ALLUXIO

Improving Data Locality for Analytics Jobs on Kubernetes Using Alluxio

Adit Madan | Alluxio
Gene Pang | Alluxio

Outline

Alluxio Overview

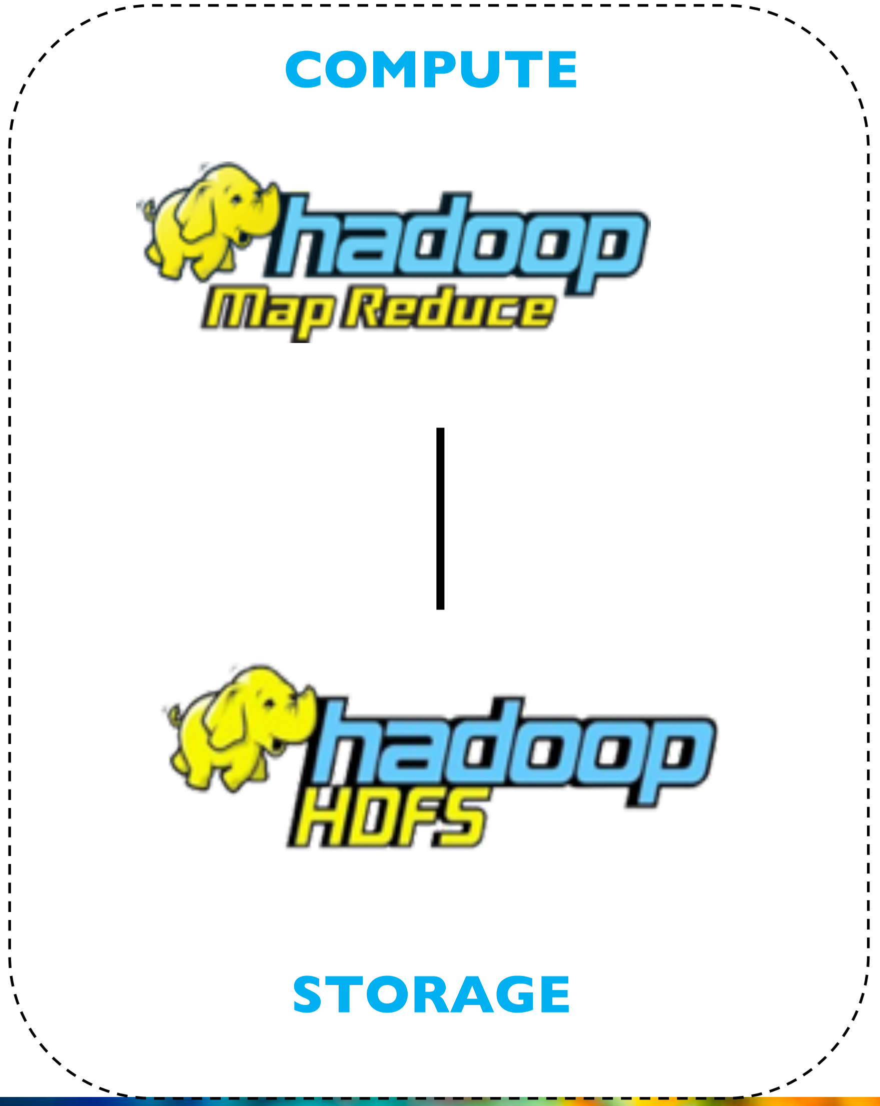
Data locality with Spark and Alluxio

Kubernetes Overview

Spark and Alluxio in Kubernetes

Alluxio Innovations for Structured Data

Data Ecosystem - *Beta*

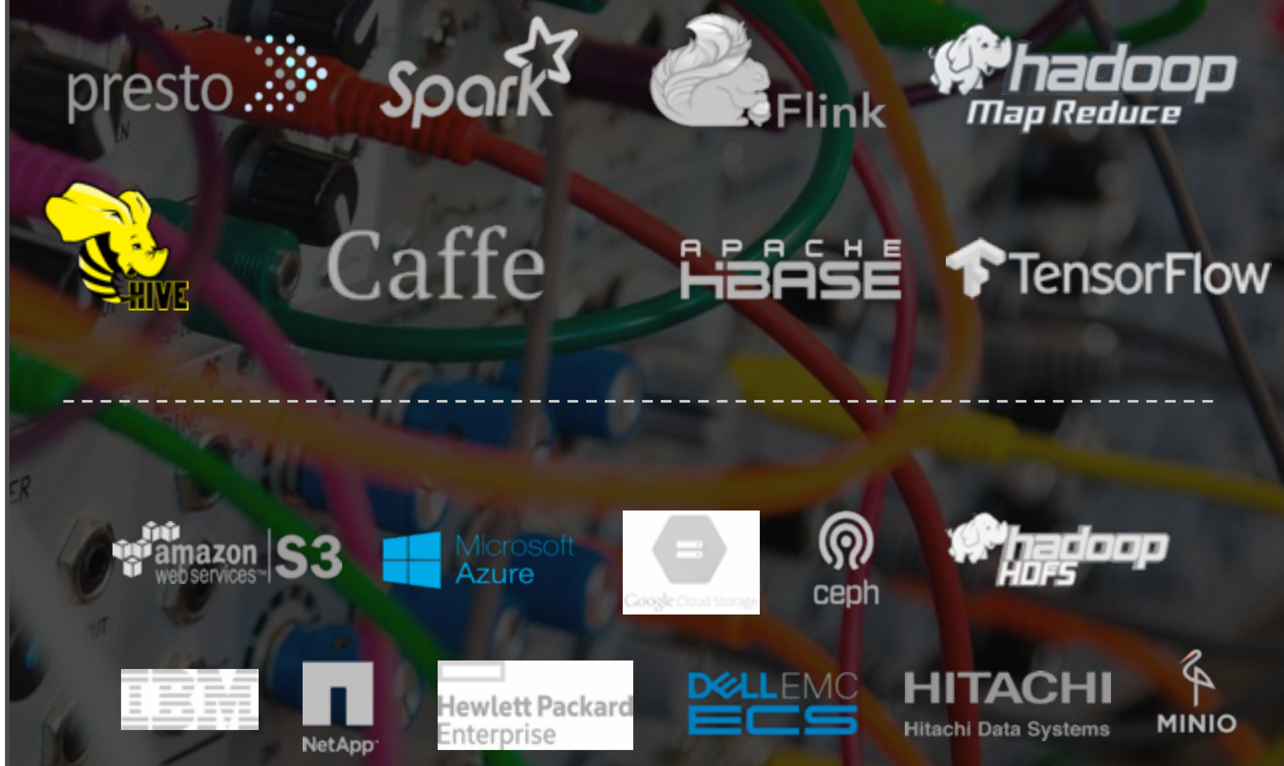


COMPUTE

STORAGE

Data Ecosystem 1.0

COMPUTE

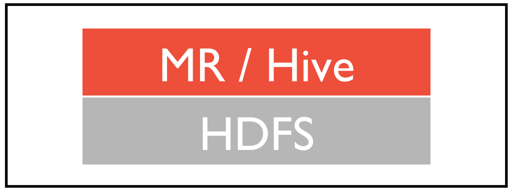


STORAGE

Data stack journey and innovation paths

Co-located

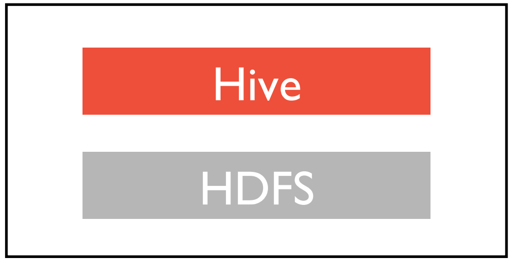
Co-located compute & HDFS on the same cluster



