ALLUXIO

aws partner network

Accelerating analytics on AWS EMR & AWS S3 with Alluxio in a Disaggregated Data Stack

What's Inside

1 / Introduction

2 / Architecture Overview

3 / Storing Data in AWS EMR

4 / Data Orchestration with Alluxio on AWS EMR

5 / Getting Alluxio into AWS EMR

6 / Performance Testing Benchmark with Alluxio and EMR

© Copyright Alluxio, Inc. All rights reserved.
Alluxio is a trademark of Alluxio, Inc.

WHITEPAPER

1 / Introduction

Many organizations have taken advantage of the scalability and cost-savings of cloud computing services to scale to meet their data demands. AWS services like S3 and EC2 are now ubiquitous and extensively used by engineering and infrastructure teams of all sizes.

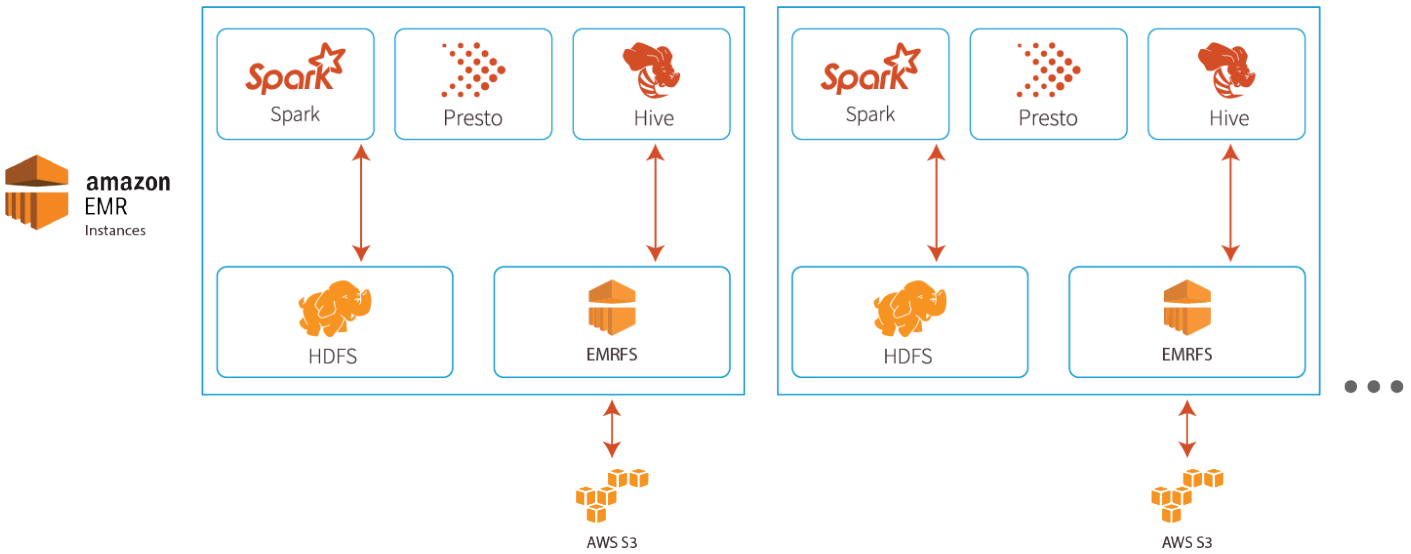
In addition to this, data analytics workloads are increasingly being migrated to the cloud. The AWS **EMR service** has made it extremely easy for enterprises to bring up a full-featured analytical stack in the cloud that elastically scales based on demand.

The EMR service along with S3 provides a robust yet flexible platform in the cloud with the click of a few buttons, compared to the highly complex and rigid deployment approach required for on-premise Hadoop Data platforms. However, because data on AWS is typically stored in S3, an object store, you lose some of the key benefits of compute frameworks like Apache Spark and Presto that were designed for distributed file systems like HDFS.

In this white paper, we'll share some of the challenges that arise because of the impedance mismatch between HDFS and S3, the expectations of analytics workloads of the object store, and how Alluxio with EMR addresses them.

2 / Architecture Overview

AWS EMR provides high flexibility in terms of the services that can be launched on each EMR node. The base model is similar to the on-premise Hadoop approach. Each EMR instance offers compute services like Spark, Presto and Hive along with a local distributed storage service (HDFS). However, EMR also provides added flexibility by using EMRFS (an abstraction that provides a Hadoop FileSystem API) on top of S3.



