ALLUXIO

Building Data Orchestration for Big Data Analytics in the Cloud

Bin Fan | Founding Engineer | Alluxio
binfan@alluxio.com

07/17/2019

About Me

@binfan

binfan@alluxio.com

@apc999

Founding Engineer & Open Source Maintainer | Alluxio

The Alluxio Story

amplab
2013

Originated as Tachyon project, at the UC Berkley's AMP Lab by then Ph.D. student & now Alluxio CTO, Haoyuan (H.Y.) Li.

ALLUXIO
2015

Open Source project established & company to commercialize Alluxio founded

ANDREESSEN HOROWITZ

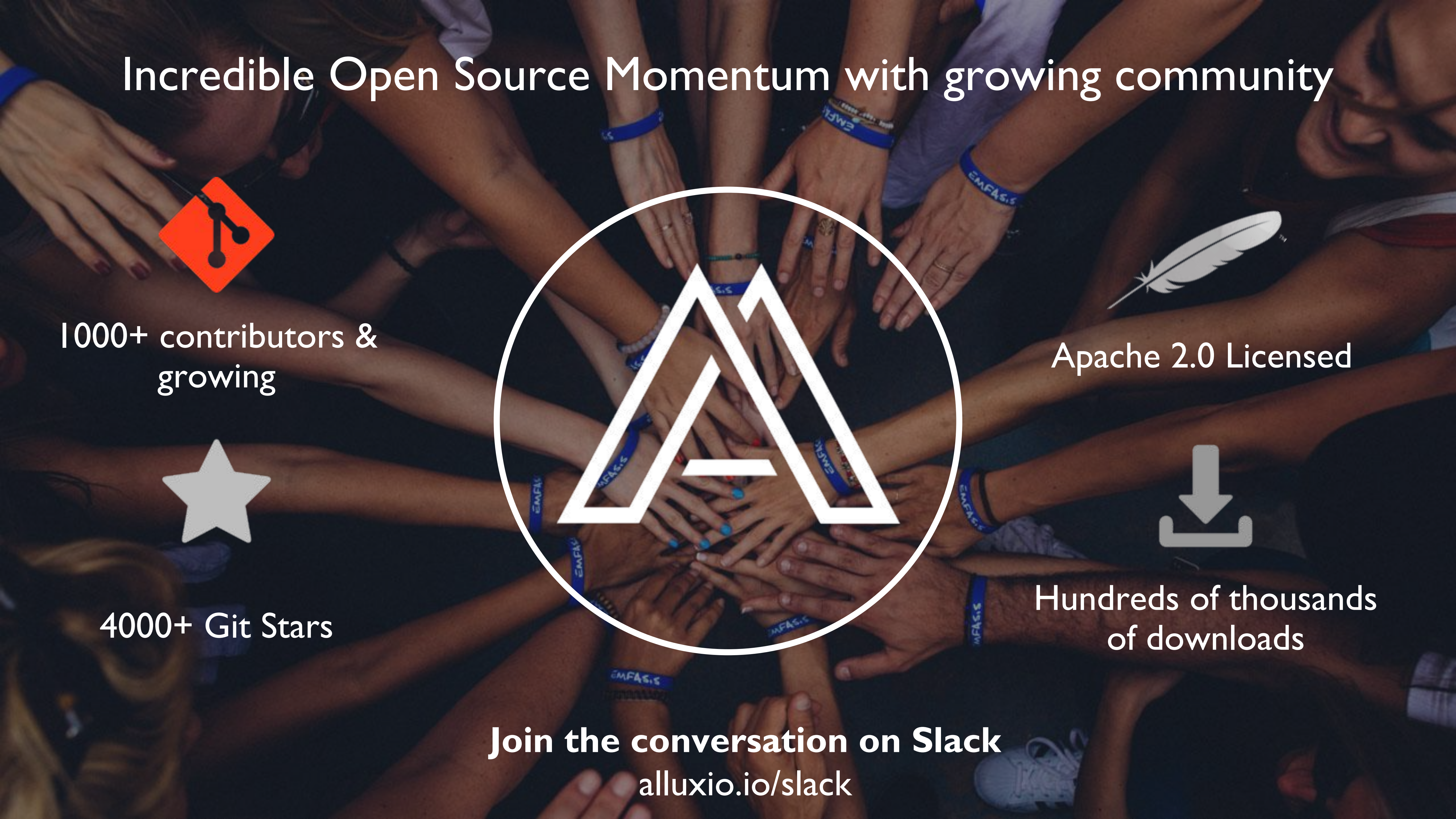
Goal: **Orchestrate Data at Memory Speed for the Cloud** for data driven apps such as Big Data Analytics, ML and AI.