- 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

HDFS for Hybrid Cloud

Burst HDFS data in the cloud, public or private

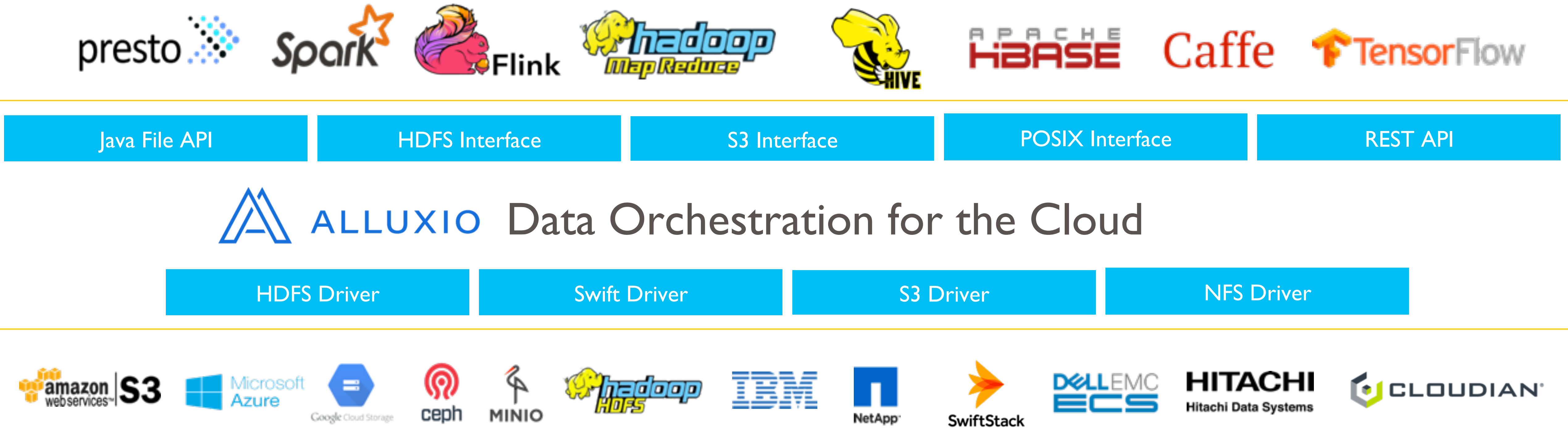
Support more frameworks

Support Presto, Spark and other computes without app changes

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

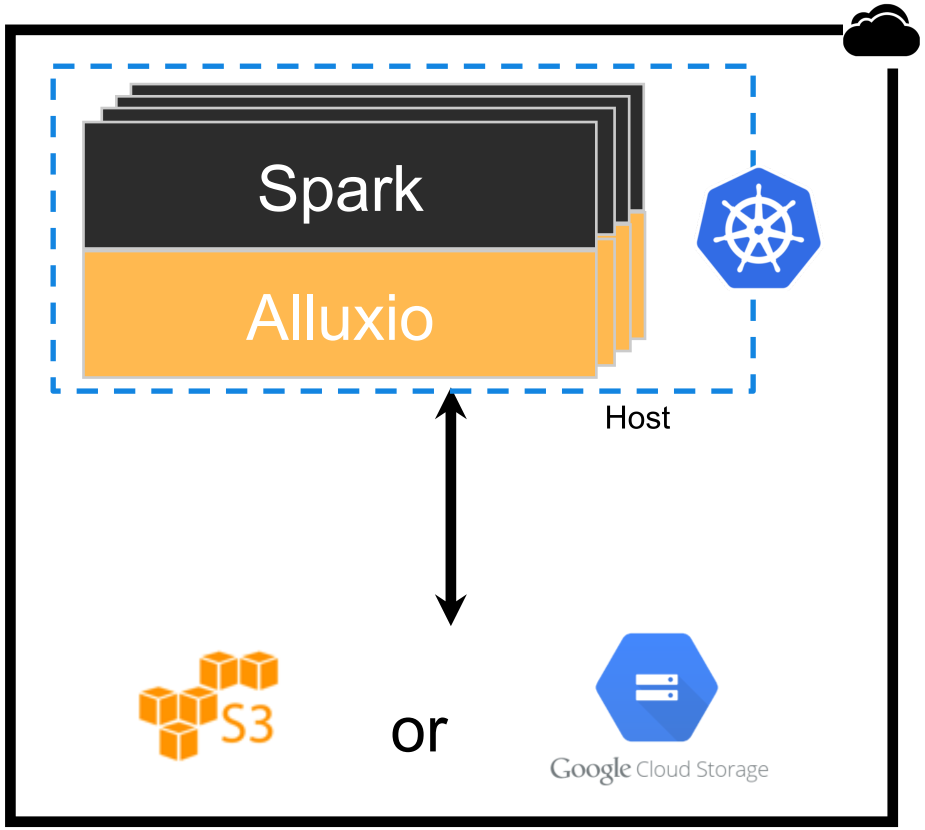
Alluxio Data Orchestration for the Cloud

- Global Namespace
- Intelligent Caching
- Data Management
- Structured Data Catalog
- Data Transformation

Why Alluxio in K8s?

Elastic Data for Elastic Compute

- Improve Data locality
  - Big data analytics or ML
  - Cache data close to compute
- Enable high-speed data sharing across jobs
  - A staging storage layer
- Unification of persistent storage
  - Data abstraction across different storage

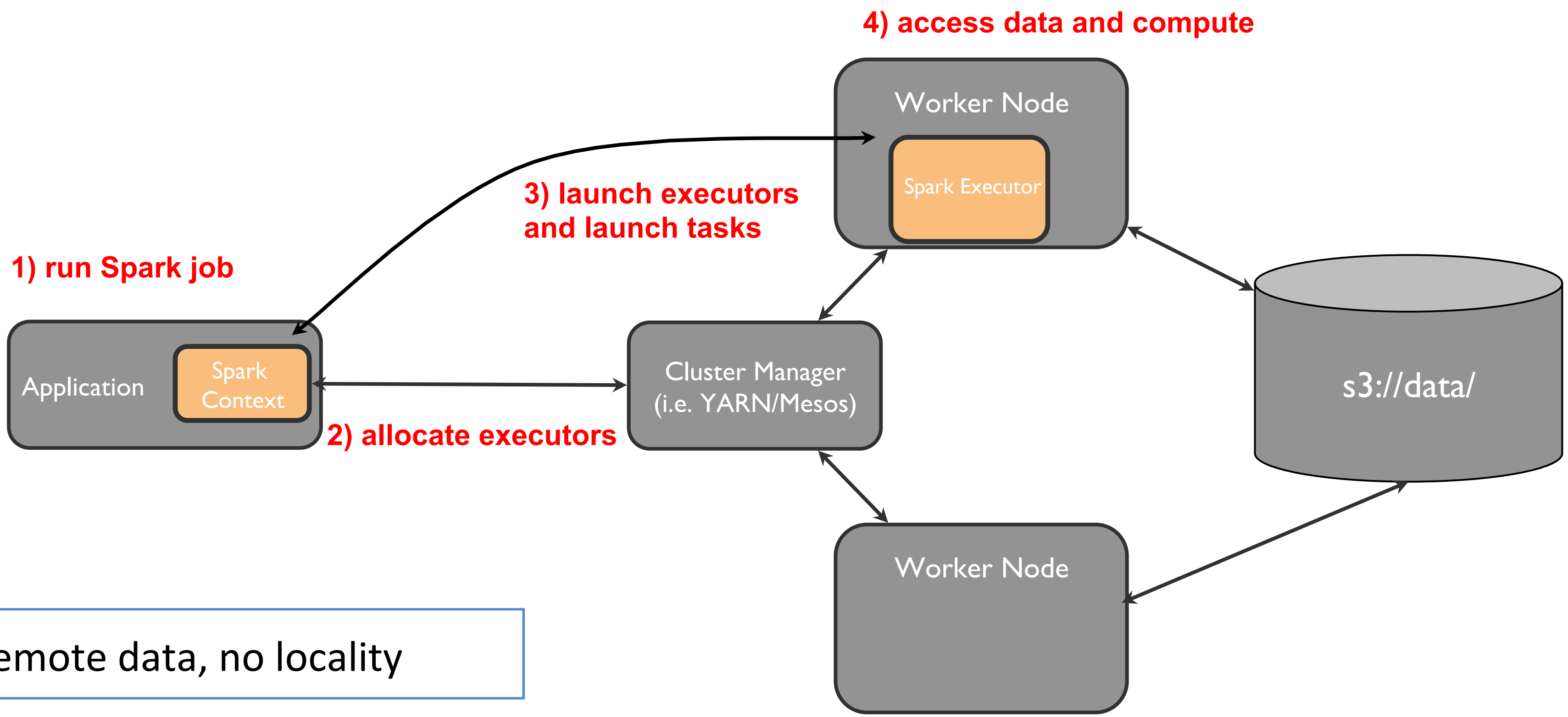


Read more at https://www.alluxio.io/blog/kubernetes-alluxio-and-the-disaggregated-analytics-stack/

Spark + Alluxio Data Locality

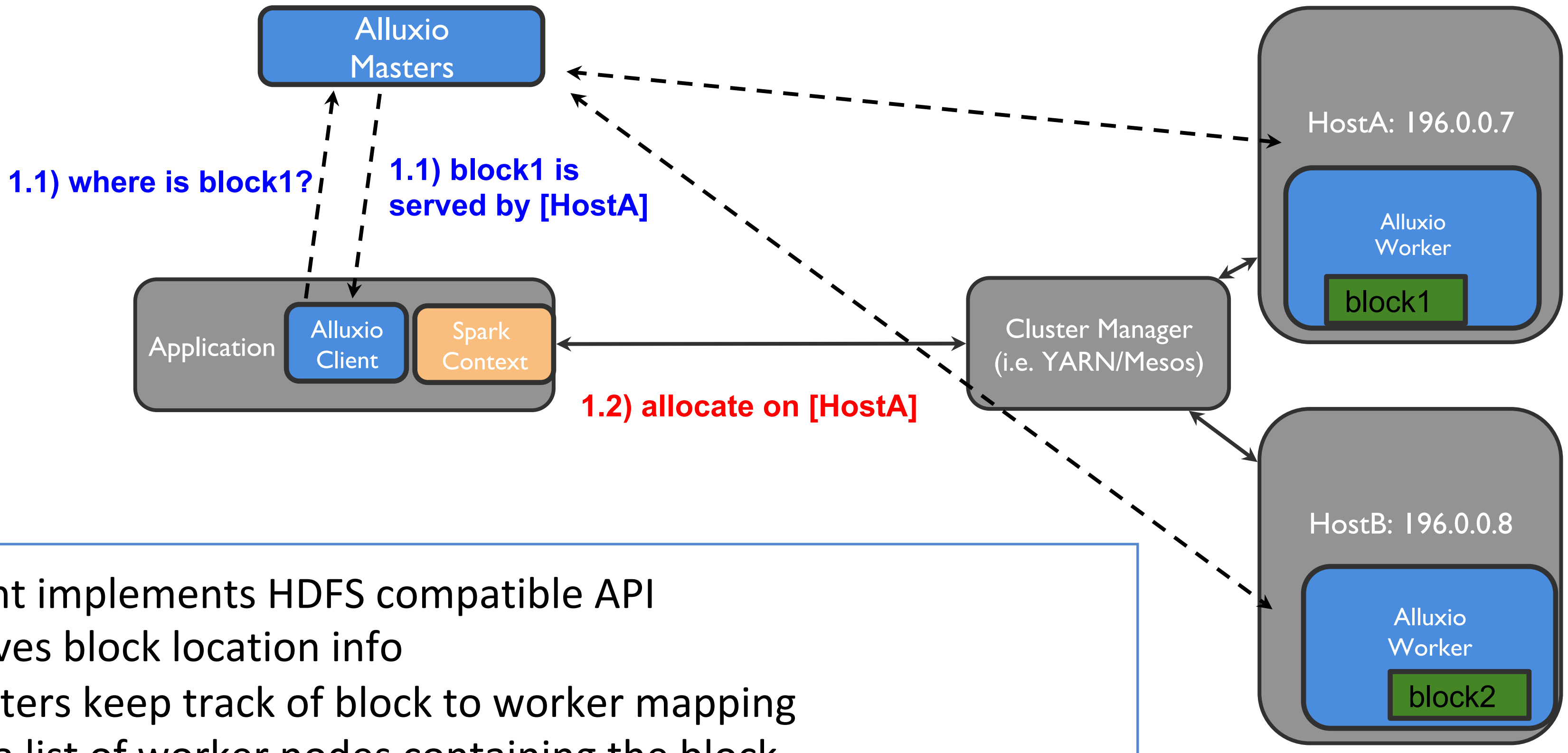
Without K8s

Spark Workflow (w/o Alluxio)