Most workloads use HDFS to store local intermediate data and persist final results using EMRFS.

3 / Storing Data in AWS EMR

Using HDFS to store data in AWS EMR

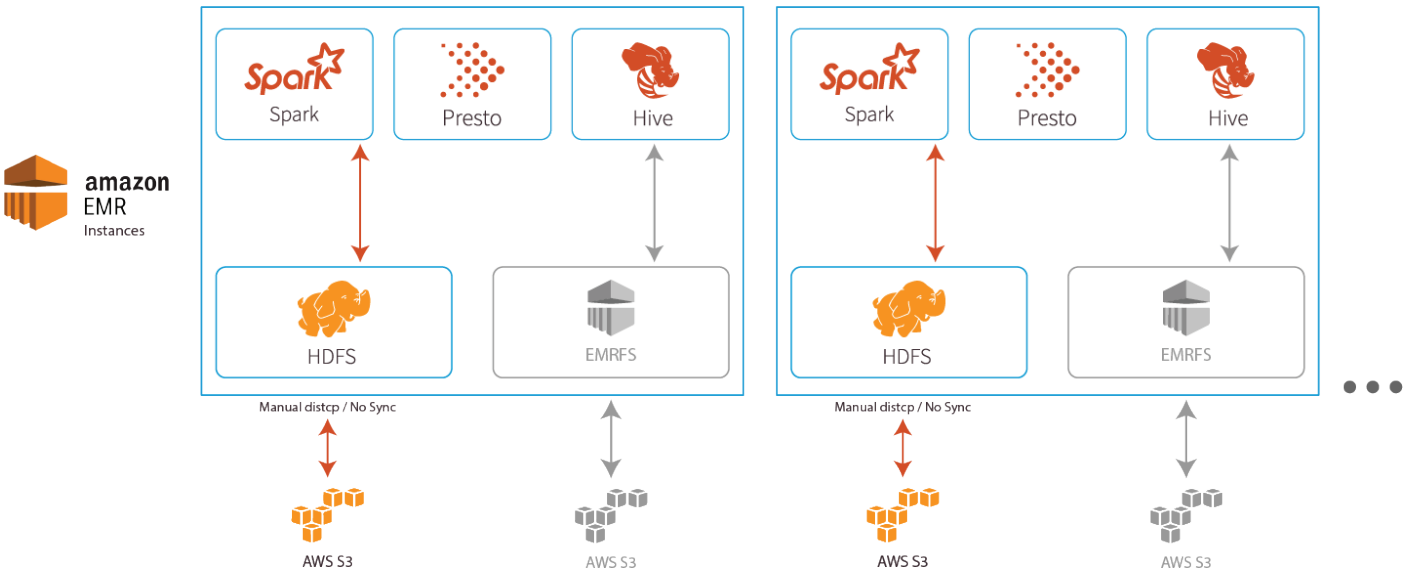
There are three considerations when using HDFS to store data in AWS EMR: ephemeral storage, manually syncing data to S3, and the performance of workloads.

EPHEMERAL STORAGE

When HDFS is used to store data using AWS EMR via any compute framework, the data is stored locally in each instance. While HDFS is useful for caching intermediate results of workloads like MapReduce and Spark, it cannot be persisted between runs as the volumes are reclaimed when the cluster is terminated.

MANUAL SYNC TO S3

If data needs to be persisted across different EMR cluster runs, then one option is a manual copy from HDFS to S3: typically Hadoop's distCP or the AWS extension S3DistCp (s3-dist-cp). In the scenario that this copy fails, the cluster step also fails and manual clean-up of files is needed before the job can be restarted.



PERFORMANCE OF WORKLOADS

The I/O for the workload has low latency for data locally cached in HDFS. But copying result sets to persistent storage like S3 after each key phase of the data pipeline increases the time taken for the analytics job to complete.