CRN
2018

NW NETWORKWORLD NW FROM IDG
2018

IMPACT 50 THE INDUSTRY'S MOST IMPACTFUL COMPANIES insideBIGDATA
2019

Incredible Open Source Momentum with growing community



1000+ contributors & growing

4000+ Git Stars

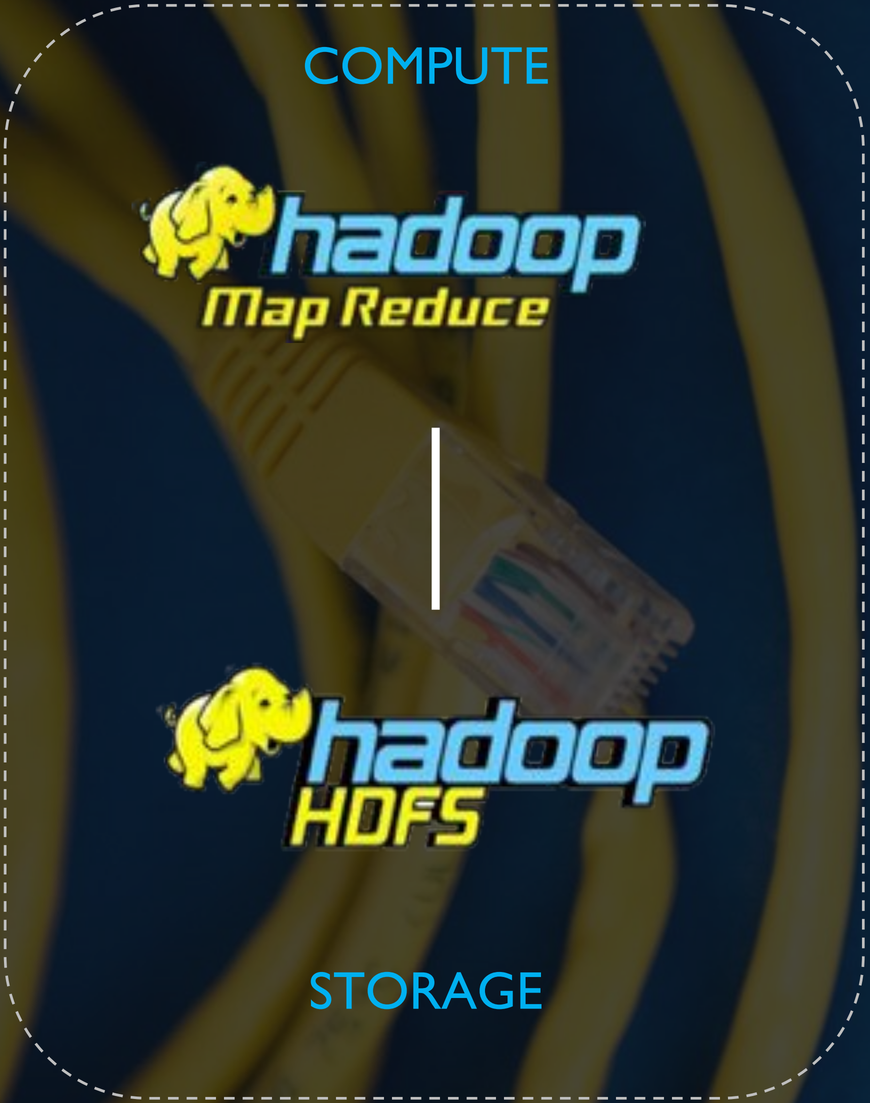
Apache 2.0 Licensed

Hundreds of thousands of downloads

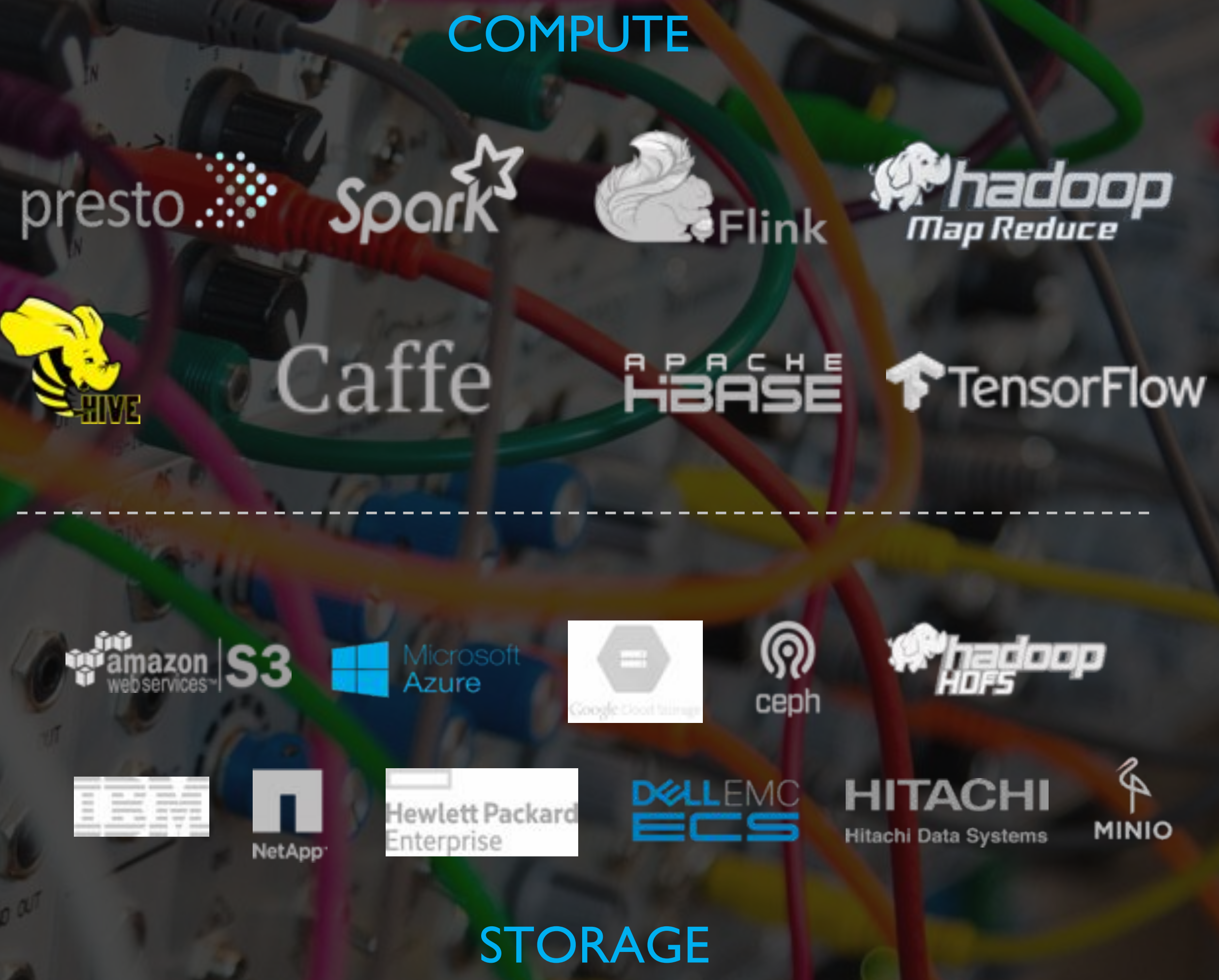
Join the conversation on Slack

alluxio.io/slack

Data Ecosystem - *Beta*

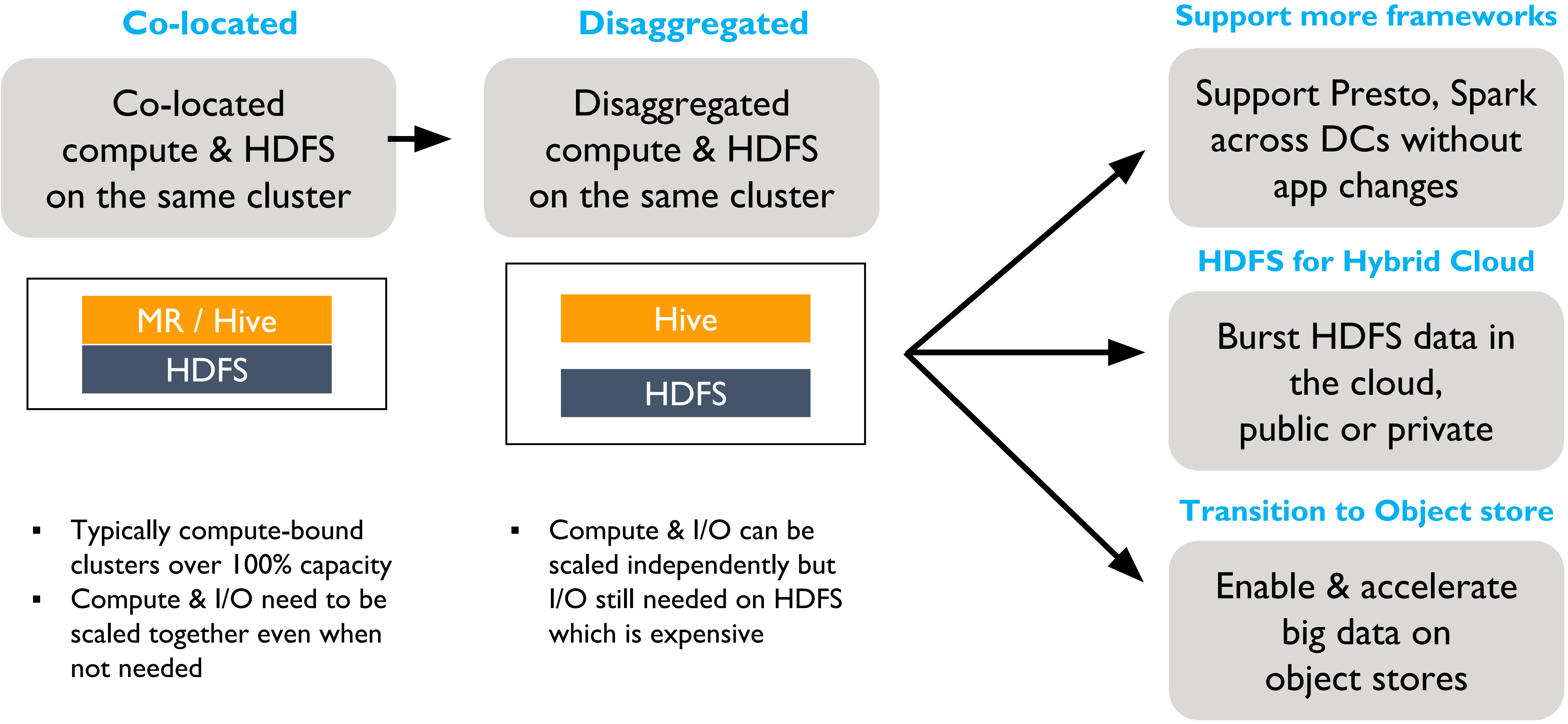


Data Ecosystem 1.0



ALLUXIO

Data stack journey and innovation paths



Co-located

Co-located compute & HDFS on the same cluster

- Typically compute-bound clusters over 100% capacity
- Compute & I/O need to be scaled together even when not needed

Disaggregated

Disaggregated compute & HDFS on the same cluster

- Compute & I/O can be scaled independently but I/O still needed on HDFS which is expensive

Support more frameworks

Support Presto, Spark across DCs without app changes

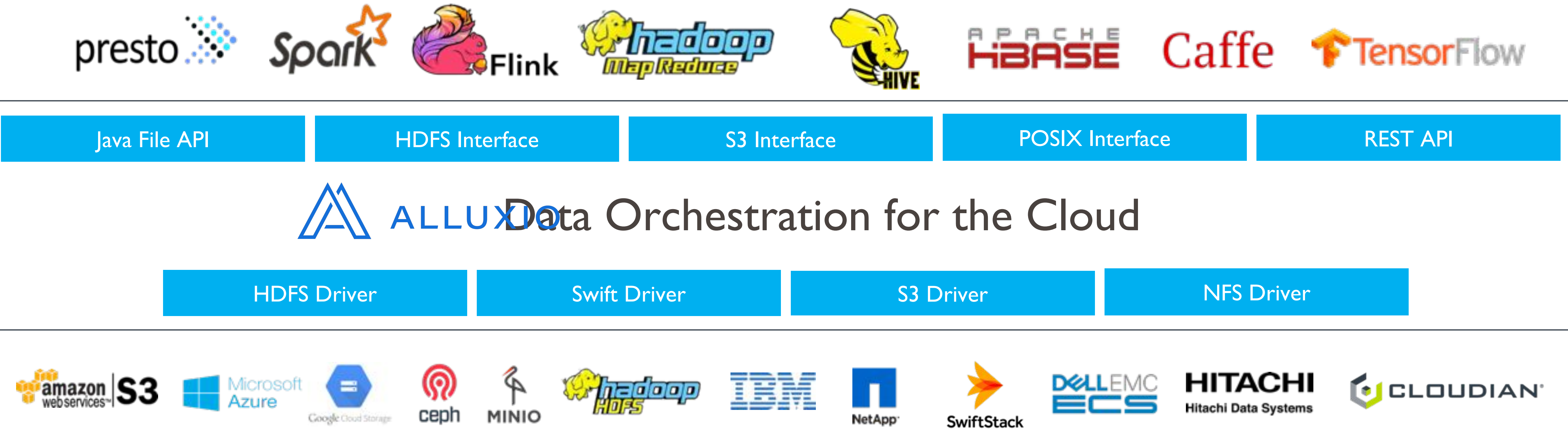
HDFS for Hybrid Cloud

Burst HDFS data in the cloud, public or private

Transition to Object store

Enable & accelerate big data on object stores

Independent scaling of compute & storage



Java File API | HDFS Interface | S3 Interface | POSIX Interface | REST API

ALLUXIO Data Orchestration for the Cloud

HDFS Driver | Swift Driver | S3 Driver | NFS Driver

APIs to Interact with data in Alluxio

Application have great flexibility to read / write data with many options

Spark

```
> rdd = sc.textFile(“alluxio://localhost:19998/myInput”)
```

Presto

```
CREATE SCHEMA hive.web
WITH (location = 'alluxio://master:port/my-table/')
```

