Architecture and Considerations in Machine Learning Pipeline

EFFICIENT DATA ACCESS STRATEGIES FOR LARGE-SCALE AI

Introduction

The adoption of artificial intelligence (AI) is rapidly growing, with 49% of CIOs indicating that they either already use or plan to use AI [1]. The recent boom of generative AI further accelerates this adoption, making AI a business imperative for revenue generation, customer satisfaction, and organizational productivity.

Successful AI projects require access to data. The ability to quickly serve data for applications is essential. As AI use cases grow more complex, we need to understand data access patterns and how to address them with the right solution.

This white paper comprehensively explains data access patterns in a modern AI/ML platform. It discusses the characteristics of data access in each stage of the machine learning pipeline and strategies to optimize your data access for large-scale AI.

Contents

1. Data Access – The Barrier to AI Implementation

Data access is a critical challenge for AI implementations. Gartner's research found that data accessibility is the **top barrier to AI implementations** [2].

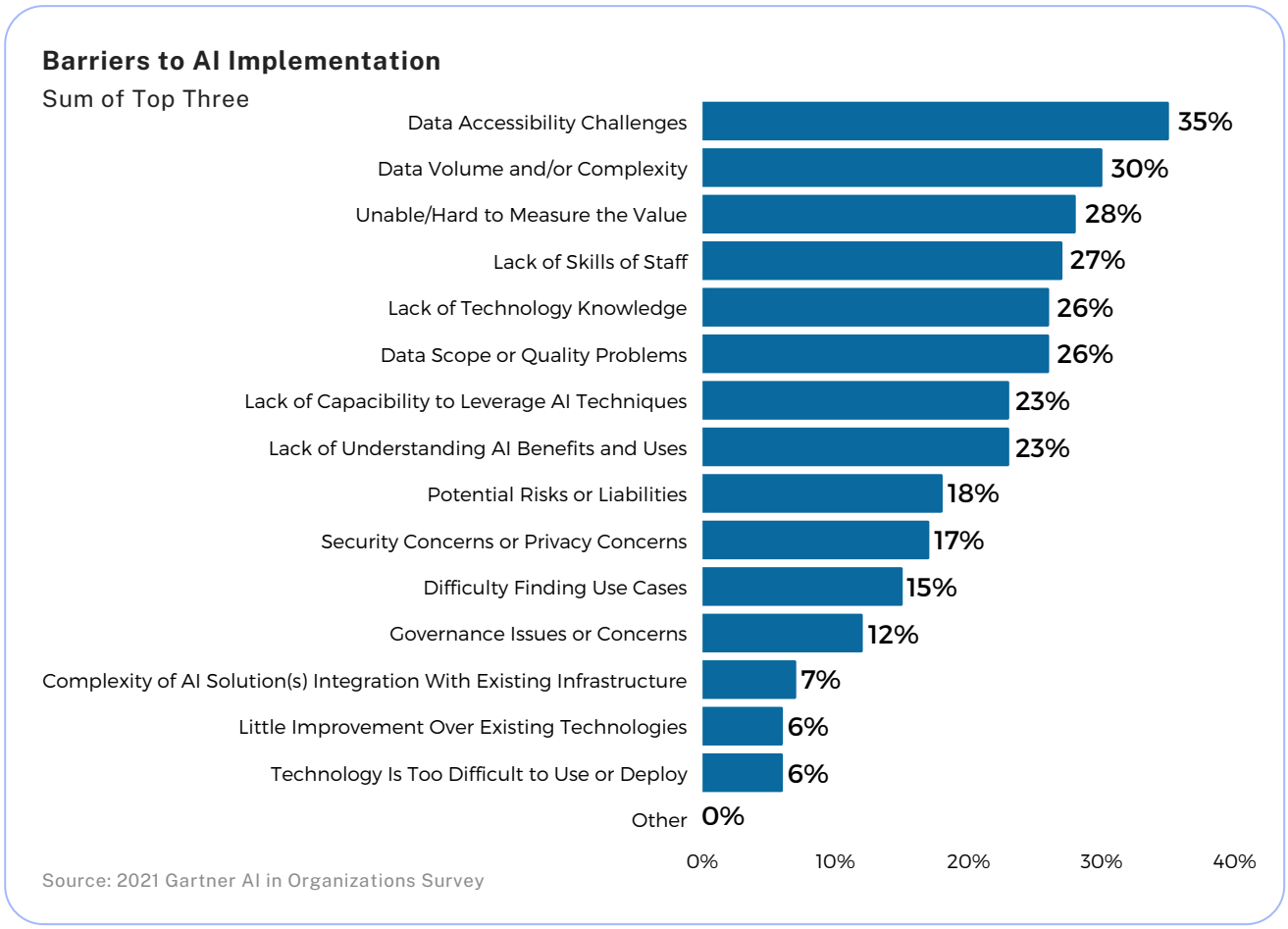


Figure 1: *Gartner - Barriers to AI Implementation [2]*

Access to data hinders the success of AI projects for several reasons.

High-Quality AI Models Requires Access to Massive Datasets

AI workloads are far more data-intensive than traditional enterprise applications. The quality and accuracy of AI models depend heavily on having access to large volumes of training data. The data requirements for effective AI are not just about absolute size, but also variety and complexity. The ability to access data can significantly impact overall outcomes of AI projects.

Data Access is Slow and Costly, Both in Hybrid/Multi-cloud and Single-cloud Environments

In enterprise contexts, relevant datasets are often spread across different cloud environments, data centers, and geographic regions. AI applications need the ability to access data regardless of where it resides. Accessing data across distributed clouds can introduce significant latency and high cloud storage API and egress costs. Even in the case of single-cloud/region, as most persistent storage is designed for massive data stores with low prices, access to data will also encounter low performance challenges.