3 /

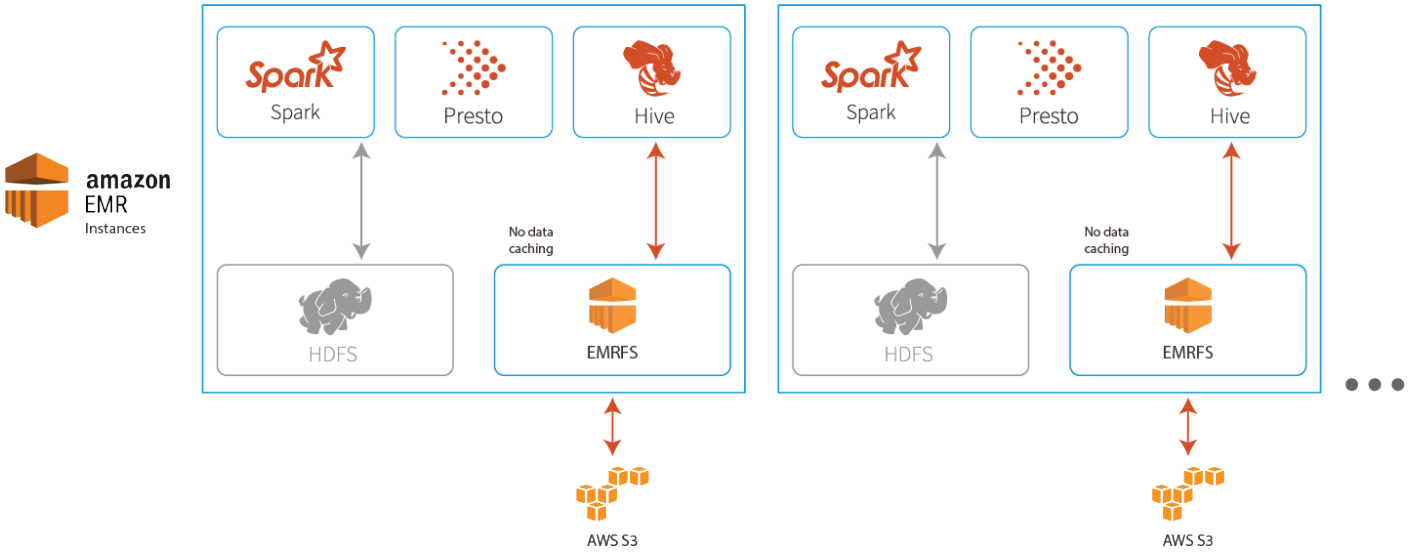
Using EMRFS to store data in AWS EMR

S3 object storage has become incredibly popular for a range of workloads. It offers scalability, resilience, cheap costs, and a simple API. However, these benefits come with limitations because cloud object stores have different guarantees when compared with distributed filesystems. Big data/analytics workloads rely on strong consistency and tree-based metadata structure to easily and reliably navigate datasets.

Performance of workloads with EMRFS on S3 is significantly lower compared with running directly on HDFS. There are three specific reasons this happens.

DATA IS NOT LOCALLY CACHED

Every piece of data that needs to be read has to be fetched via EMRFS from S3.



DATA IS EVENTUALLY CONSISTENT

Most Hadoop applications rely on a strongly consistent storage system. EMRFS solves this partially by caching some object metadata in AWS DynamoDB, but metadata still gets added to EMRFS only when an object is written by EMRFS during the course of an Amazon EMR job or an object is synced with or imported to EMRFS metadata by using the EMRFS CLI. Thus, there are situations where metadata can fall out of sync with S3 and this can be resolved using emrfs sync.

In general, a few key requirements emerge from these challenges.