Takeaway: Remote data, no locality

Step 1: Schedule Compute to Data Location



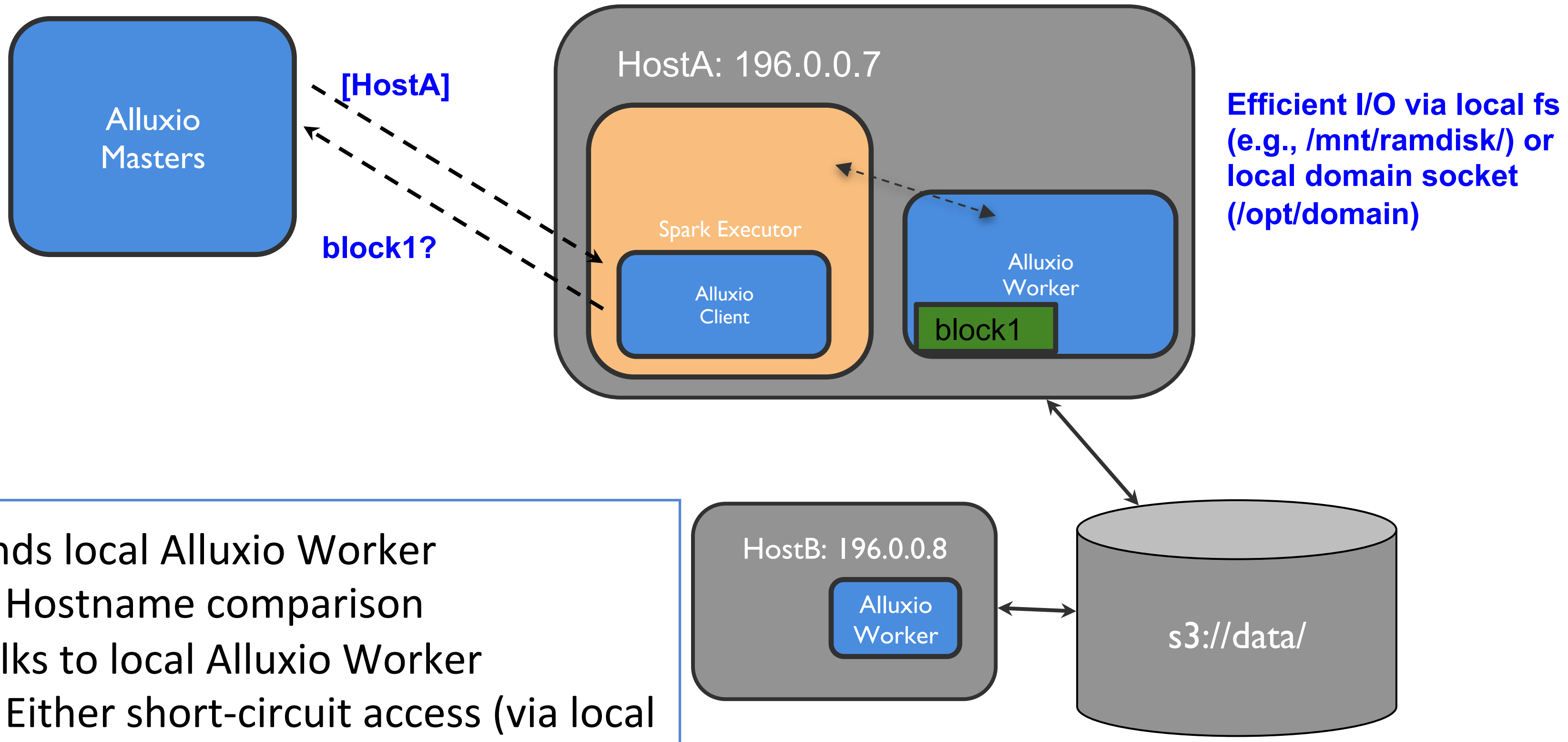
Alluxio client implements HDFS compatible API

Retrieves block location info

Alluxio masters keep track of block to worker mapping

Serve a list of worker nodes containing the block

Step 2: Detect+Exchange Data w/ local Worker



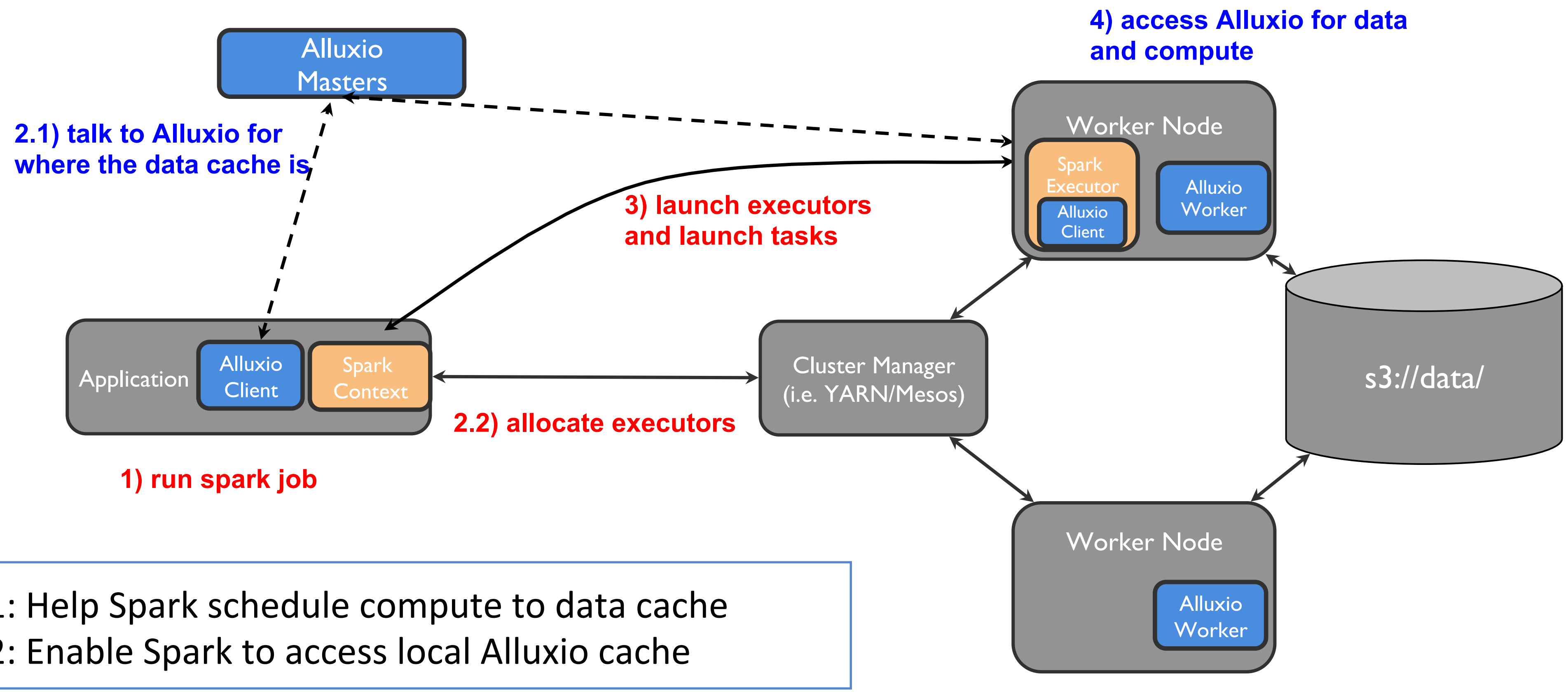
Spark Executor finds local Alluxio Worker

Mechanism: Hostname comparison

Spark Executor talks to local Alluxio Worker

Mechanism: Either short-circuit access (via local FS) or local domain socket

Recap: Spark Architecture w/ Alluxio

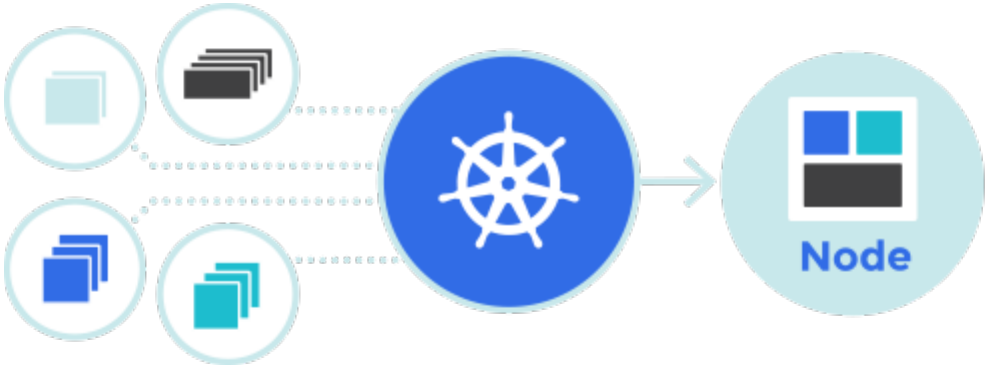


Step 1: Help Spark schedule compute to data cache
Step 2: Enable Spark to access local Alluxio cache

Kubernetes Overview

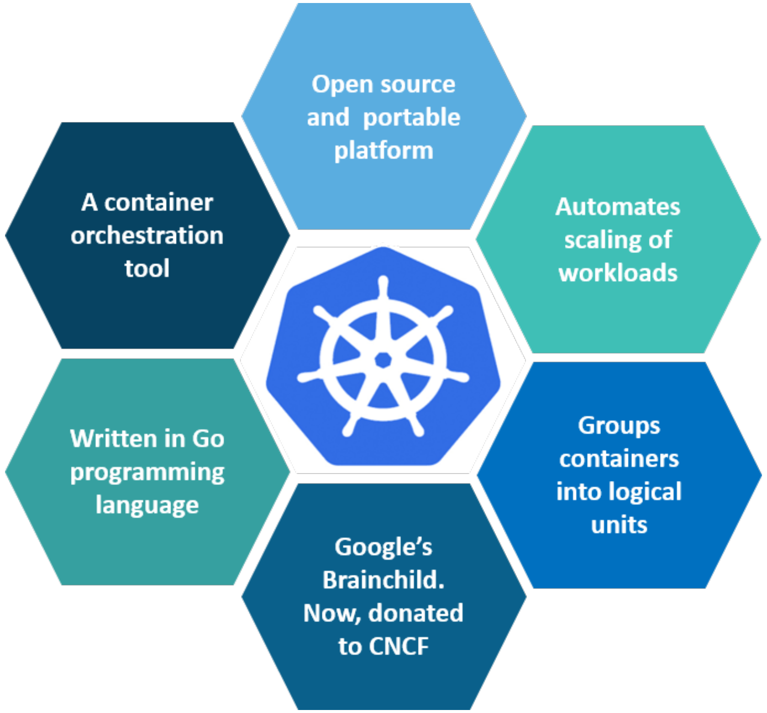
Kubernetes (K8s) is…

“an open-source container-orchestration system for automating application deployment, scaling, and management.”