POSIX

```
$ cat /mnt/alluxio/myInput
```

Java

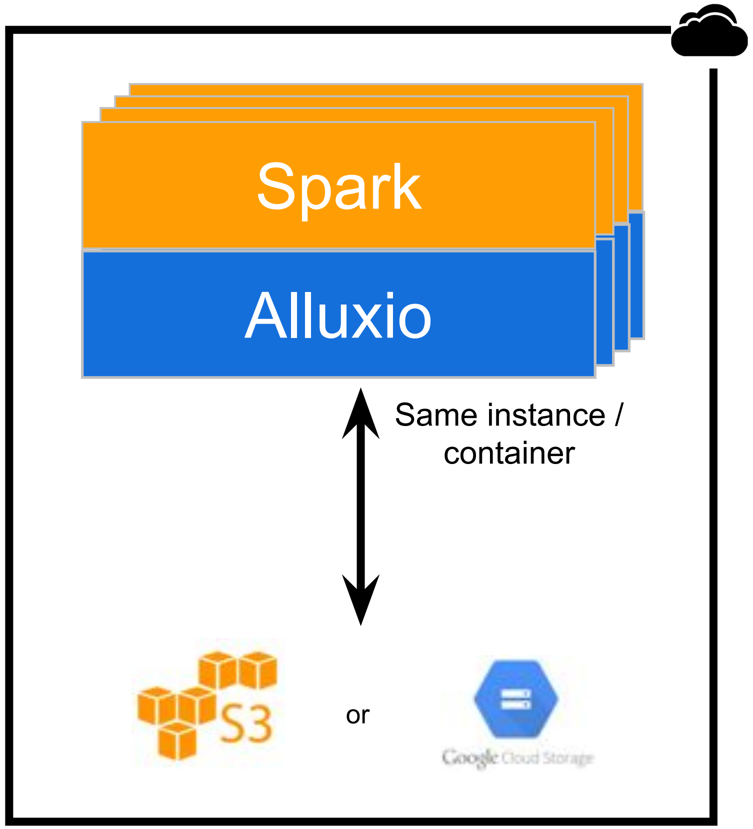
```
FileSystem fs = FileSystem.Factory.get();
FileInStream in = fs.openFile(new AlluxioURI("/myInput"));
```

Use Case: Distributed Caching for Cloud Storage

Compute caching for S3 / GCS

- S3 performance is variable and consistent query SLAs are hard to achieve
- S3 metadata operations are expensive making workloads run longer
- S3 egress costs add up making the solution expensive
- S3 is eventually consistent making it hard to predict query results

Accelerate analytical frameworks on the public cloud

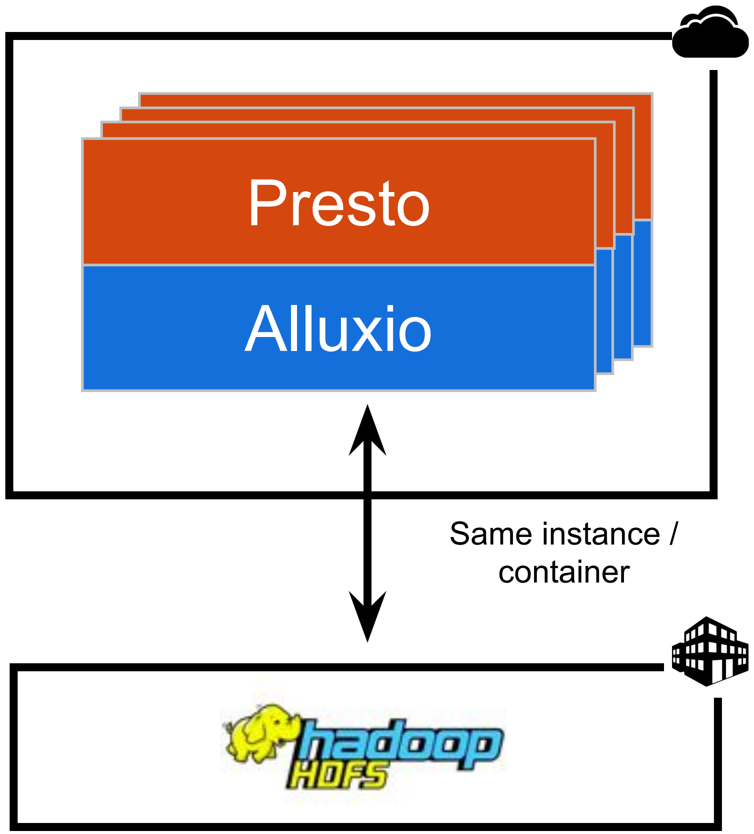


Use Case: Data Federation with Hybrid Cloud

HDFS for Hybrid Cloud

- Accessing data over WAN too slow
- Copying data to compute cloud time consuming and complex
- Using another storage system like S3 means expensive application changes
- Using S3 via HDFS connector leads to extremely low performance

Burst big data workloads in hybrid cloud environments



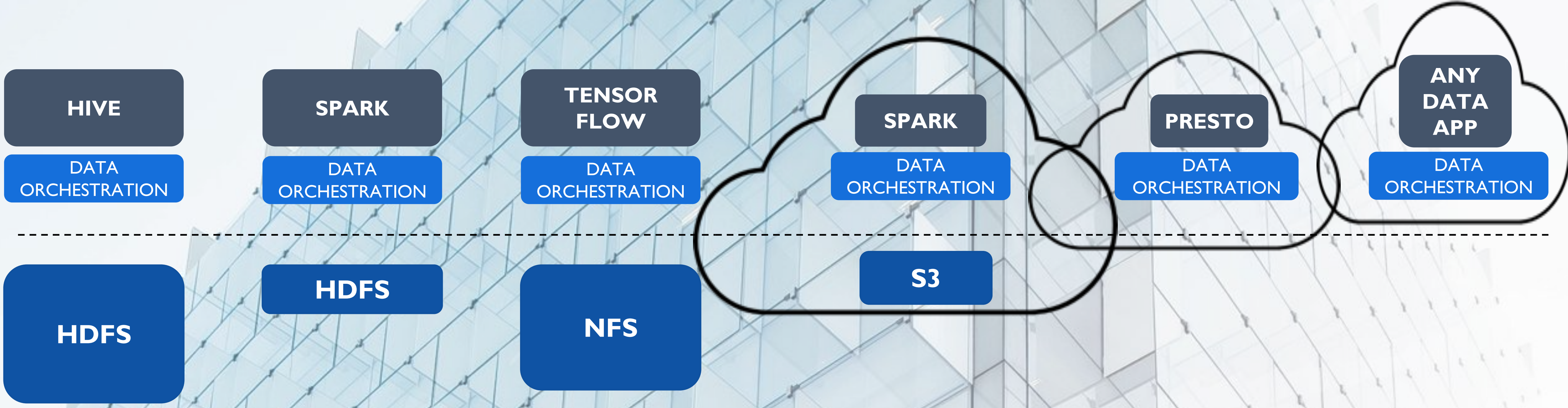
Solution Benefits

- Same performance as local
- Same end-user experience

- 100% of I/O is offloaded

Abstract & orchestrate data across data silos

COMPUTE SPREAD ACROSS MANY DIFFERENT FRAMEWORKS



DATA IN DISPARATE STORAGE SYSTEMS

ALLUXIO

Alluxio – Key Innovations

Data Locality
with Intelligent Multi-tiering

Accelerate big data workloads with transparent tiered local data

Data Accessibility
for popular APIs & API translation

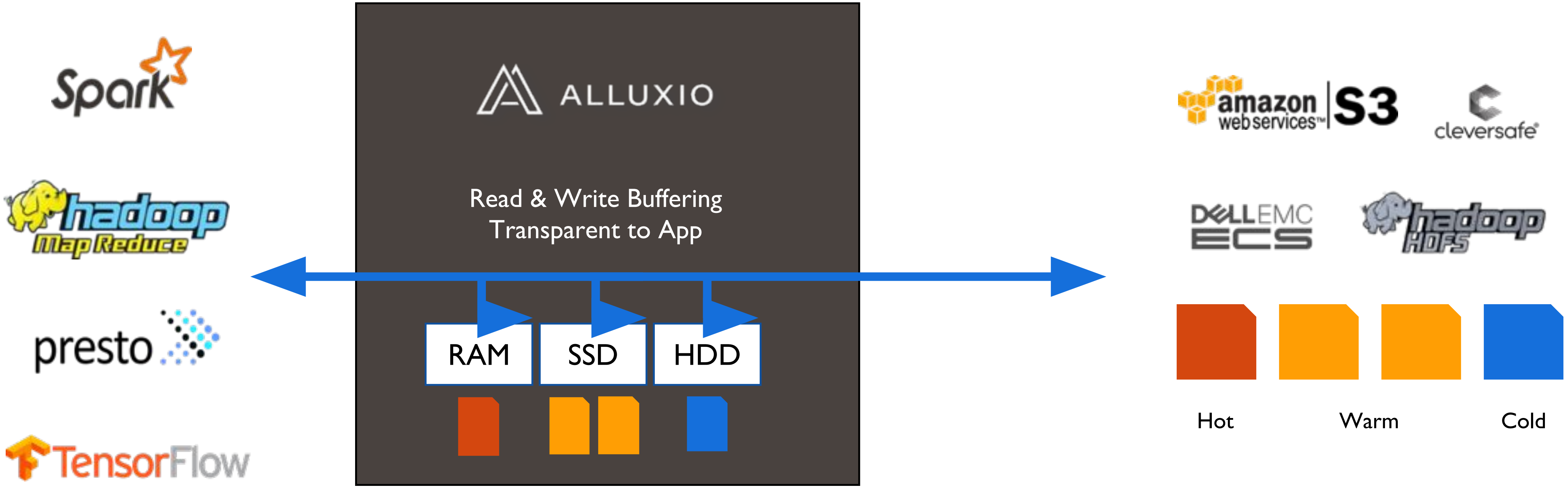
Run Spark, Hive, Presto, ML workloads on your data located anywhere

Data Elasticity
with a unified namespace

Abstract data silos & storage systems to independently scale data on-demand with compute