DATA LOCALITY AND THE NEED FOR LOCAL DATA CACHING

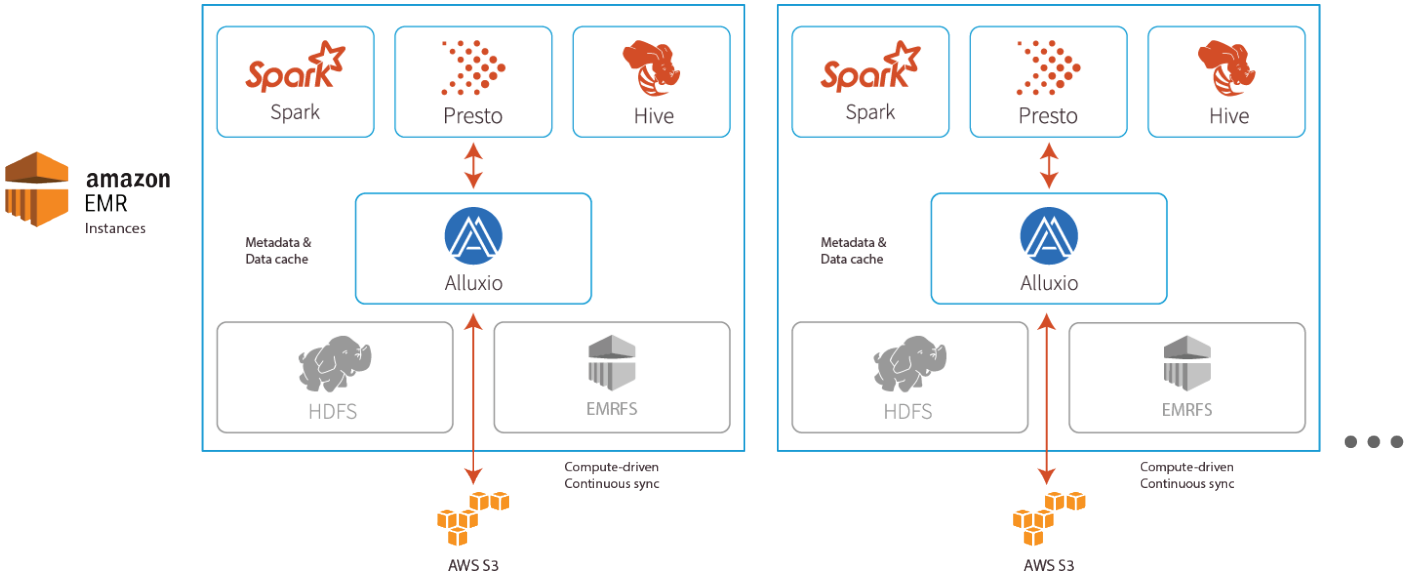
Performance of workloads, particularly interactive workloads, depends heavily on data locality all the way down to the node level.

API COMPATIBILITY FOR BOTH HDFS AND S3

Since the compute frameworks still widely expect HDFS on the programming interface, compatibility becomes important in addition, data will be increasingly persisted in S3 when durability is required, driving the need for a native S3 interface.

4 / Data Orchestration with Alluxio on AWS EMR

Alluxio is a [data orchestration layer] that enables users to increase performance and scope of analytic workloads running on AWS EMR using S3 as the storage.

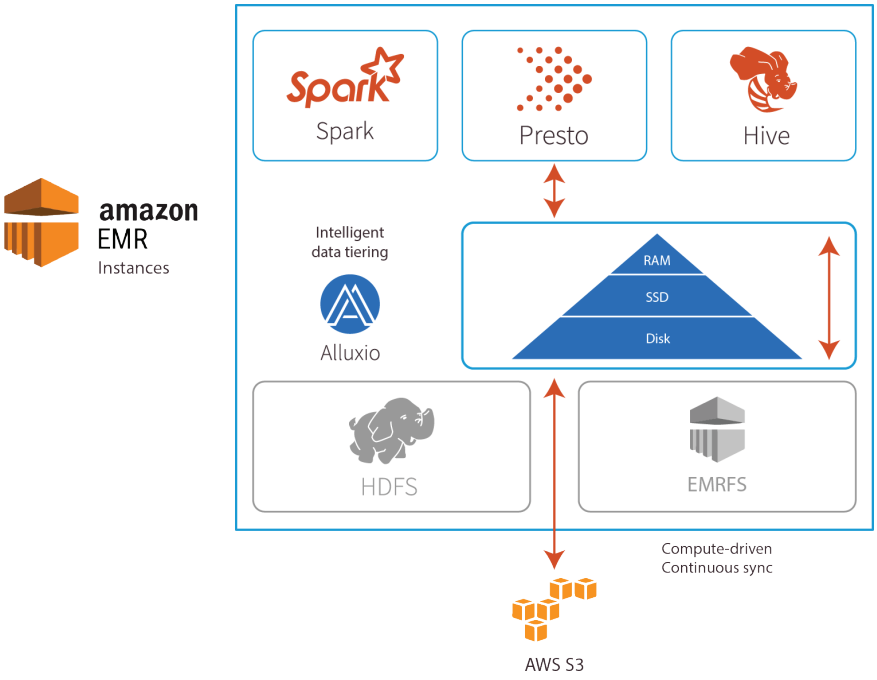


Alluxio solves the challenges above by providing a tiered caching layer using memory, SSDs and/or disks for better performance. It provides HDFS and S3 API compatibility for both compute frameworks on top and storage systems below Alluxio and it provides a unified namespace across multiple storage systems allowing for disaggregation of compute and storage within AWS.