Container Orchestration

- Platform agnostic cluster management
- Service discovery and load balancing
- Storage orchestration
- Horizontal scaling
- Self-monitoring and self-healing
- Automated rollouts and rollbacks


Open source and portable platform
A container orchestration tool
Automates scaling of workloads
Written in Go programming language
Groups containers into logical units
Google's Brainchild. Now, donated to CNCF

Key K8s Terms

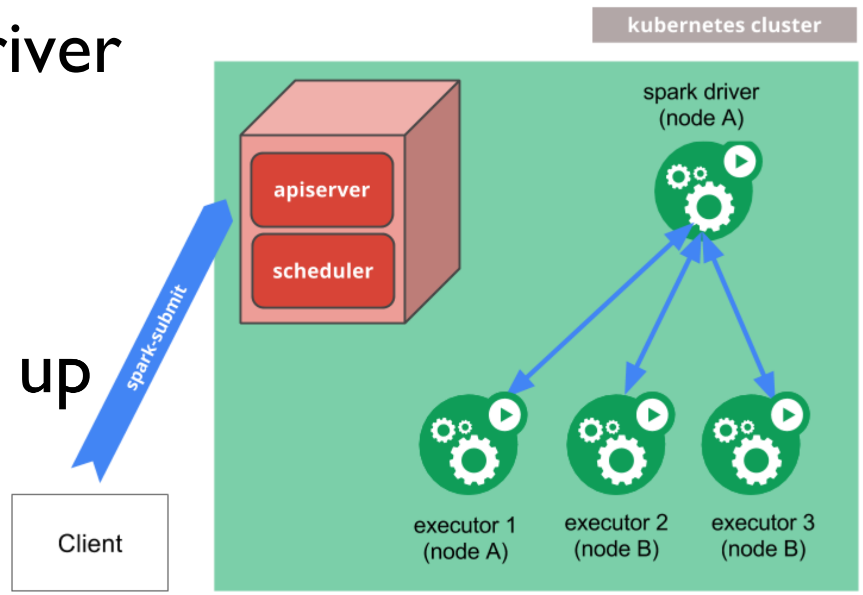
- Node
  - A VM or physical machine
- Container
  - Container = Image once running on Docker Engine
- Pod
  - Schedulable unit of one or more containers running together
- Controller
  - Controls the desired state such as copies of a Pod
- DaemonSet
  - A Controller that ensures each Node has only one such Pod
- Persistent Volume
  - A storage resource with lifecycle independent of Pods

Spark + Alluxio Data Locality

In K8s environment

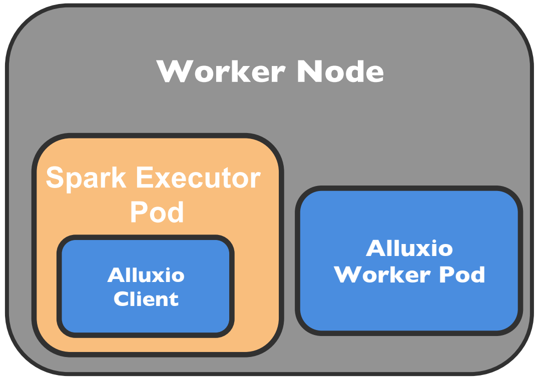
Spark on K8s Architecture

- Spark 2.3 added native K8s support
- *spark-submit* talks to API Server to launch Driver
- Spark Driver launches Executor Pods
- When the application completes,
  - Executor Pods terminate and are cleaned up
  - Driver Pod persists logs and remains in "COMPLETED" state

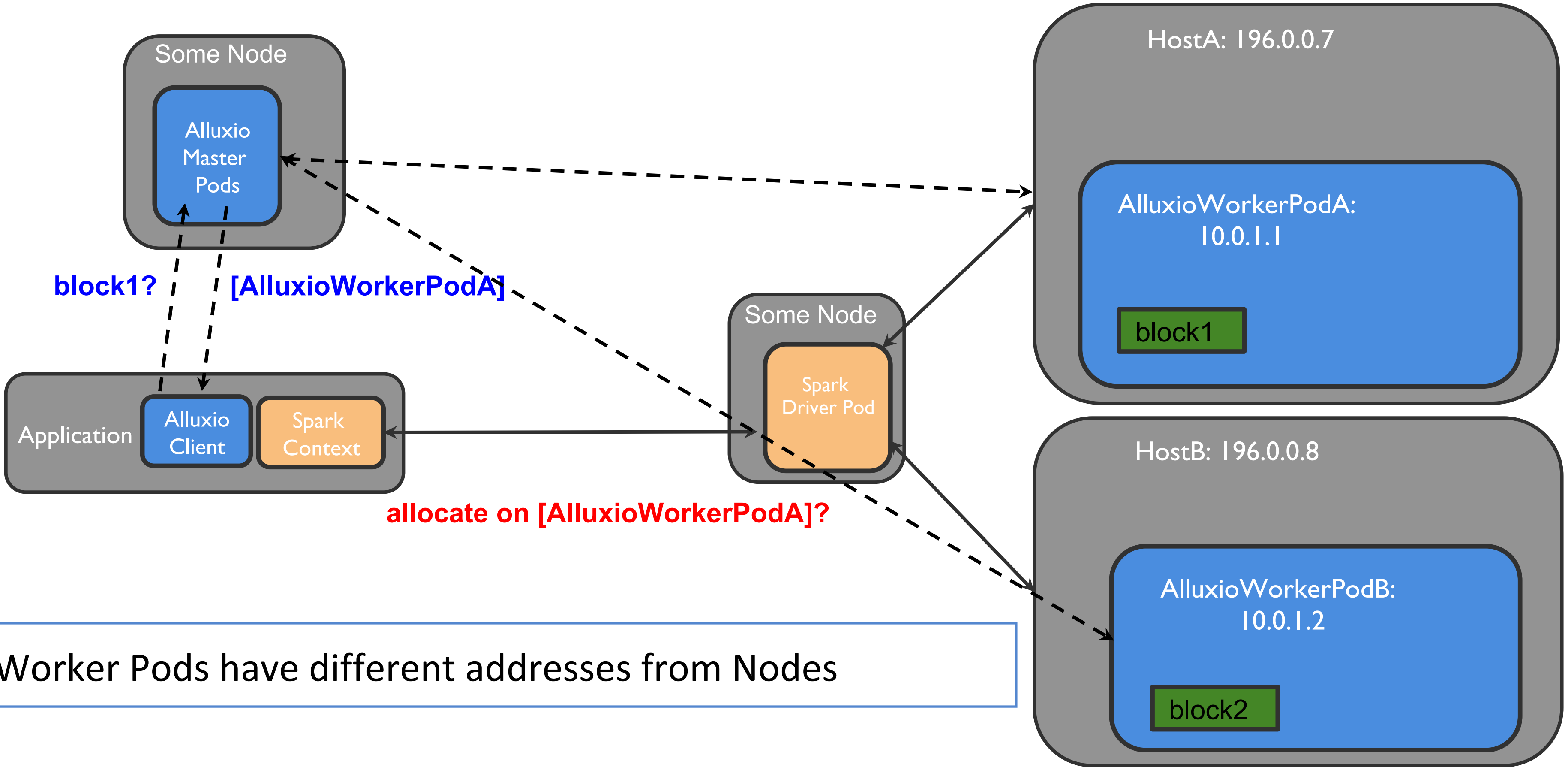


Deployment Model: Co-location

- Co-locate Spark Executor Pod w/ Alluxio Worker Pod
- Lifecycle
  - Spark Executors are ephemeral
  - Alluxio Workers persist across all Spark jobs
- Deployment order:
  - Deploy Alluxio cluster first (masters+workers)
  - An Alluxio Worker on each Node, by DaemonSet
  - *spark-submit* launches Spark Driver + Executors