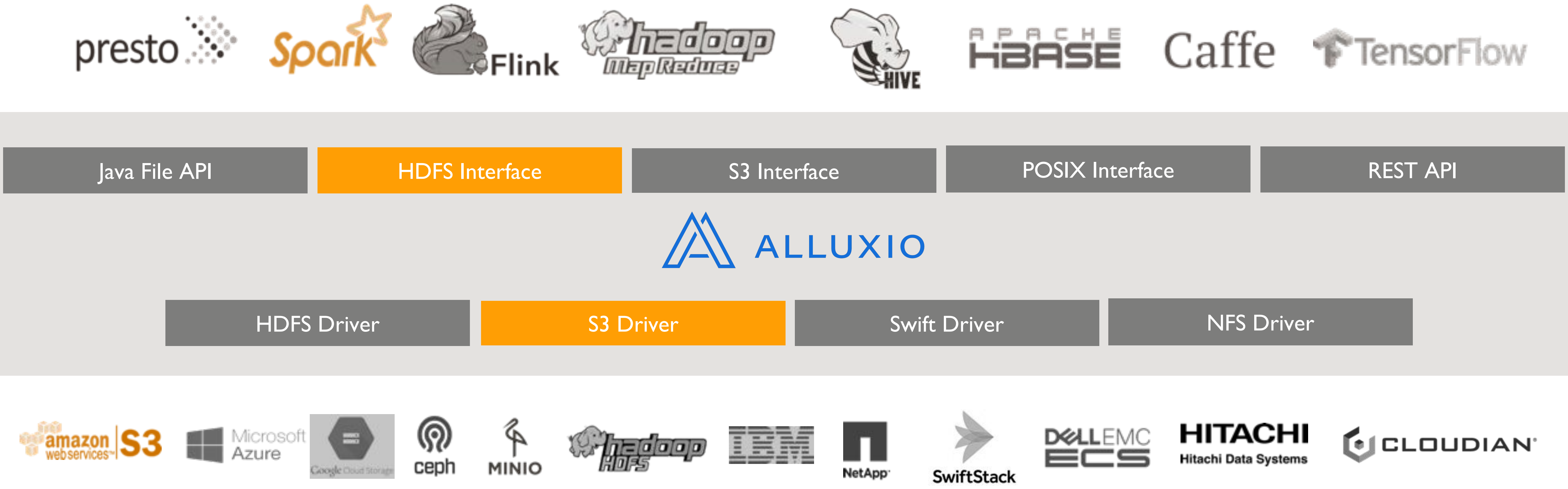
Data Locality with Intelligent Multi-tiering

Local performance from remote data using multi-tier storage



Data Accessibility via popular APIs and API Translation

Convert from Client-side Interface to native Storage Interface



Java File API | HDFS Interface | S3 Interface | POSIX Interface | REST API

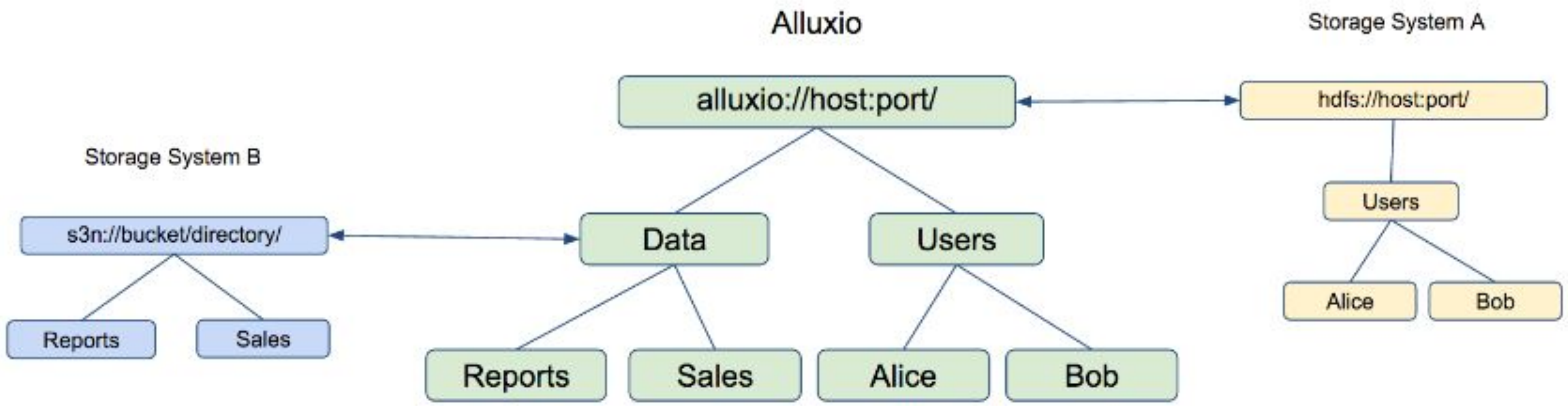
ALLUXIO

HDFS Driver | S3 Driver | Swift Driver | NFS Driver

Data Elasticity via Unified Namespace

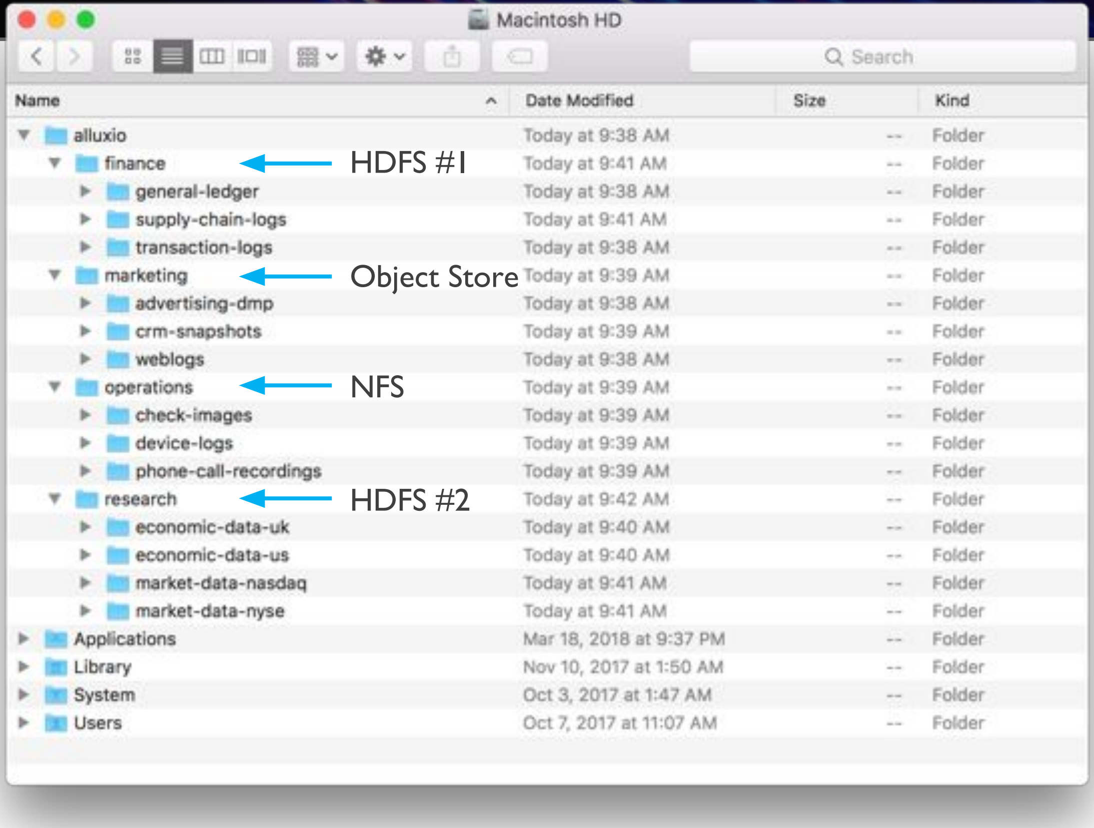
Enables effective data management across different Under Store

- Uses Mounting with Transparent Naming



Unified Namespace: Global Data Accessibility

Transparent access to understorage makes all enterprise data available locally



Name	Date Modified	Size	Kind
alluxio	Today at 9:38 AM	--	Folder
finance	Today at 9:41 AM	--	Folder
general-ledger	Today at 9:38 AM	--	Folder
supply-chain-logs	Today at 9:41 AM	--	Folder
transaction-logs	Today at 9:38 AM	--	Folder
marketing	Today at 9:39 AM	--	Folder
advertising-dmp	Today at 9:38 AM	--	Folder
crm-snapshots	Today at 9:39 AM	--	Folder
weblogs	Today at 9:38 AM	--	Folder
operations	Today at 9:39 AM	--	Folder
check-images	Today at 9:39 AM	--	Folder
device-logs	Today at 9:39 AM	--	Folder
phone-call-recordings	Today at 9:39 AM	--	Folder
research	Today at 9:42 AM	--	Folder
economic-data-uk	Today at 9:40 AM	--	Folder
economic-data-us	Today at 9:40 AM	--	Folder
market-data-nasdaq	Today at 9:41 AM	--	Folder
market-data-nyse	Today at 9:41 AM	--	Folder
Applications	Mar 18, 2018 at 9:37 PM	--	Folder
Library	Nov 10, 2017 at 1:50 AM	--	Folder
System	Oct 3, 2017 at 1:47 AM	--	Folder
Users	Oct 7, 2017 at 11:07 AM	--	Folder