Increasing Size of Models Slows Down Application Performance

As AI techniques advance, models are becoming larger and more complex. The size of state-of-the-art AI models has been doubling every 3.4 months, according to research from OpenAI [3]. Also, for maximum accuracy, models need to be updated and served frequently. For downstream applications, highly concurrent access to large model files is challenging.

Limited Availability of GPU Instances Necessitates Remote Data Transfer

GPUs are scarce resources these days. For example, Amazon EC2 P4 instances with A100 GPUs may only be available in certain AWS regions [4], while your training data resides remotely. When transferring data to GPU instances for model training, this results in slow model training and high egress costs.

GPUs Waiting for Data Retrieval Results in Underutilized GPUs

GPUs are a critical accelerator technology for AI workloads. However, GPU time is expensive. It is essential to maximize GPU utilization and reduce any wait time stemming from data access. The challenge is keeping GPUs continuously fed with data to avoid computation sitting idle. The speed of data access is now the bottleneck.

To overcome data accessibility challenges, you need to thoroughly understand data access characteristics at every step of your ML pipeline.

2. Demystify Data Access Patterns in Machine Learning Pipeline

2.1 Stages in the Machine Learning Pipeline

AI workloads are far more data-intensive than traditional enterprise applications. The quality and accuracy of AI models depend heavily on having access to large volumes of training data. The data requirements for effective AI are not just about absolute size, but also variety and complexity. The ability to access data can significantly impact overall outcomes of AI projects.

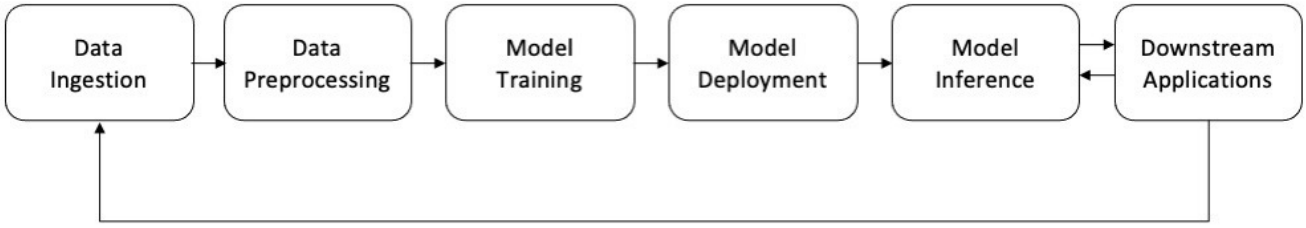


Figure 2: *Stages in the Machine Learning Pipeline*

The machine learning pipeline includes the following stages of the model development life cycle.