Data Locality with Intelligent tiering

Alluxio includes a cache buffer which includes multi-tiered storage. When tiered storage is enabled, the eviction process intelligently accounts for the ordering of tiers. Alluxio assumes that tiers are ordered from top to bottom based on I/O performance. For example, users often specify the following tiers:

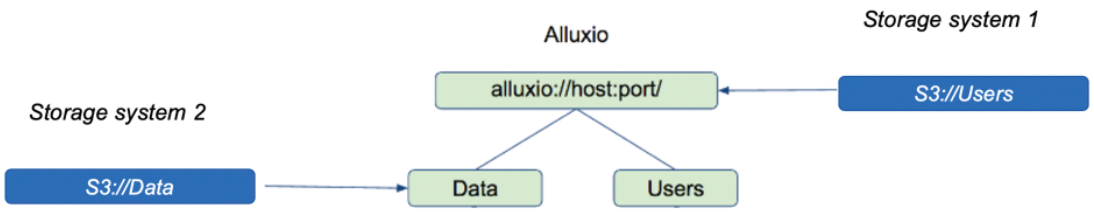
- MEM (Memory)
- SSD (Solid State Drives)
- HDD (Hard Disk Drives)



Global namespace

One of the key benefits that Alluxio provides is a unified namespace to the applications. This is an abstraction that allows applications to access multiple independent storage systems through the same namespace and interface. Rather than communicating to each individual storage system, applications can delegate this responsibility by connecting through Alluxio, which will handle the different underlying storage systems.

Using this multiple S3 buckets from different accounts can be mounted into Alluxio and can be accessed via Spark, Presto or Hive without knowing they are from different accounts or even different regions.



5 / Getting Alluxio into AWS EMR

Bootstrapping with Alluxio

Alluxio can be added to each AWS EMR node using the EMR bootstrap approach.

Using the Alluxio bootstrap script, an EMR cluster can be configured to run the Alluxio service as well. This provides an out of the box Alluxio cluster mounted to a chosen Amazon S3 bucket.

EXAMPLE

You can use the bootstrap script located here to add Alluxio to every node of the EMR cluster.
You can run the following command on the AWS CLI with a range of different options including a pointer to a configuration file and root under storage. You can also specify the default write option for Alluxio.

```
aws emr create-cluster \
  --release-label ${RELEASE_LABEL} \
  --instance-count ${NUM_INSTANCES} \
  --instance-type ${INSTANCE_TYPE} \
  --applications Name=Presto Name=Hive Name=Spark \
  --name "${CLUSTER_NAME}" \
  --bootstrap-actions Path=${BOOTSTRAP_PATH},Args=[${ALLUXIO_DOWNLOAD_URL},${ROOT_UFS_URI},  ${ADDI-
TIONAL_PROPERTIES}] \
  --configurations file:///Users/diptiborkar/Downloads/alluxio-emr.json \
  --ec2-attributes KeyName=${KEY_PAIR}
```

5 /

ARGUMENTS

```
RELEASE_LABEL="emr-5.23.0"
NUM_INSTANCES=5
INSTANCE_TYPE="m4.xlarge"
CLUSTER_NAME="emr-v12"
BOOTSTRAP_PATH="s3://dipti-alx-2019/emr/alluxio-emr.sh"
ALLUXIO_DOWNLOAD_URL="https://downloads.alluxio.io/downloads/files/2.0.0/alluxio-2.0.0-RC3-bin.tar.gz"
ROOT_UFS_URI="s3a://dipti-alx-2019/emr/ufs/"
ADDITIONAL_PROPERTIES="alluxio.underfs.s3.owner.id.to.user-
name.mapping=${S3_ID}=hadoop;alluxio.user.file.writetype.default=ASYNC_THROUGH"
```

