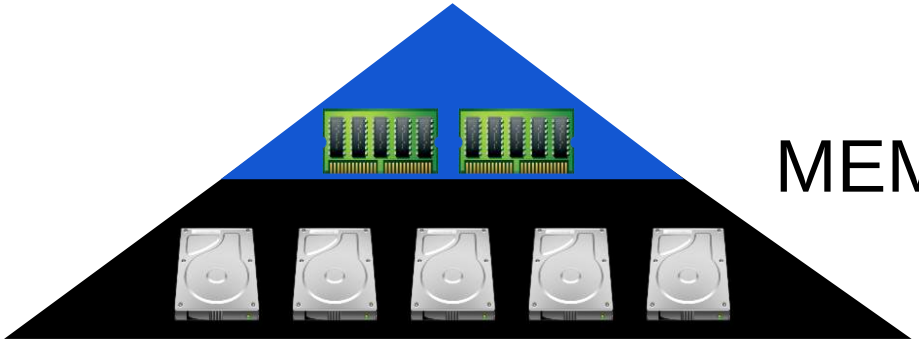
Alluxio Manages All Local Storage



Configurable Storage Tiers



MEM only



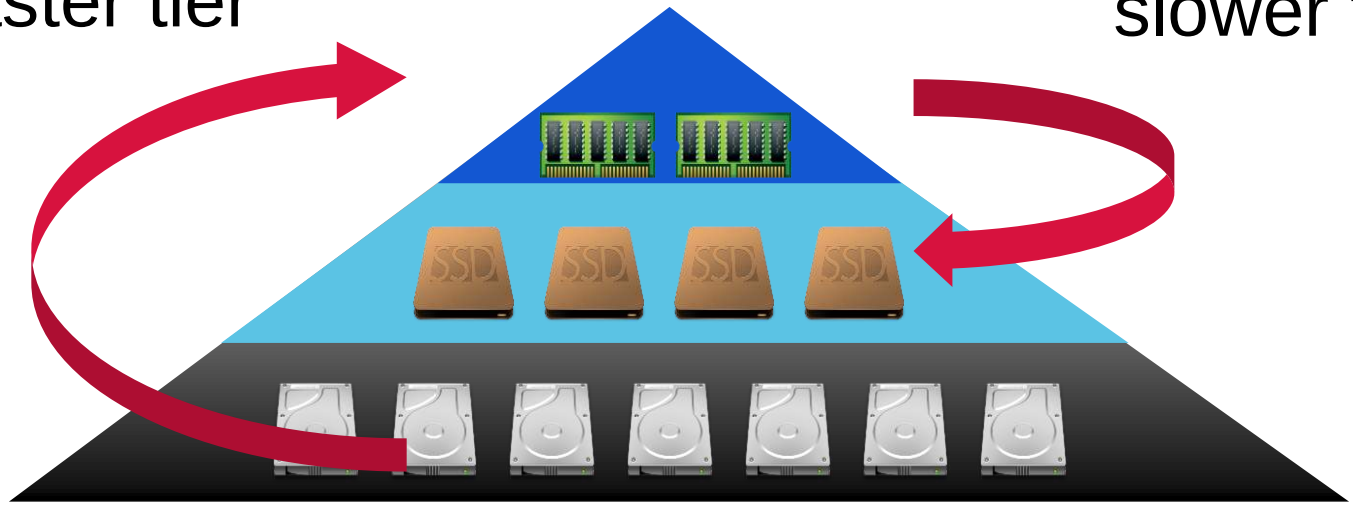
MEM + HDD

SSD only

Pluggable Tier Management Policies

Promote hot data to faster tier

Evict stale data to slower tier



Outline

- Industry Trend & Alluxio overview
- Features and Use Cases
  1. Off-heap memory to alleviate resource pressure
  2. Fast data sharing between jobs
  3. Accelerate access to remote storage
  4. Unified namespace
- **Summary**

Takeaway: When to use Alluxio

- Two or more jobs access the same dataset
- Job(s) may not always succeed
- Spark with memory/GC pressure
- Jobs/Apps are pipelined
- Resulting data does not need to be immediately persisted
- Interact with remote data
- Data stored across different storage systems

ALLUXIO

Try out Alluxio 1.2.0

http://www.alluxio.org/releases

Resources

- Alluxio Project: http://www.alluxio.org
- Development: https://github.com/Alluxio/alluxio
- Meet Friends: http://www.meetup.com/Alluxio
- Alluxio, Inc.: http://www.alluxio.com
- Training: http://www.alluxio.com/alluxio-training/
- Contact us: info@alluxio.com

Thank you!

Contact us anytime at info@alluxio.com.

Follow us on Twitter @Alluxio.

Check out www.alluxio.com and www.alluxio.org.

T11

2016

THANKS

TalkingData