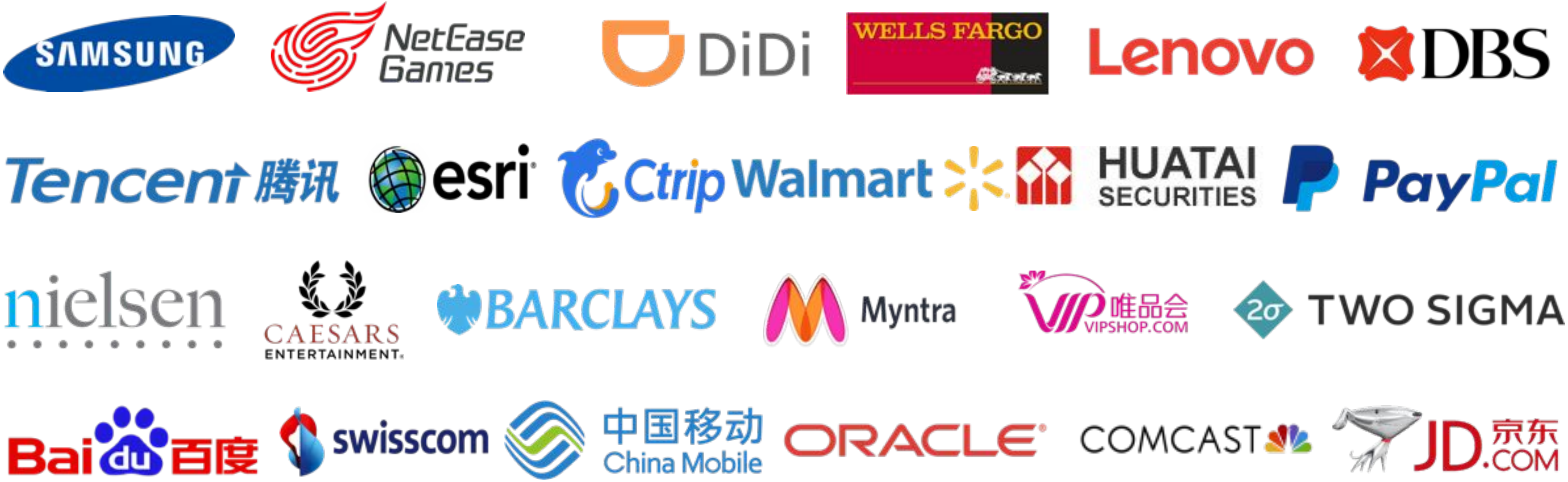
SUPPORTS

- HDFS
- NFS
- OpenStack
- Ceph
- Amazon S3
- Azure
- Google Cloud

IT OPS FRIENDLY

- Storage mounted into Alluxio by central IT
- Security in Alluxio mirrors source data
- Authentication through LDAP/AD
- Wireline encryption

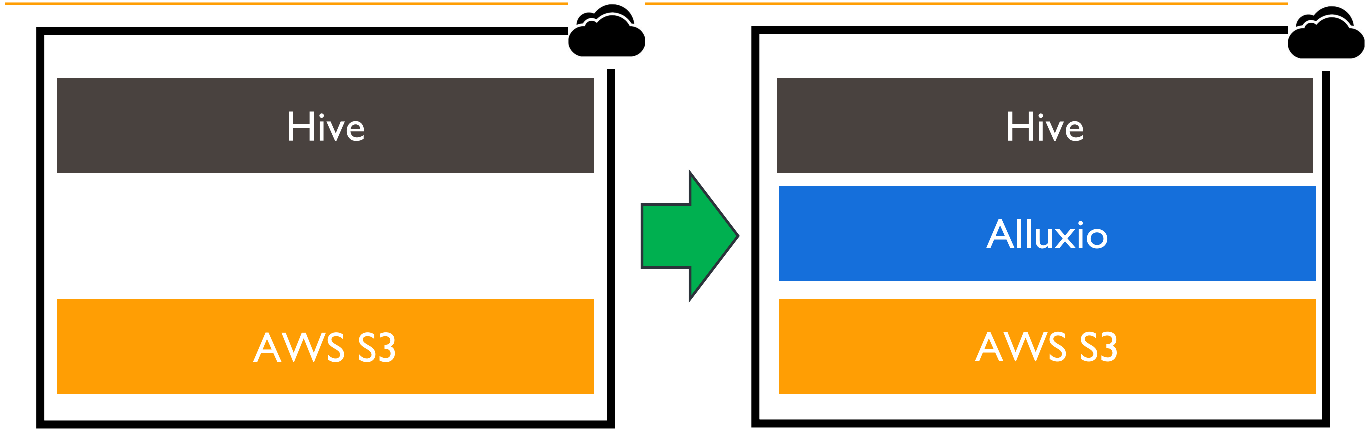
Companies Using Alluxio



Bazaarvoice

Leading Digital marketing Company in Austin

Use Case | Compute Caching for Cloud



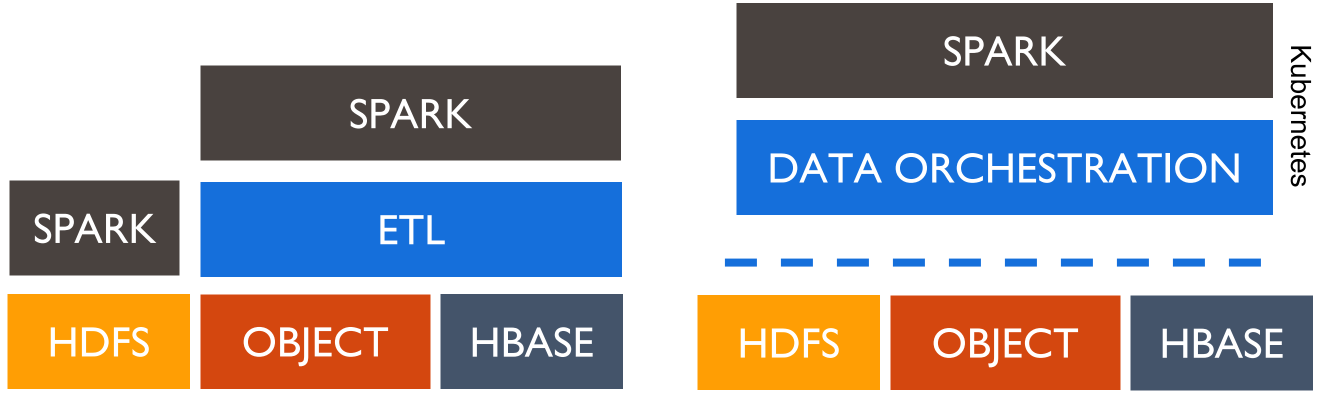
- Cache hot data in Alluxio, keep all data in S3
- Faster time to insights with seamless data orchestration
- Accelerated workloads with memory-first data approach by 10x

https://www.alluxio.io/blog/accelerate-spark-and-hive-jobs-on-aws-s3-by-10x-with-alluxio-tiered-storage/

China Unicom

Leading Chinese Telco serving 320 million subscribers

Use case | Data orchestration for agility

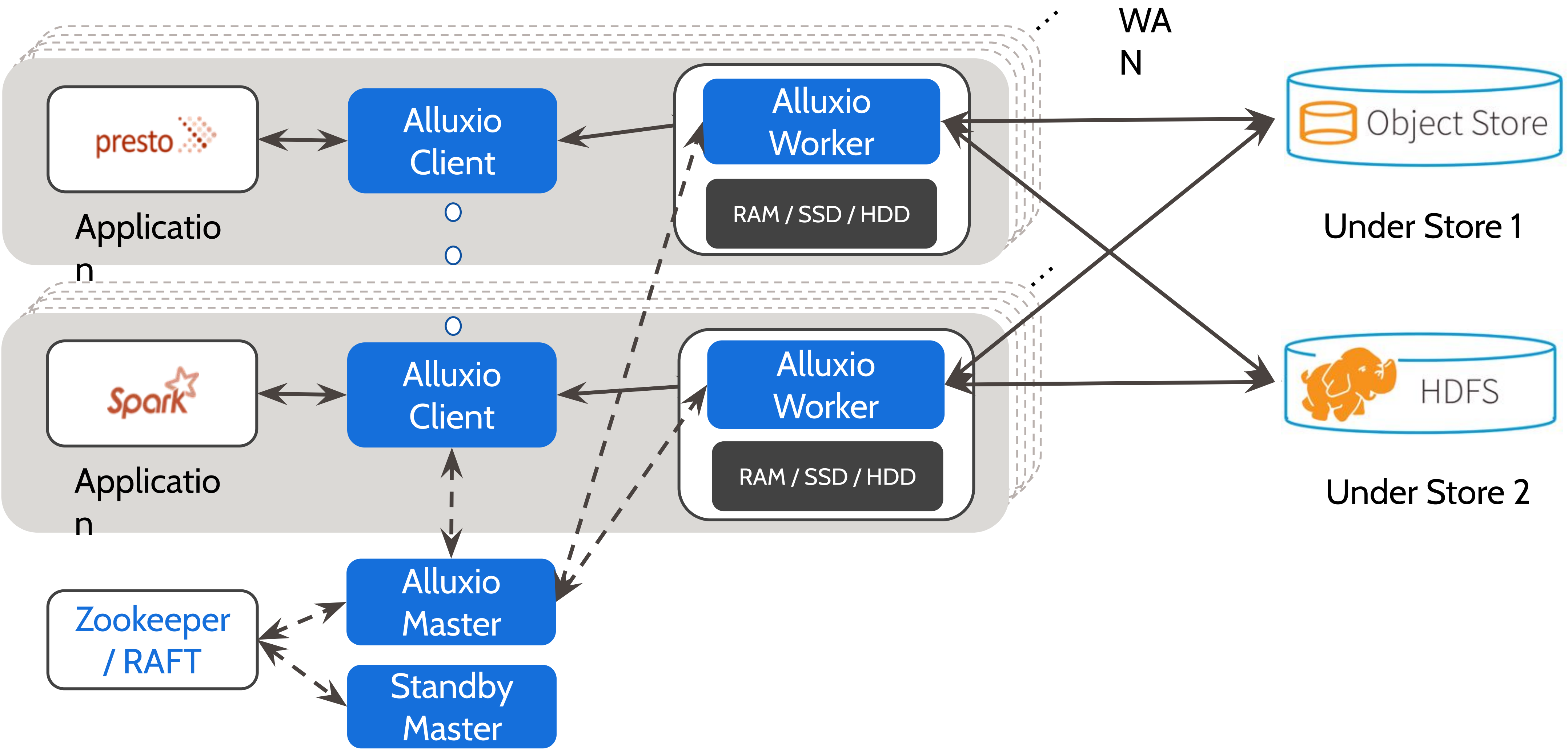


- Single namespace to access & address all data
- Data local to compute accelerates workloads