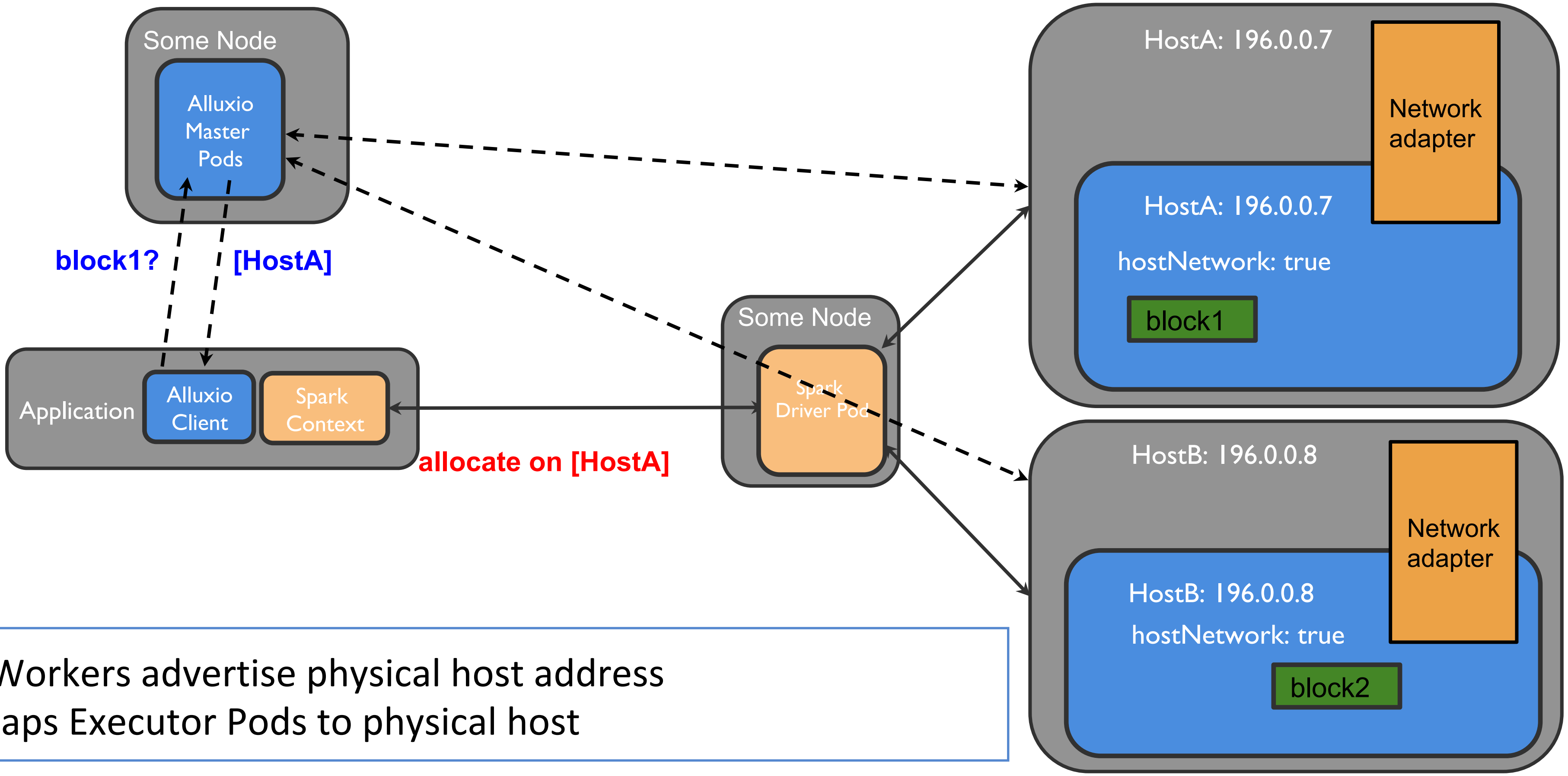


Challenge 1: Executor Allocation to Workers



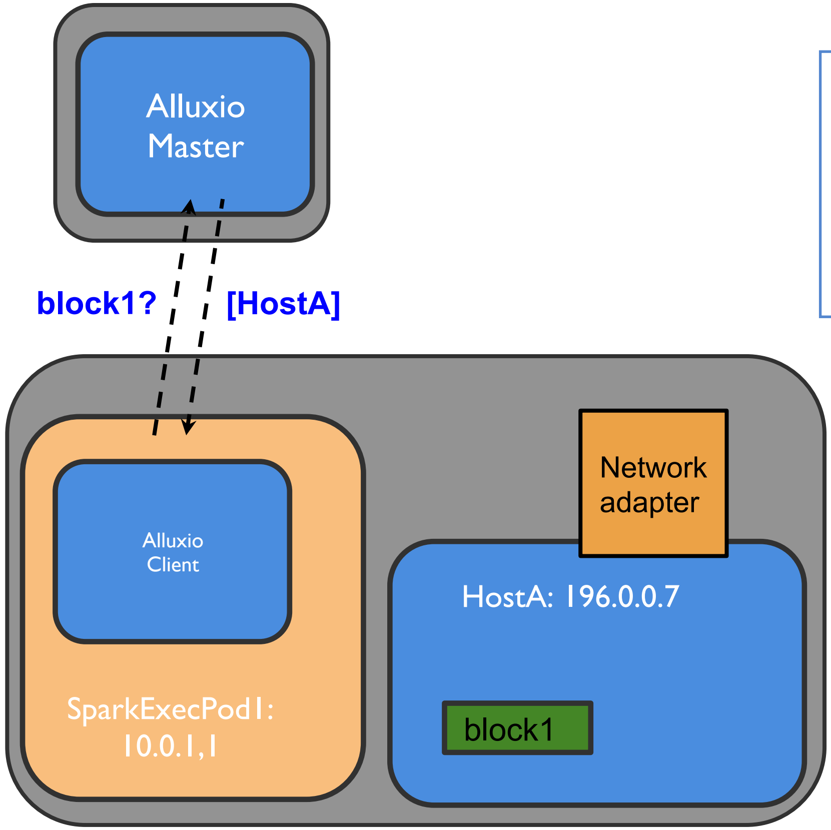
Problem: Worker Pods have different addresses from Nodes

Solution: Alluxio Workers w/ ***hostNetwork***



Alluxio Workers advertise physical host address
Spark maps Executor Pods to physical host

Challenge 2: Identify host-local Alluxio Pod

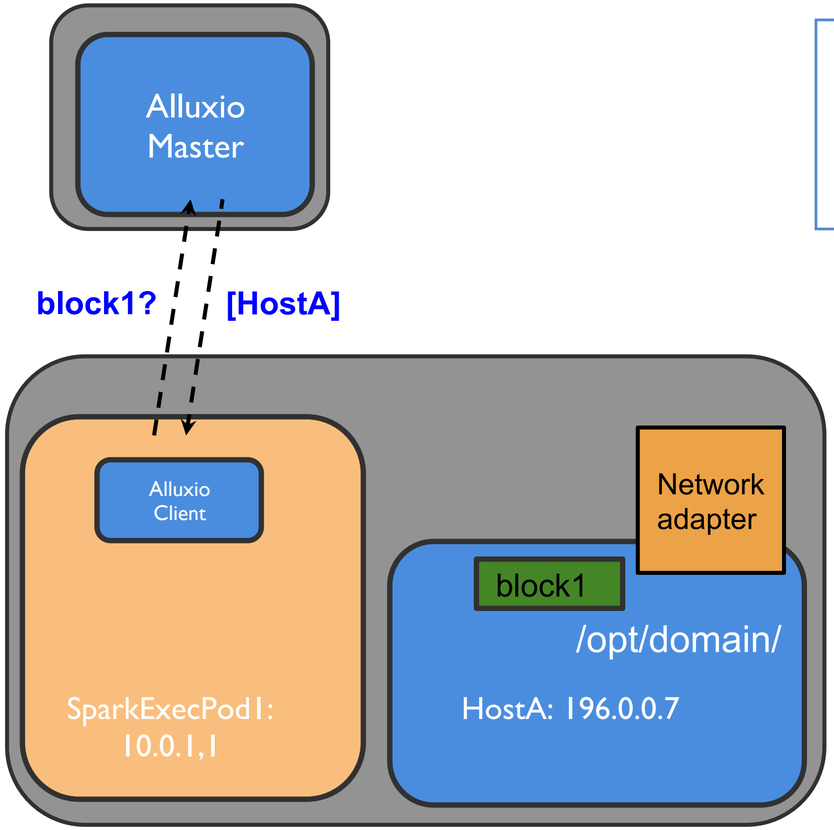


Problem:
Spark Executor Pod has different hostName

Hostname HostA != SparkExecPod1

Local Alluxio Worker not identified!

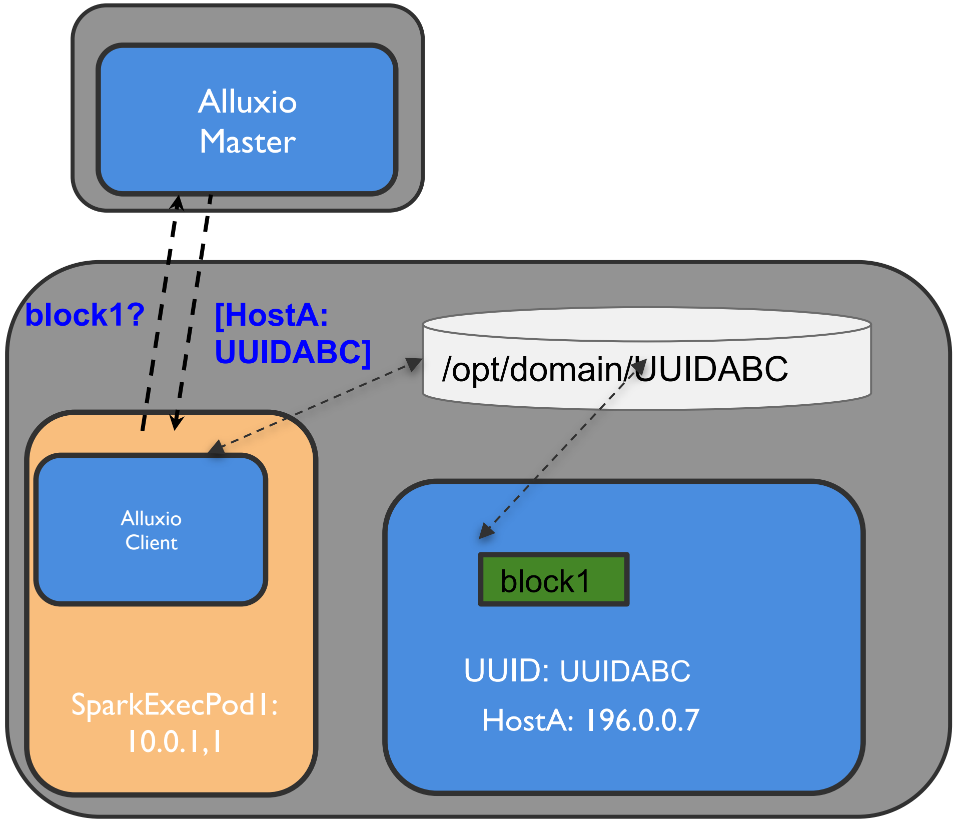
Challenge 3: Executor fails to find domain socket



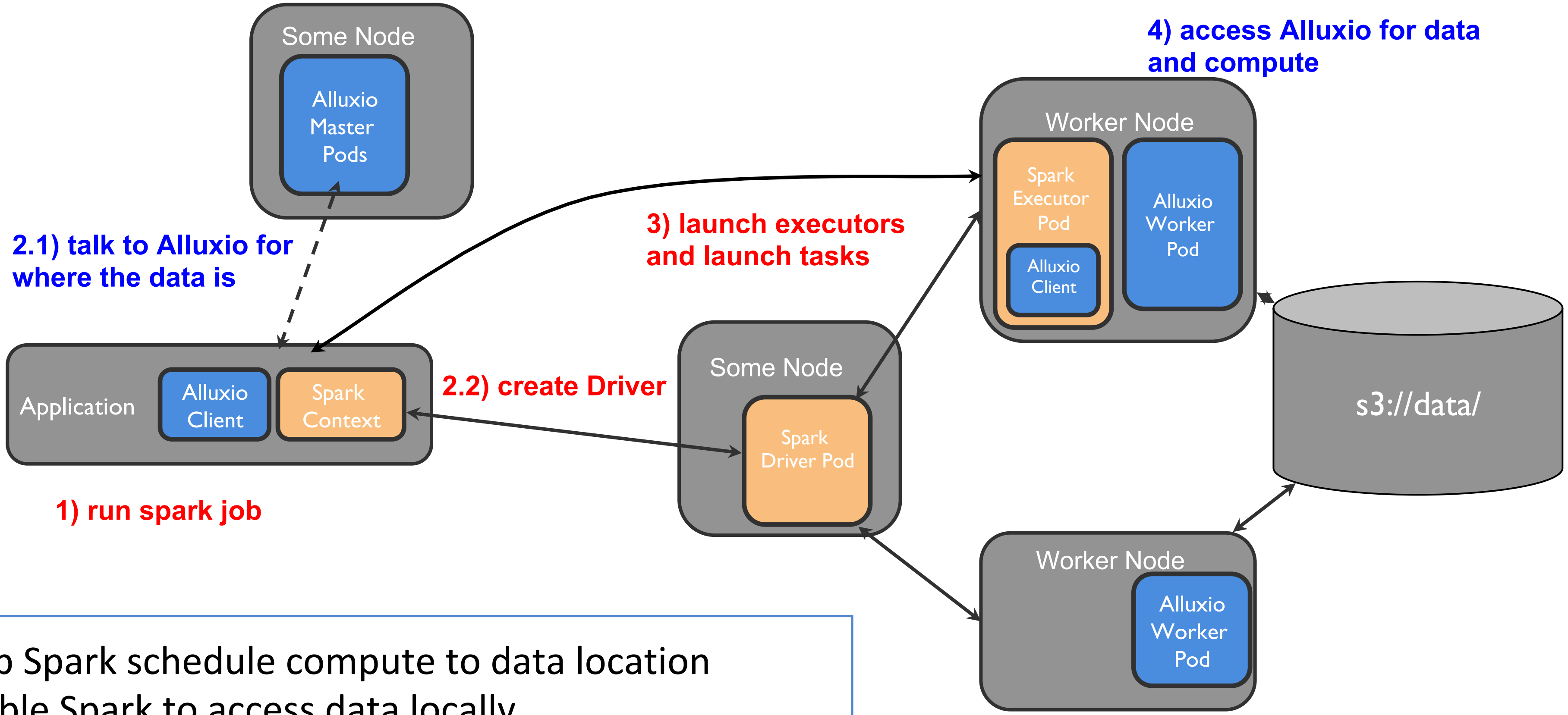
Problem:
- Pods don’t share the File System
- Domain socket */opt/domain* is in Alluxio Worker Pod