- **Data ingestion** is the step of bringing data from various sources into a primary data pipeline. This can be done using data integration tools, which can extract, transform, and load data from a variety of sources.
- **Data preprocessing** is the process of preparing data for model training. This includes cleaning the data, removing outliers, and transforming the data into a format that can be used by the model. Feature engineering is also a part of data preprocessing, and it involves creating new features from the existing data.
- **Model training** is building a model that can make predictions based on the data. This is done by using machine learning algorithms to identify patterns in the data. The processed training and retraining data is forwarded for use in the execution of ML routines (such as A/B testing, model tuning, and hyperparameter tuning).
- **Model deployment** is to make a model available for production use. This involves packaging the model and making it accessible to applications that need to use it.
- **Model inference** is the process of using a model to make predictions. This involves feeding new data into the model and obtaining a prediction from the model. The inferences from the model — model score, output data stream, and insights — now affect the outcome of downstream applications.

The ML pipeline is an iterative process, and there is a feedback loop. Once the model is deployed, you need to measure its effectiveness and performance by capturing its performance and new training data so that the model can be improved and updated to generate better results.

2.2 Data Access Patterns

2.2.1 What is Data Access Pattern?

Data access pattern refers to the characteristics of how data is accessed from storage systems. It provides important information that can be leveraged to optimize the data processing workflows of your AI platform. The main aspects of data access patterns include:

- **Type of access:** The operations done after opening the file, such as read and write operations, and the characteristics of access, like read-only, write-only, etc.
- **Access mode:** It can be random read/write or sequential read/write. In random access, data blocks are accessed in arbitrary order based on application logic. In sequential access, data blocks are read/written linearly from start to end.
- **File size:** The amount of data read or written in a single access. This is categorized as:
  - Small: <100KB
  - Medium: 100KB~100MB
  - Large: 100MB~100GB
- **Number of files:** The total files in the dataset being accessed. This is classified as:
  - Small: < 1K
  - Medium: 1K~ 1 million
  - Many: 1 million ~ 100 million
  - Massive: 100 million ~ 10 billion or more
- **File format:** The format of data, including structured, like Parquet, ORC, and unstructured, like jpeg images.