ALLUXIO

Architecture & Data Flow

Alluxio Reference Architecture



Alluxio Files and Blocks

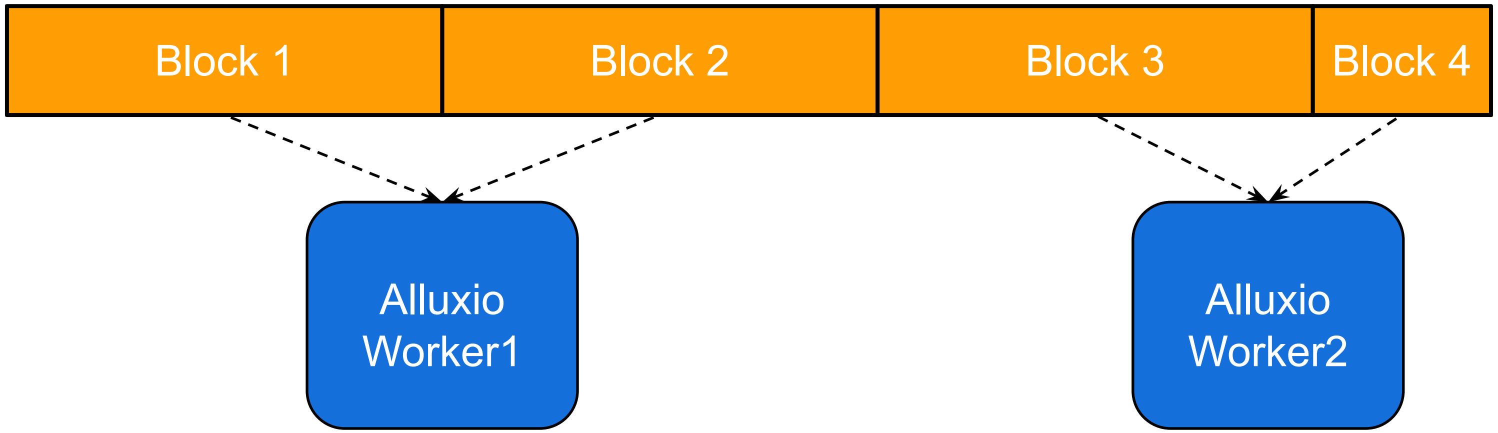
- Files are immutable once completed
- Blocks are stored on Alluxio Workers

  Blocks of a file can be on different workers

Flexible Block Sizes

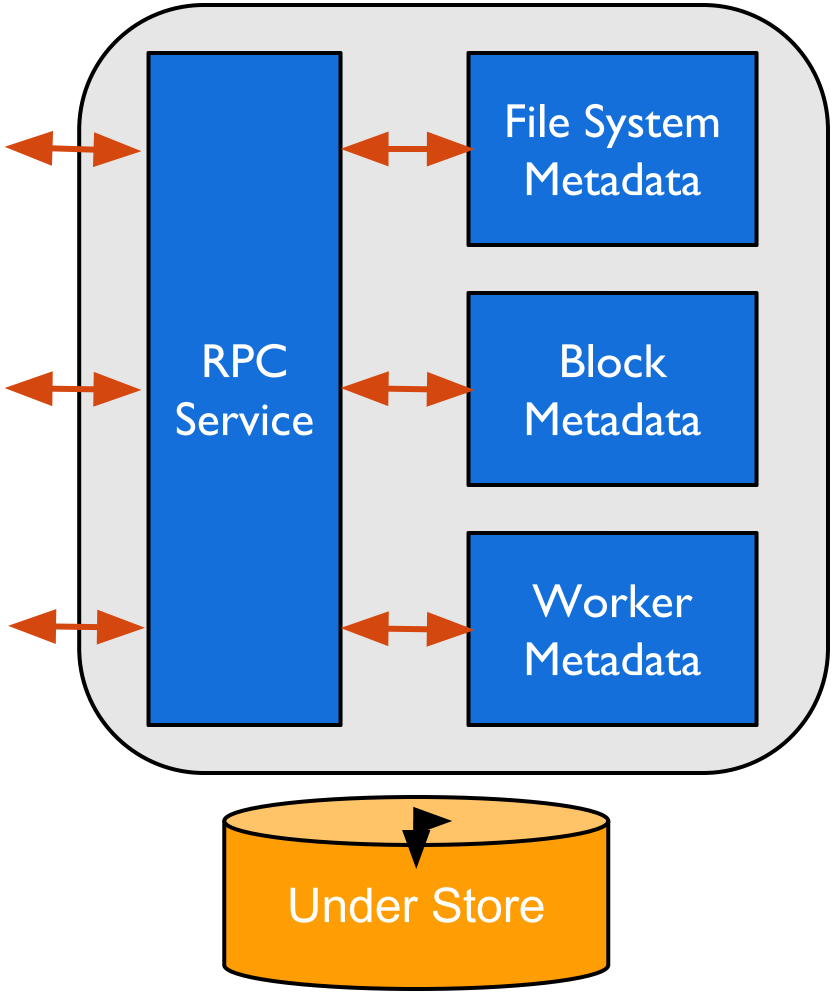
- Default block size is (512 MB)
- If understore block size is greater: The file will only take up as much space as needed
- If understore block size is smaller: File will be split up among multiple blocks
- Last block of a file is not required to be a full block size

Alluxio File



Alluxio Master – Metadata Service

- Master responsible for managing metadata
  - File system namespace (inode tree)
  - Block / worker info
- Standby masters used for checkpointing and fault tolerance mode
  - Zookeeper / RAFT used for leader election
- Master writes journal for durable operations
  - Standby masters replay changes from the journal
- Performs Under Store metadata operations

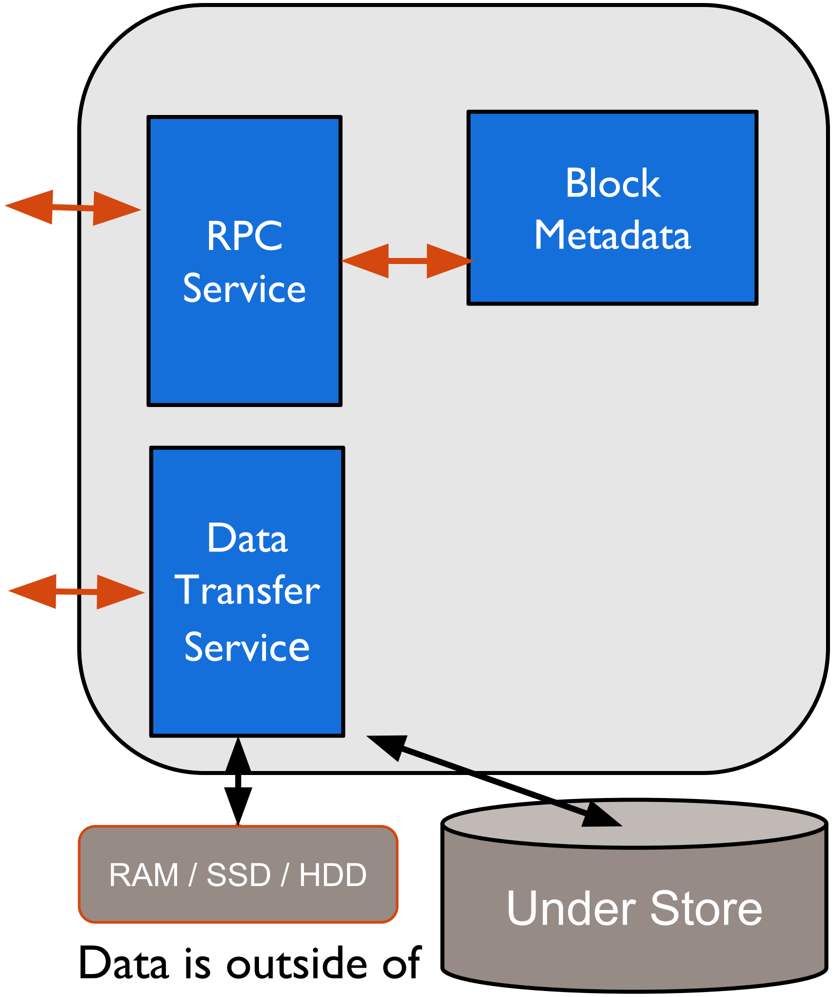


Efficient Metadata Operations: Alluxio on S3

- Efficient bucket listing:
  - Key operations for SparkSQL/Presto query planning
  - Object metadata will be cached in Alluxio after 1st read
- Efficient file rename
  - Slow operations on S3 as a copy followed by delete
  - Alluxio implements “persist after rename”
  - Enables Speculative execution
- Batching UFS operations to S3

Alluxio Workers – Data Service

- Workers responsible for storing and serving block data
- Each worker manages the metadata for the block data it stores
- Workers store block data on various local storage mediums
  - Memory
  - SSD
  - HDD
- Performs Under Store data operations



Data is outside of worker JVM

Key Innovations & Optimization in Data Service

- Avoid JVM GC:
  - Storing blocks off-heap (e.g., RAMDISK)
- Data Capacity:
  - Tiered Storage Management using HDD, SSD, MEM
- Data Throughput:
  - Fine grained block locking for high concurrency
  - gRPC based streaming-RPC service stub
- Async Data Archival to S3
  - Apps write to Alluxio (at Alluxio speed), then Alluxio persist data to S3 async (at S3 speed)