Solution: Share a ***hostPath*** Volume b/w Pods

- Each Alluxio Worker has a UUID
- Share domain socket by a *hostPath* Volume
- Alluxio Client finds local worker's domain socket by finding file matching Worker UUID
- Worker domain socket path
  - */opt/domain/d -> /opt/domain/UUIDABC*
- Mount hostPath Volume to Spark Executor
  - Enabled in Spark 2.4



Recap: Spark + Alluxio Architecture on K8s



Step 1: Help Spark schedule compute to data location
Step 2: Enable Spark to access data locally

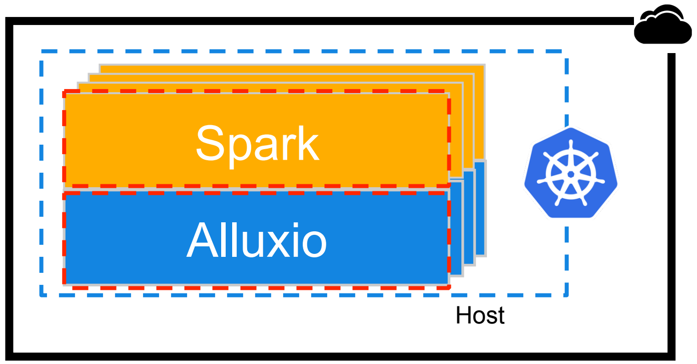
Limitations and Future Work

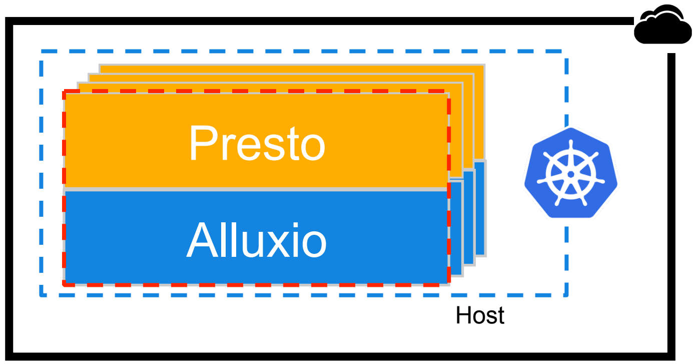
- Enterprise environments may restrict *hostNetwork* and *hostPath*
- Alluxio workers need *hostNetwork*
  - Plan: Support container network translation
- The domain socket file requires a *hostPath* volume
  - Plan: Using Local Persistent Volumes
- Feedback/collaboration are welcome!

ALLUXIO

Alternate Deployment Options

Deploying Alluxio in K8s





Legend: Pod

Alluxio and Compute in different pods on the same host

When do you use this?

- Compute, like Spark, is short running and ephemeral
- Alluxio data orchestration & access layer is long running and used across many jobs

Alluxio and Compute framework in the same pod

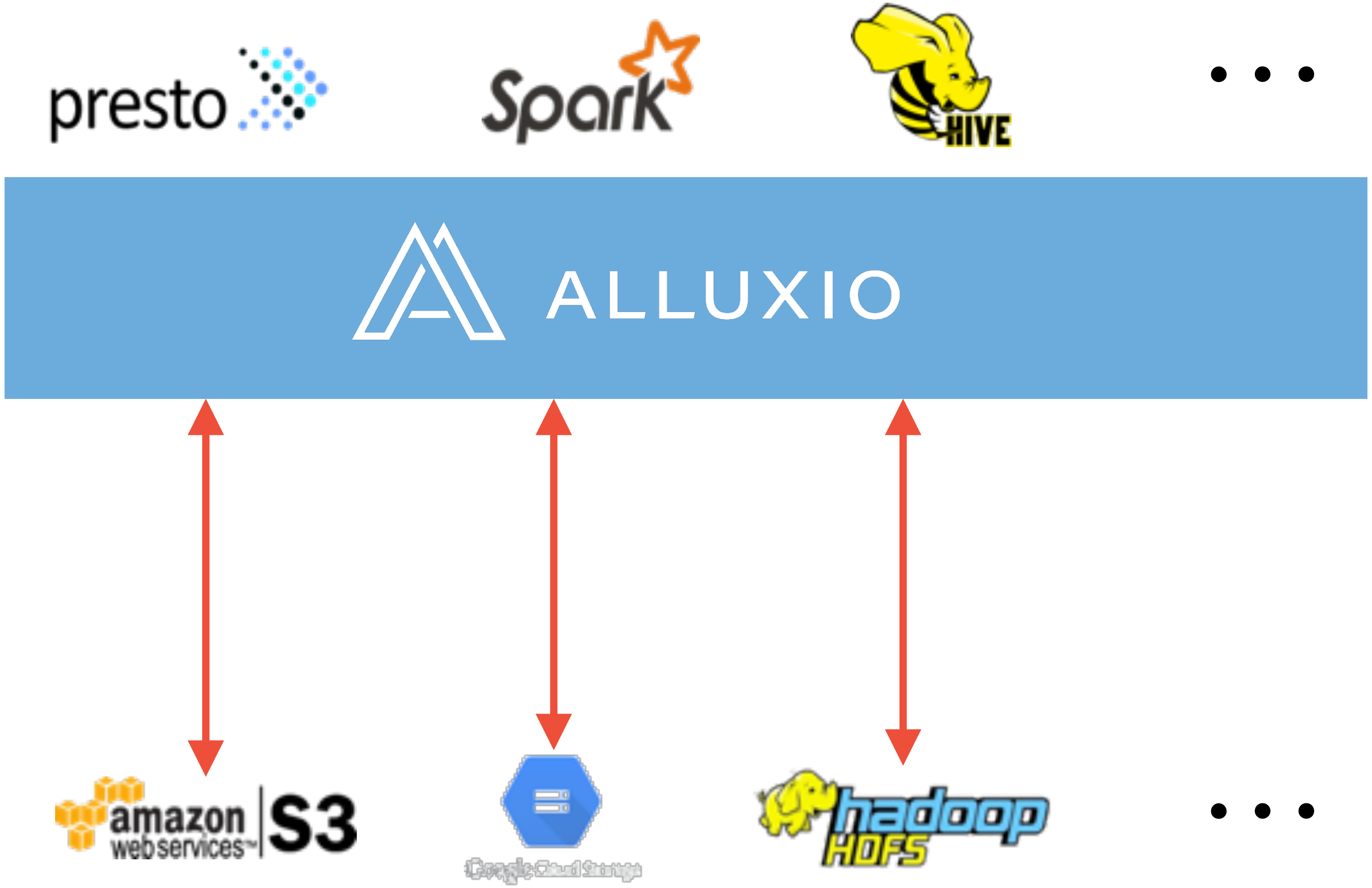
When do you use this?

- Compute, like Presto, is long running
- Data tier with Alluxio needs to be scaled along with compute tier