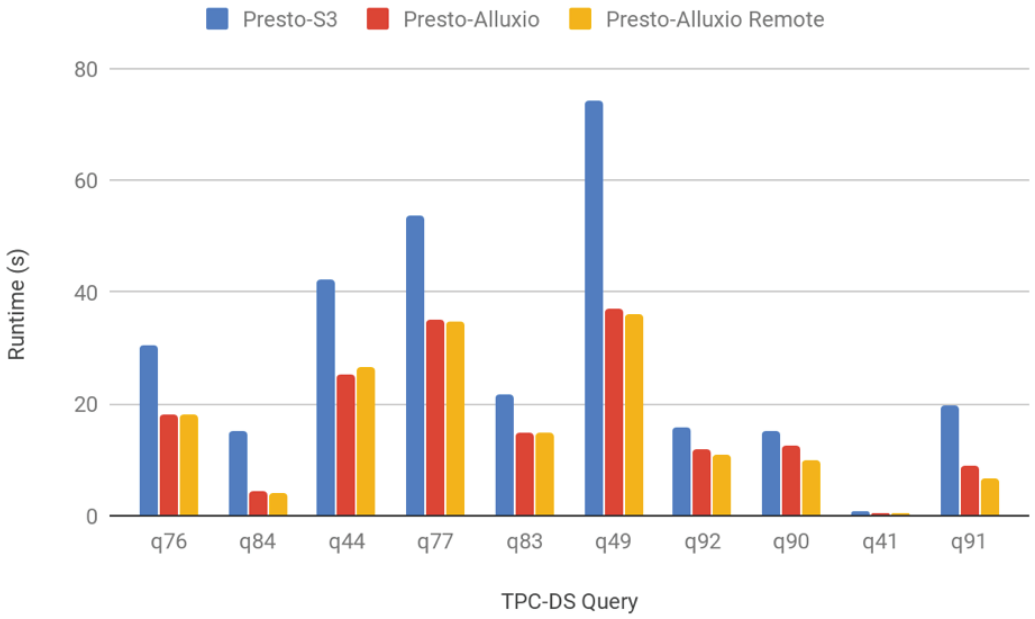
Tutorial: Alluxio on AWS EMR

Alluxio also integrates with **AWS Glue** which allows users to persist table definitions and catalog metadata between different EMR runs. Users can then spin up on-demand EMR clusters pre-configured with Alluxio, mount the S3 locations they wish to use, and accelerate existing workloads to run faster.

6 / Performance testing benchmark with Alluxio and EMR

Performance testing of Alluxio on EMR (version 5.23.0) has yielded some promising initial results. The below chart shows that Alluxio's caching of warm data results in performance gains for data stored in S3. The tests were run on five r4.2xlarge machines (1 master and 4 workers) in the us-east-1 AWS region. The data and query set used was the TPC-DS suite with data at a scale factor of 100.

Below is a chart detailing 10 queries with the highest performance increase from Alluxio. The chart also shows that Alluxio allows remote storage to be accessed as if it were local by intelligently caching the data that is queried. This was done by using an EMR cluster in a different region (ap-southeast-1) but with an S3 bucket that was from us-east-1. This demonstrates that AWS EMR along with Alluxio can be used to analyze and process data they may even be remote. This is particularly the case for workloads that have the "write once - read many times" pattern.



The cost savings after using Alluxio is also significant. By caching the data that users access, Alluxio mitigates charges incurred by AWS inbound/outbound data transfer fees across regions. There are additional charges that Alluxio saves, such as metadata listing and S3 endpoint calls, since Alluxio will store the file-system tree structure of the underlying S3.

By localizing the data and metadata to the EMR cluster, Alluxio boosts performance while also saving significant costs. Alluxio can localize your objects to the existing region for the most important data or hot data based on the specific objects the user requires.

6 /

To conclude, initial findings show that Alluxio pairs strongly with Amazon EMR to orchestrate data storage in the cloud. The combination of on-demand compute, managed Hadoop services, and a data orchestration service enables users to leverage and enhance the services that cloud providers offer. It also enables new use-cases for AWS users such as migrating data from on premise to AWS for Hadoop HDFS offload as well as hybrid cloud deployments with AWS.

7 / Additional Resources

- Alluxio - Presto sandbox with AWS AMI
- [Webinar] - Apache Spark on AWS S3 wIth EMR
- [Tutorial] - EMR Bootstrap

Proven at global web scale in production for modern data services, Alluxio is the developer of open source data orchestration software for the cloud. Alluxio moves data closer to big data and machine learning compute frameworks in any cloud across clusters, regions, clouds and countries, providing memory-speed data access to files and objects. Intelligent data tiering and data management deliver consistent high performance to customers in financial services, high tech, retail and telecommunications. Alluxio is in production use today at seven out of the top ten internet companies. Venture-backed by Andreessen Horowitz and Seven Seas Partners, Alluxio was founded at UC Berkeley's AMPLab by the creators of the Tachyon open source project.