Interacting with data in Alluxio – flexible app patterns

Application have great flexibility to read / write data with many options

Writing Data

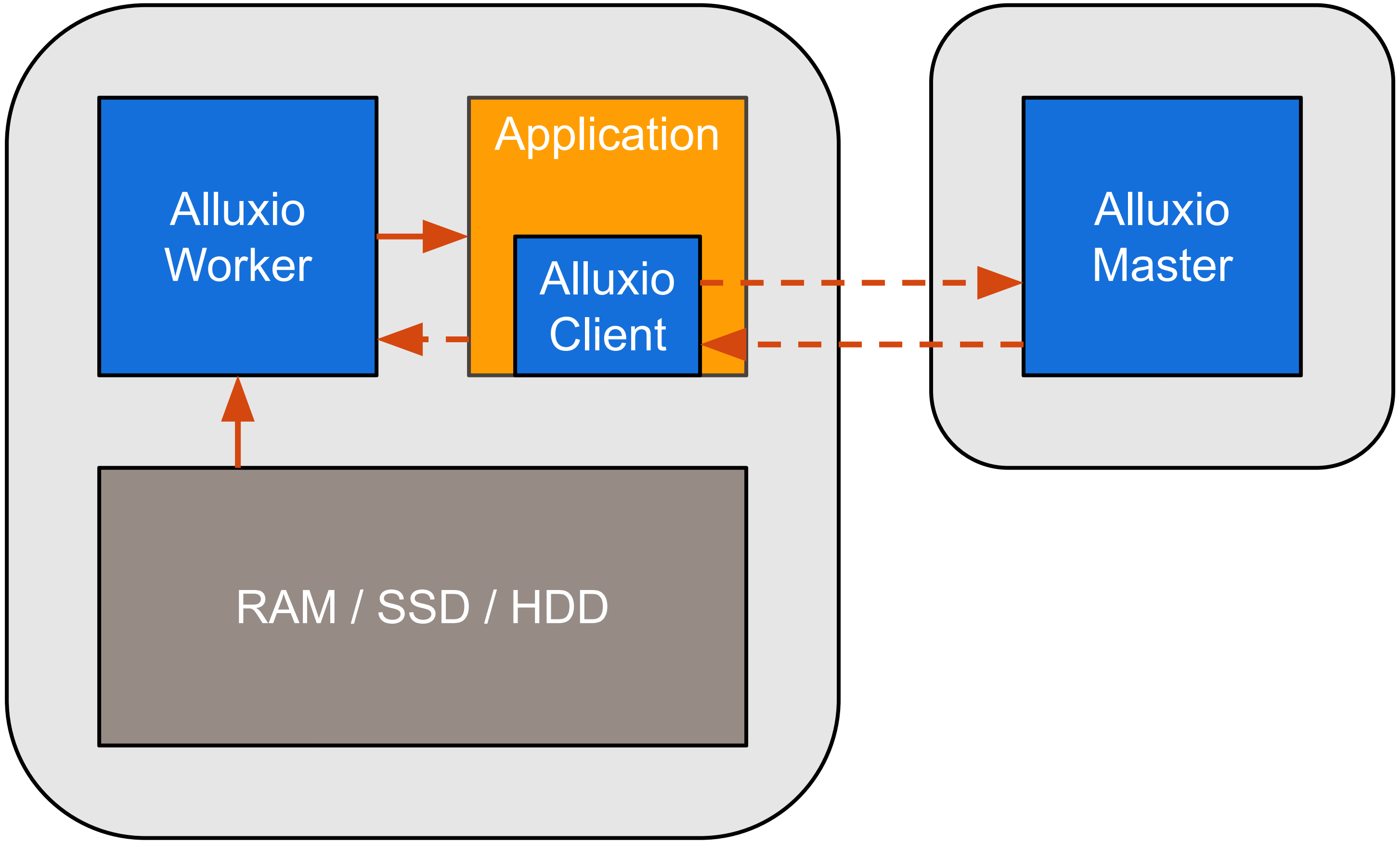
- Write only to Alluxio
- Write only to Under Store
- Write synchronously to Alluxio and Under Store
- Write to Alluxio and asynchronously write to Under Store
- Write to Alluxio and replicate to N other workers
- Write to Alluxio and async write to multiple Under stores

Reading Data

- From under store
- From a co-located Alluxio node
- From a different Alluxio node

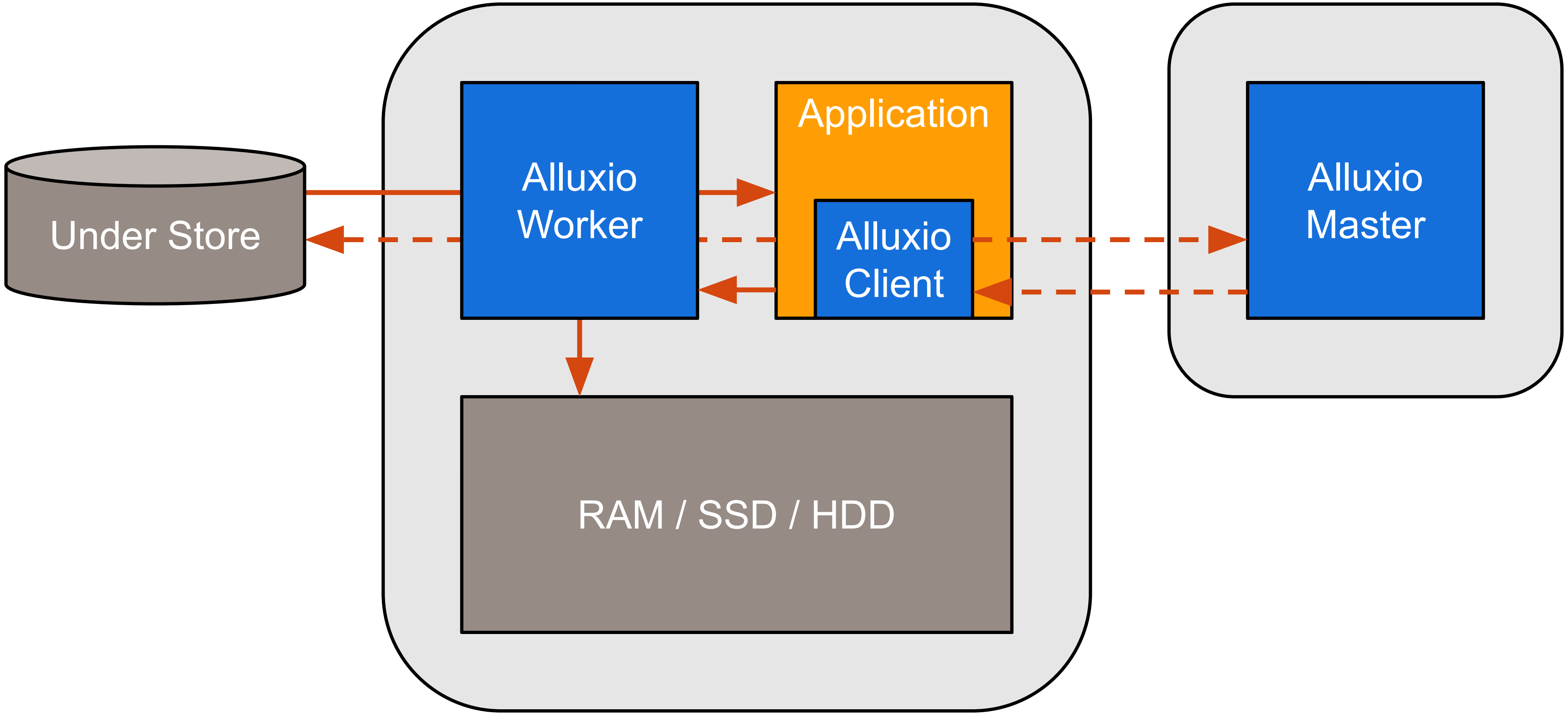
Read data in Alluxio, on same node as client