Alluxio
Structured Data Management

Innovations for Structured Data

Common Alluxio Use Cases



SQL Engines are popular

Unified Interface

Unified Namespace

Caching and Locality

Storage Systems

Files/Objects

Directories

Raw Bytes

Storage Optimized

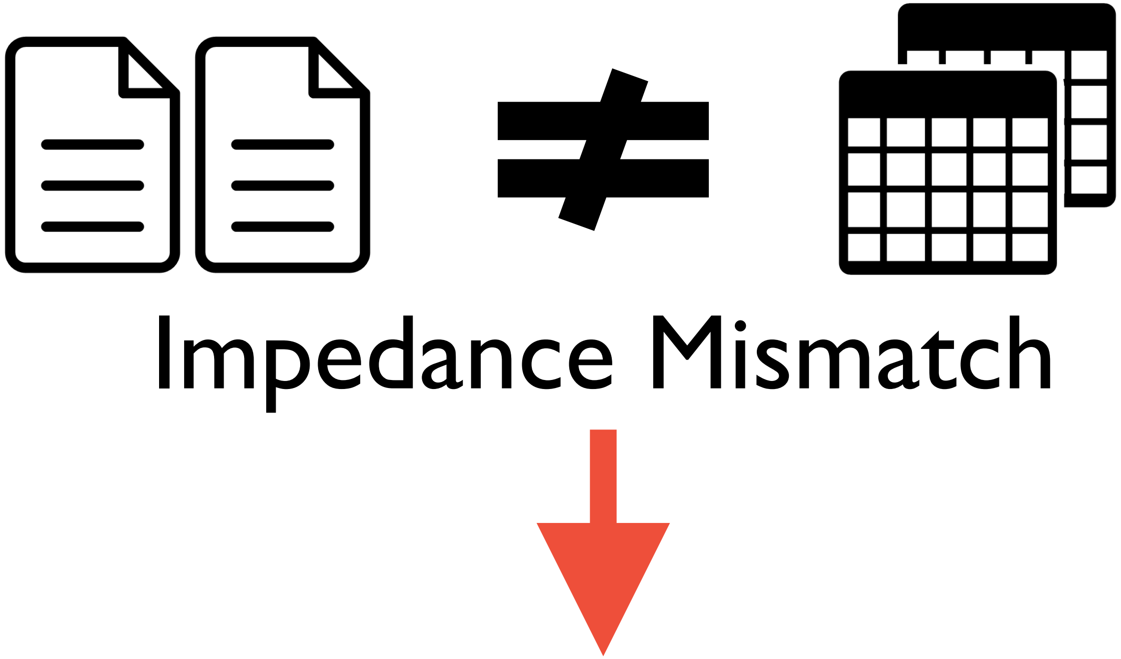
SQL Frameworks

Tables

Schemas

Rows/Columns

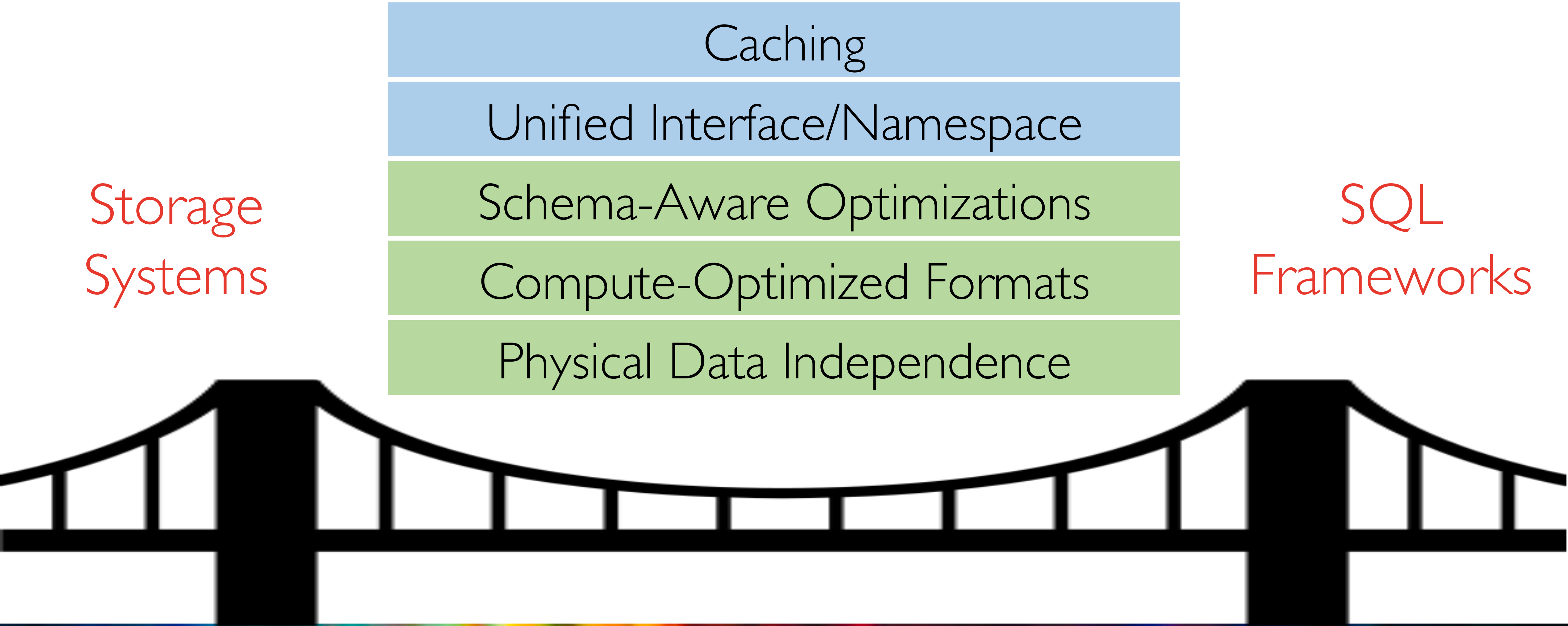
Compute Optimized



Impedance Mismatch

Further Expand Benefits!

Benefits of Alluxio Data Orchestration



Storage Systems

Caching

Unified Interface/Namespace

Schema-Aware Optimizations

Compute-Optimized Formats

Physical Data Independence

SQL Frameworks

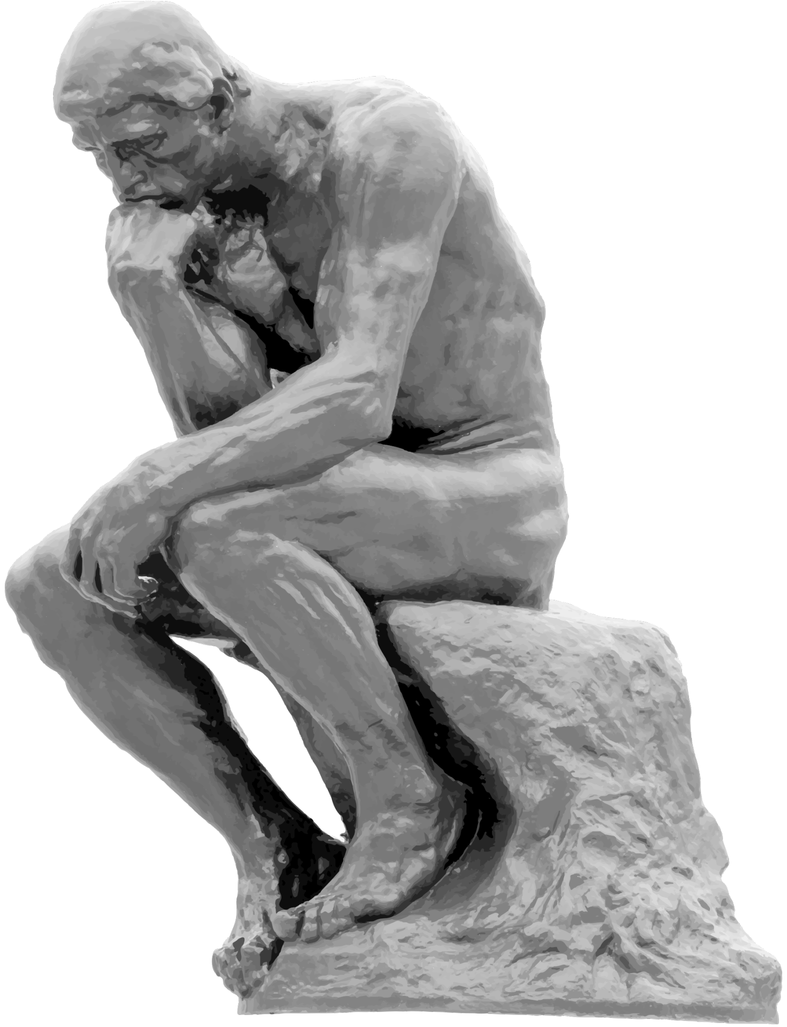
High-Level Philosophy

Provide Structured Data APIs

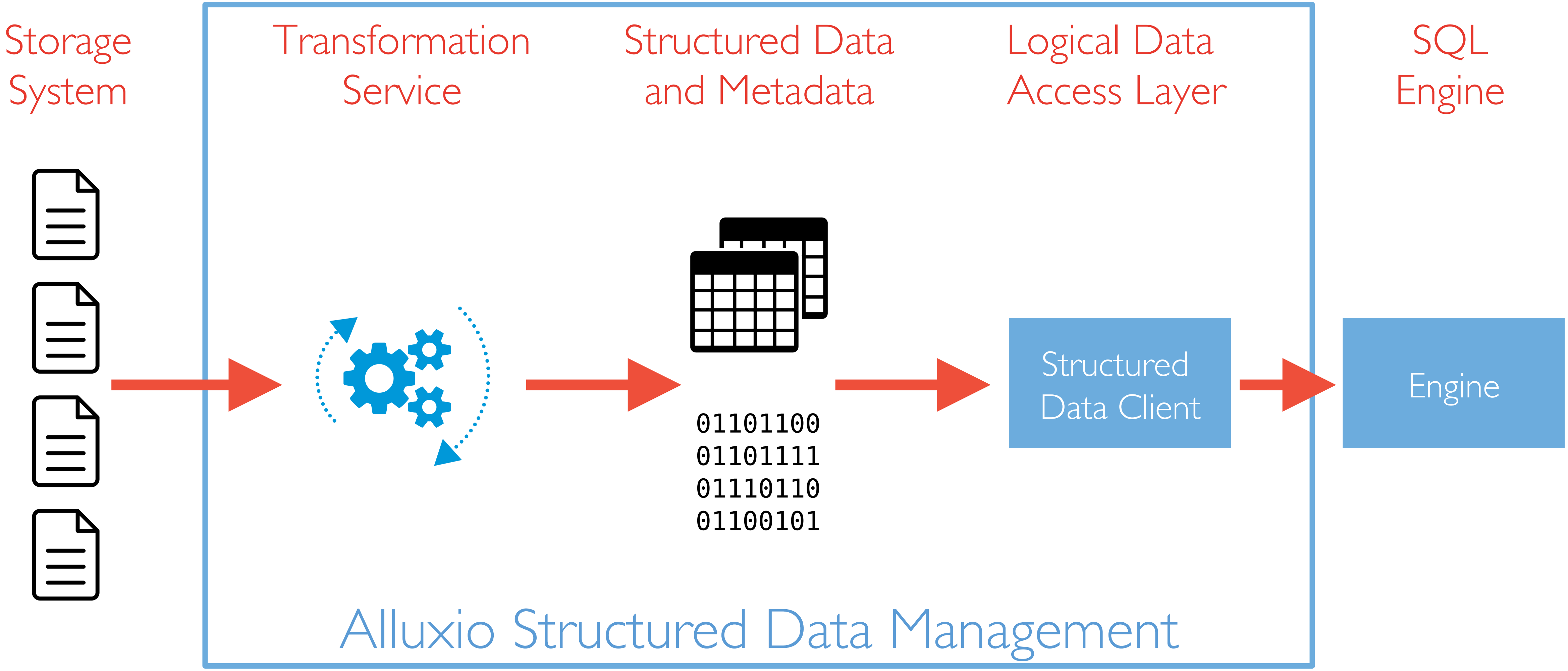
Focus on how frameworks interact with data

Cache Logical Data Access

Focus on caching what frameworks want

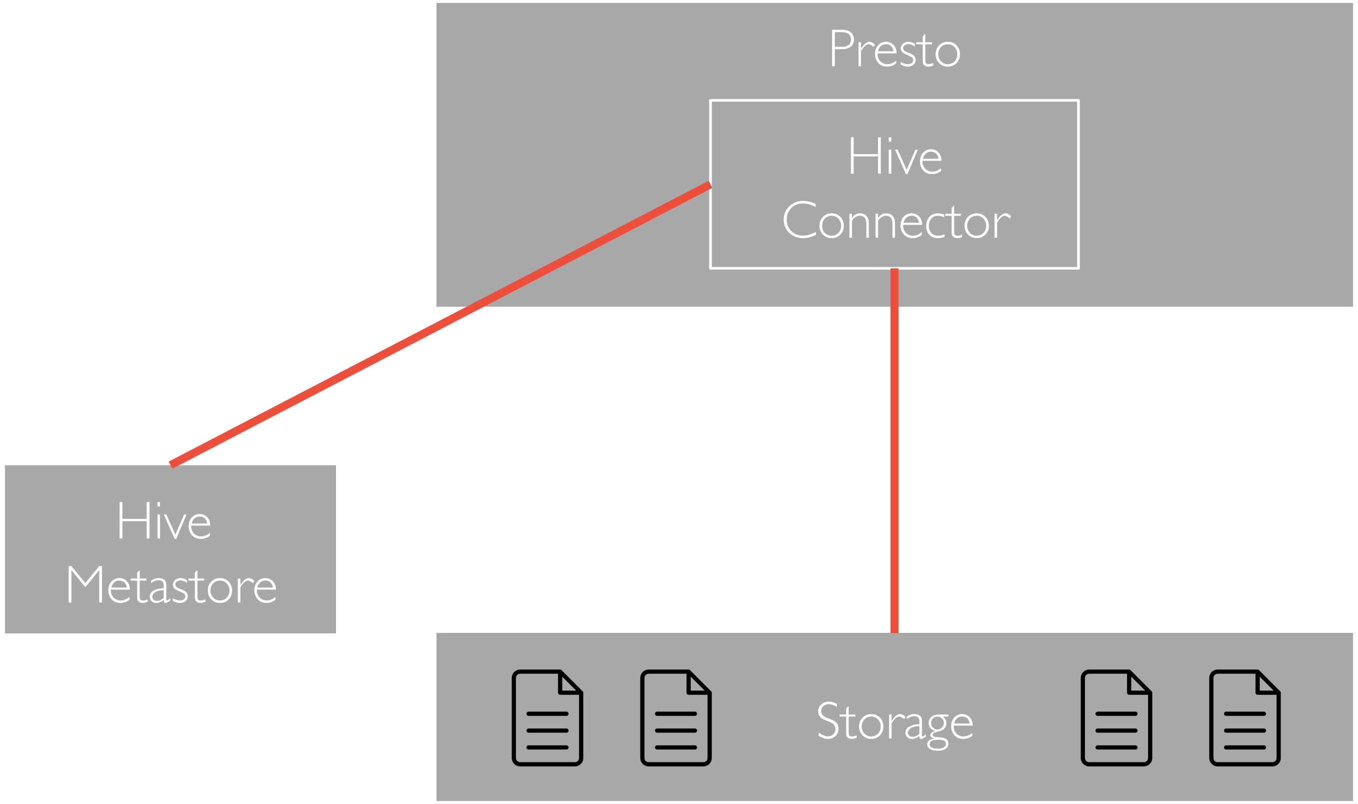


Alluxio Structured Data Management

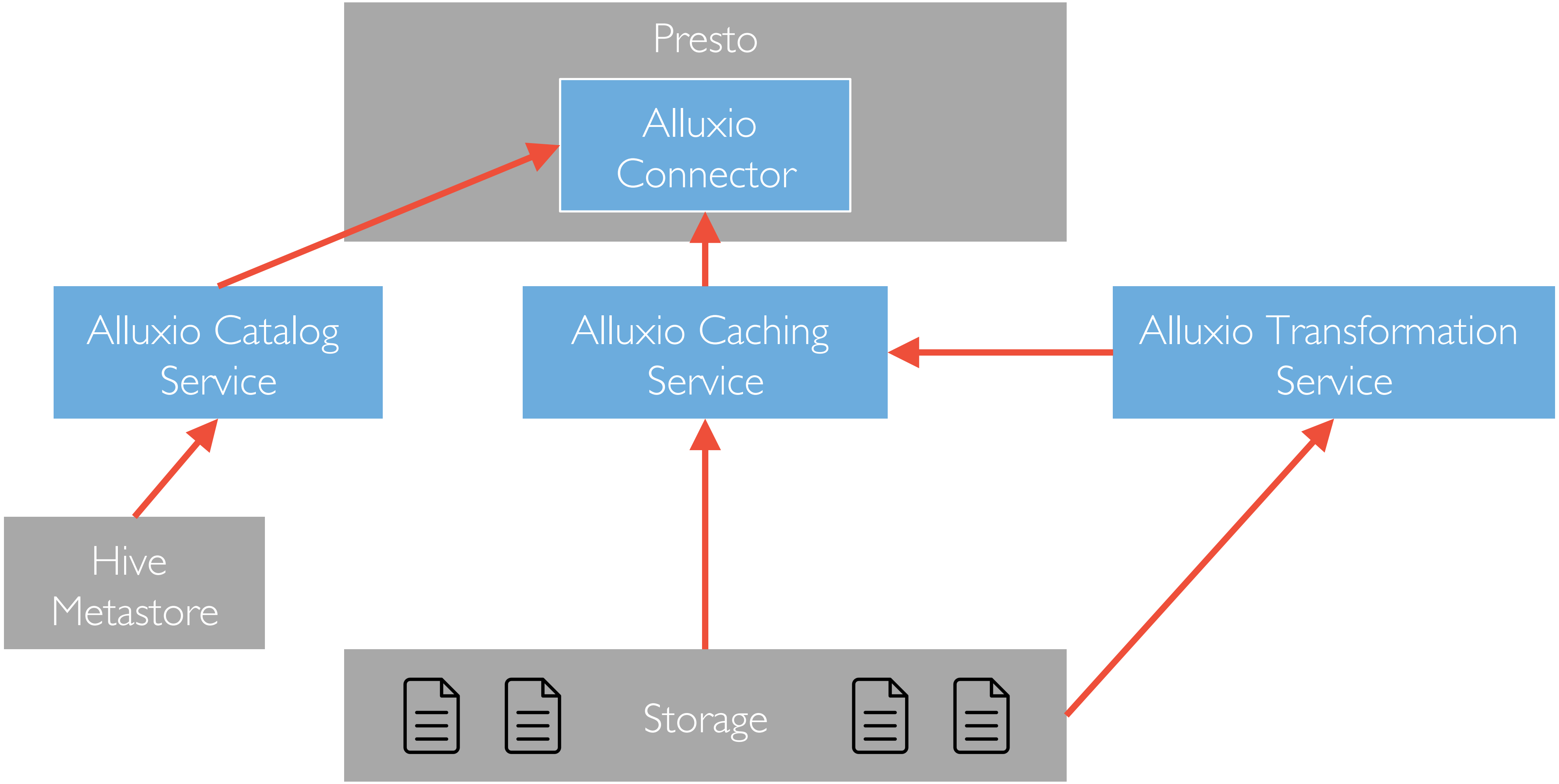


Developer Preview in Alluxio 2.1

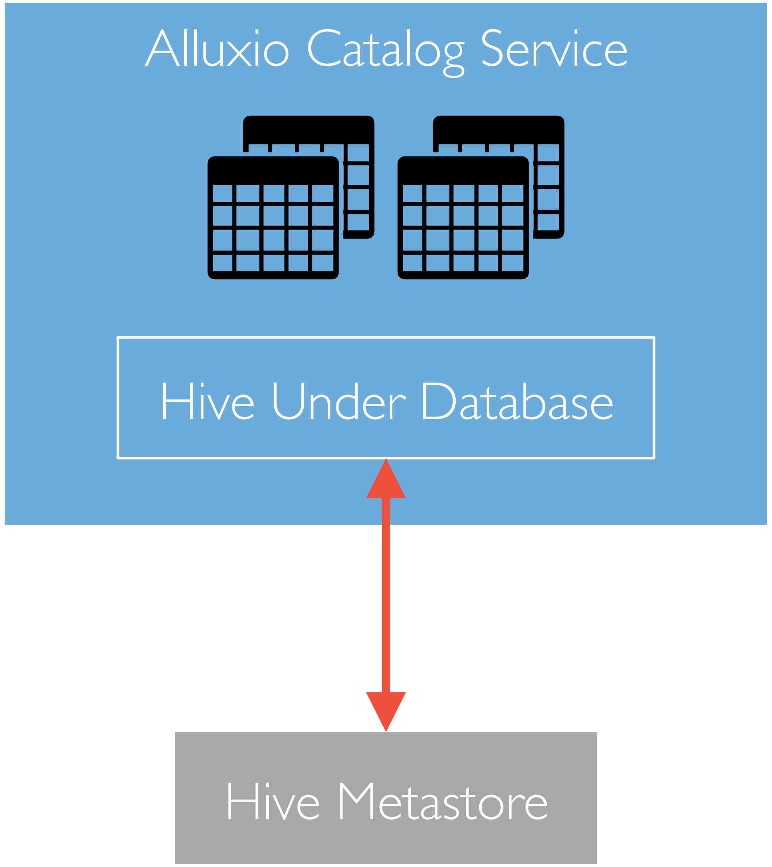
Target Environment



Alluxio Structured Data Management



Alluxio Catalog Service



Functionality

Manages metadata for structured data

Abstracts other database catalogs as Under Database (UDB)

Benefits

Schema-aware optimizations

Simple deployment

Alluxio Presto Connector

Tighter integration with Presto

New plugin based on the Presto Hive connector

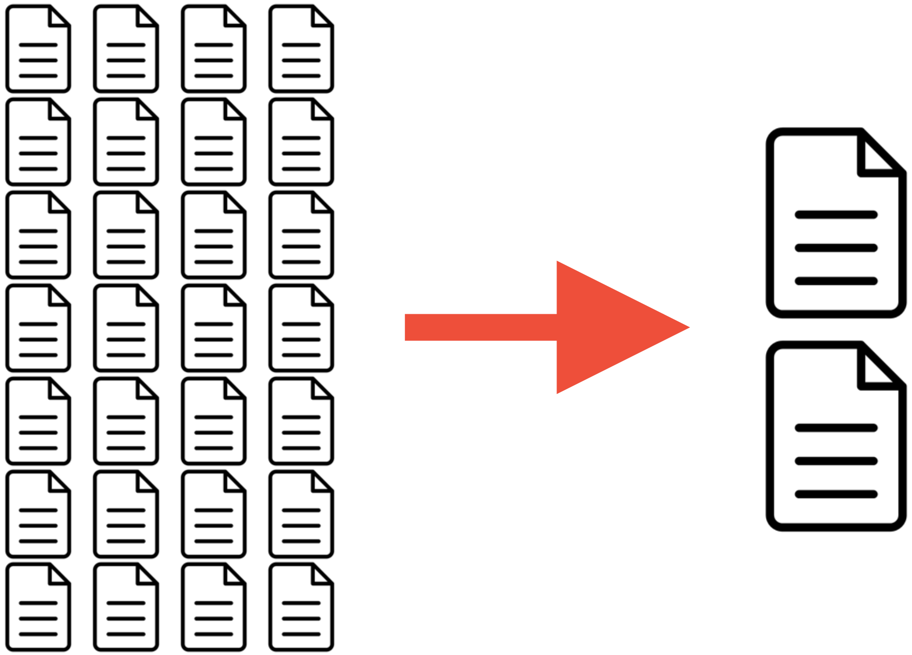
Available in Alluxio 2.1 distribution

In Progress: Merging connector into Presto codebase

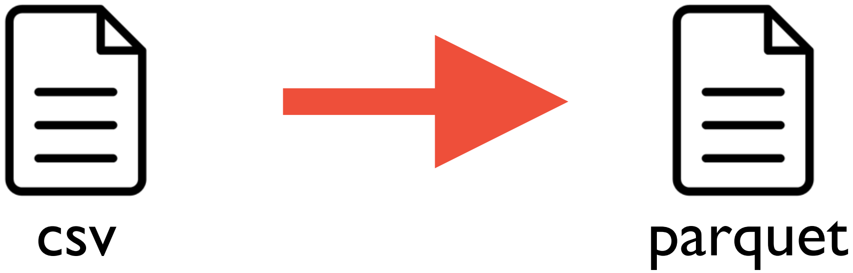
Transformation Service

Transform data to be compute-optimized independent from storage-optimized format

Coalesce



Format Conversion



Demo!

Demo

2 isolated AWS 10-node clusters

Presto + Hive Metastore + S3 Data

Presto + Alluxio + Hive Metastore + S3 Data

TPCDS sample dataset on S3

~10,000 CSV files

Demo Summary

Attached existing Hive database into Alluxio Catalog

Alluxio Catalog served table metadata for Presto

Transformed store_sales by coalescing and converting CSV to Parquet

Presto Without Alluxio	Alluxio Transformations	Alluxio Transformations With Caching
20s	7s	3s

Future Work

User community feedback/collaboration is important!

Future projects

- New UDB implementations (AWS Glue)
- More conversion formats (json)
- DDL/DML workloads (CREATE TABLE, INSERT, etc.)
- New Client APIs for structured data (Arrow)

Developer Preview Available in Alluxio 2.1

Try it out!

Documentation

Provide feedback

Feature requests and issues in Github Alluxio/alluxio

Thank You!