Memory Speed Read of Data



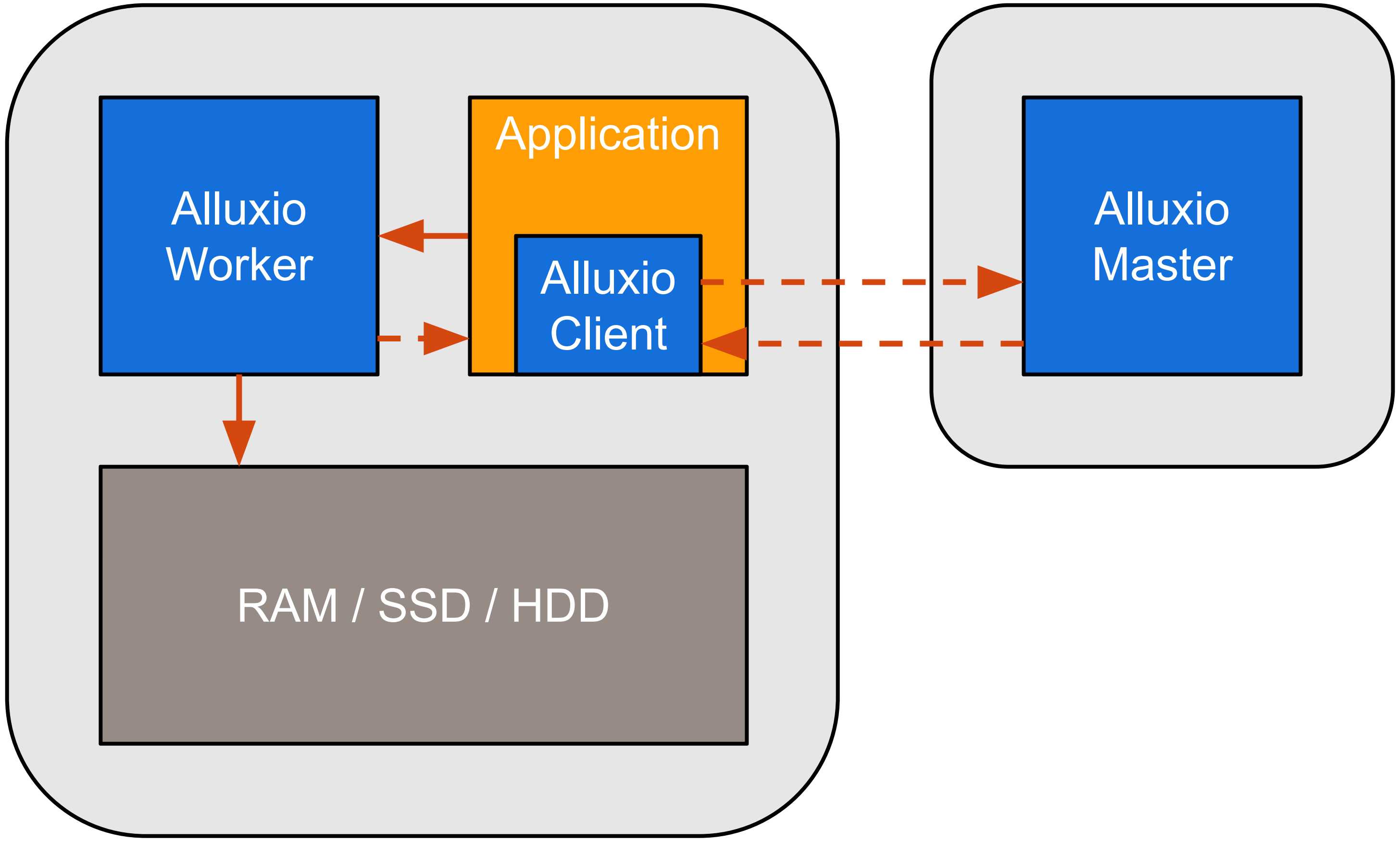
Read data not in Alluxio + Caching

Network / Disk Speed Read of Data



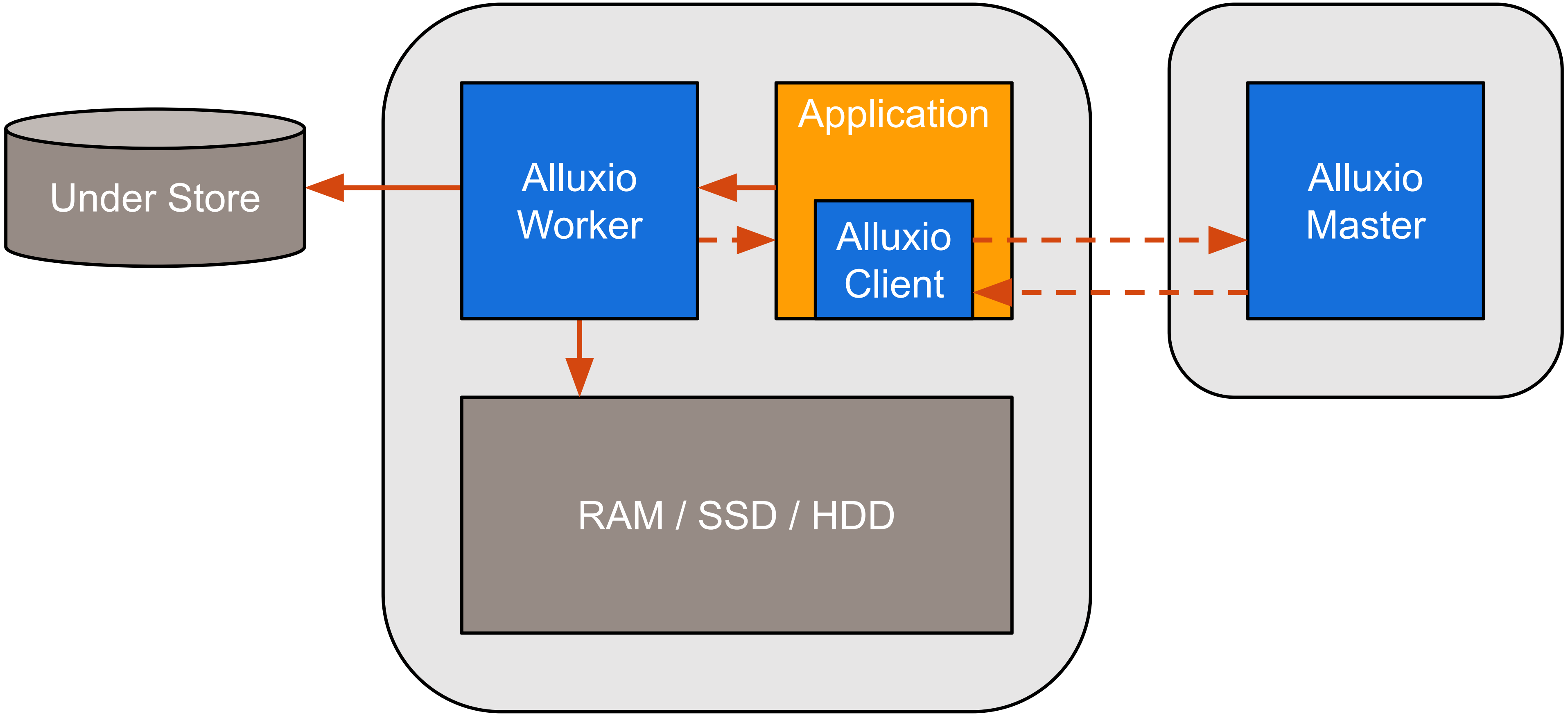
Write data only to Alluxio on same node as client

Memory Speed Write of Data



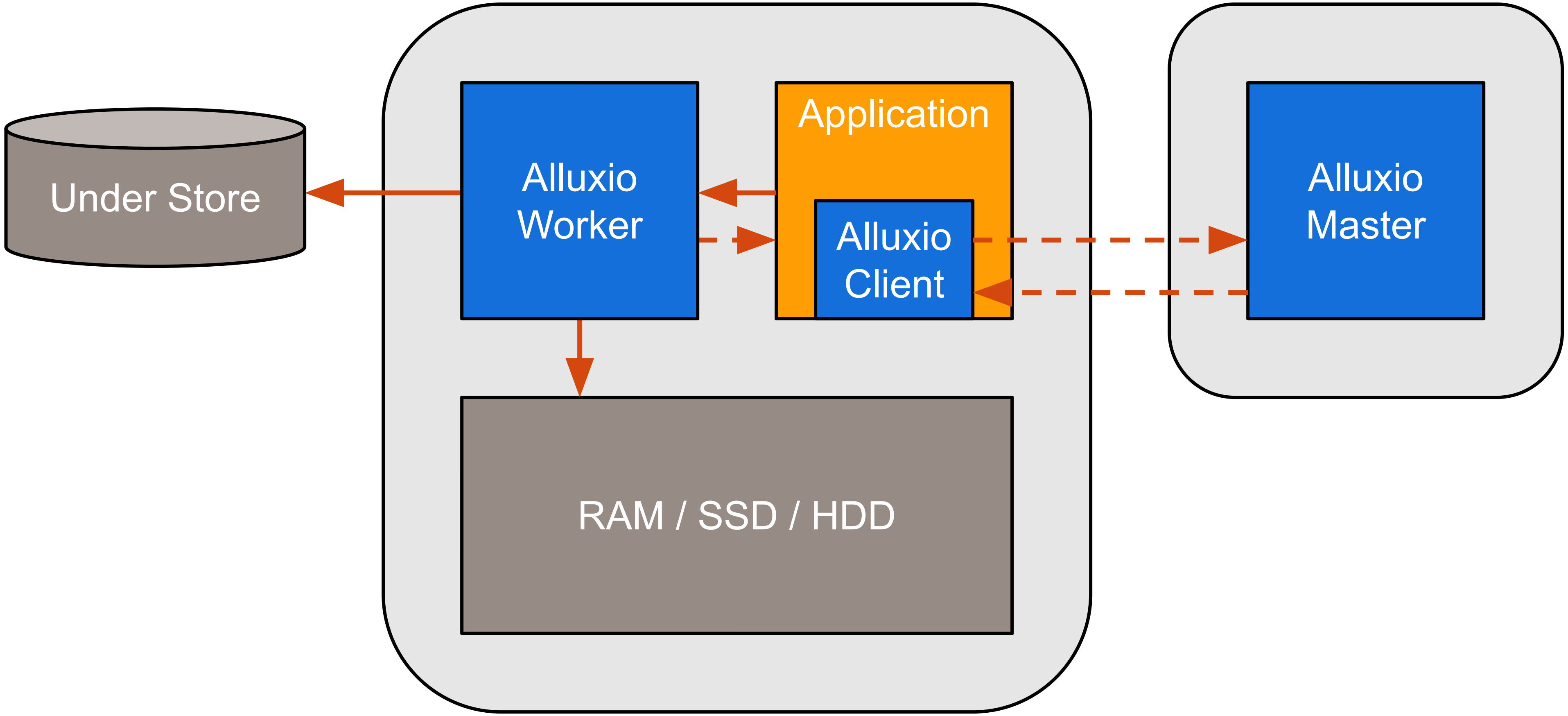
Write data to Alluxio and Under Store synchronously

Network / Disk Speed Write of Data



Write data to Alluxio, Alluxio writes it to Under Store asynchronously

Network Speed Write of Data



Architectural Improvement in 2.0 (released in June)

- Off heap metadata storage (namespace scaling)
- gRPC transport layer (cluster and client scaling)
- Improved POSIX API (new workloads)
- Job Service (enable data management)
- Embedded Journal and Internal Leader Election (better integration with object stores, fewer external dependencies)

ALLUXIO

Questions?

Welcome to join the Alluxio Open Source Community!

www.alluxio.io | @alluxio | slackin.alluxio.io