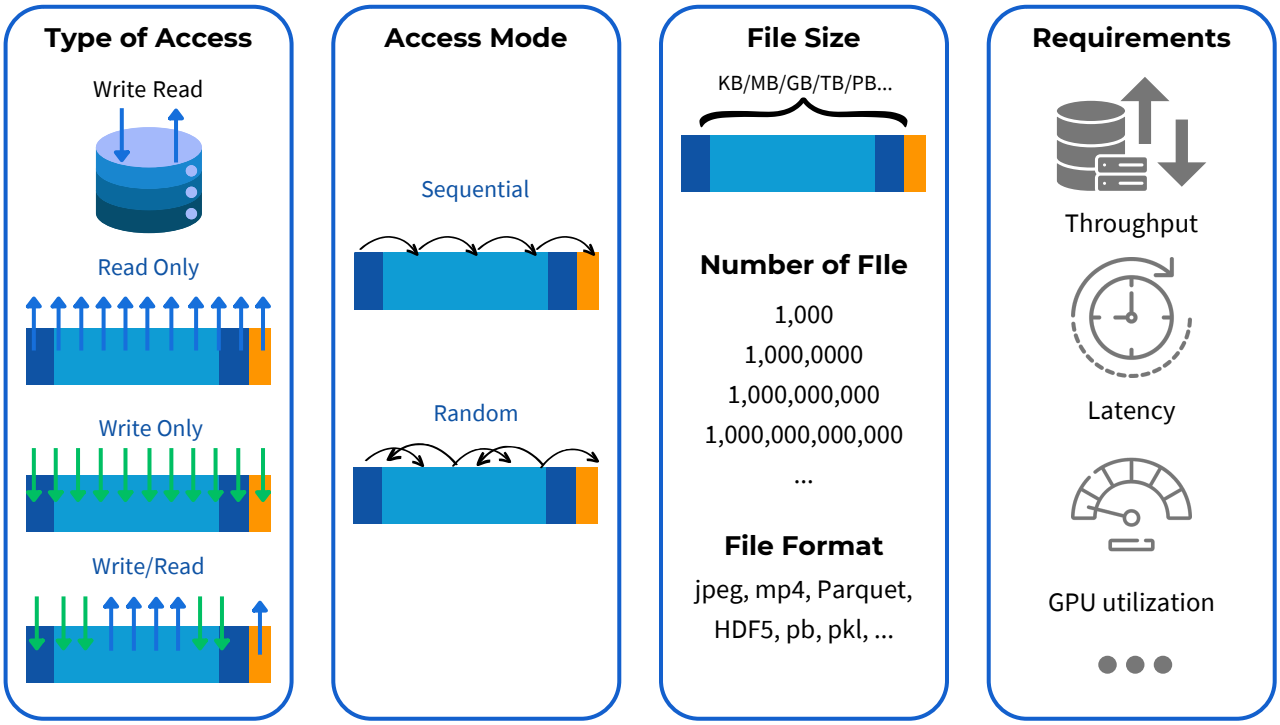


Figure 2: *What is Data Access Pattern?*

2.2.2 Data Access Pattern in the Machine Learning Pipeline

Each stage of the ML pipeline has distinct data access patterns and corresponding requirements. Data ingestion and model training require high throughput while preprocessing requires mixed read/write handling. Inference, on the other hand, requires low latency and high throughput.

Table 1 shows different stages of the machine learning workflow and the corresponding data access patterns.

- **Data ingestion** workloads typically have sequential access patterns and use files of any kind and size. They are not very sensitive to latency, except in the case of streaming data. Writes account for 90% of the workload's input/output activity.
- **Data preprocessing** workloads use both random and sequential access. They have balanced read-and-write patterns, use multiple data types and sources, and manage files of all sizes, from small to large. Real-time data processing requires high latency, and batch-style data processing requires high throughput.
- **Model training, deployment, and inference** workloads typically have sequential access patterns and process a single type of data in small files. They require low latency and high throughput and can benefit significantly from GPU acceleration. Data analytics algorithms, on the other hand, run faster on traditional CPUs.

Different access patterns require infrastructure to be optimized differently - ingestion requires high write throughput, training needs high read throughput and high GPU utilization, deployment requires low latency and high concurrency, and inference requires low latency and high availability. See table 1 for details.

	Data Ingestion	Data Preprocessing		Model Training			Model Deployment	Model Inference
		Unstructured or Semi Structured	Structured	CV	NLP	Checkpoint Write		
Type of Access	Mostly write	Both read and write	Both read and write	Mostly Read	Both read and write	Write Only	Mostly Read	Read Only
Access Mode - Read	N/A	Sequential Read	Random Read(4k)	Sequential Read	Random Read(4k)	N/A	Sequential Read	Sequential Read
Access Mode - Write	Sequential Write or Append	Sequential Write or Append	Sequential Write or Append	N/A	N/A	Sequential Write or Append	Sequential Write	N/A
File Size	Small to Large	Small to Large	Medium to Large	Small	Large	Large	Small to Large	Small to Large
Number of Files	Small to Medium	Many	Small	Massive	Small	Small	Small	Small
File Format	Parquet, ORC, Avro, Arrow	jpeg, gif, json or text, mp4	Parquet or ORC	Unstructured data, like jpeg	Structured or semi-structured data	NPZ, HDF5, tf-native	pb, pkl, h5, onnx, mlmodel	pb, pkl, h5, onnx, mlmodel
Requirements for Data & AI Platform	• High throughput • Combine all data sources	• High throughput (batch processing) • Low latency (real-time processing) • High CPU utilization		• High throughput • High read performance • High GPU utilization		• High throughput • High write performance	• Low latency • High concurrency	• Low latency • High throughput • High availability

Table 1: *Data Access Patterns in Each Stage of The ML Pipeline*

2.2.3 Single Cloud Data Access Pattern

When training in a single cloud or a single data center, unstructured and structured training datasets produce different data access patterns, which would impact the performance of data access.

Training on Unstructured Dataset

When accessing unstructured data (such as JPEG or GIF), the data access pattern is mostly sequential to read the entire file. This type of read pattern results in streaming rather than random reads for cold and warm reads (where a warm read hits the local cache on local NVMe storage) when reading a production ML dataset with more than 10k files.

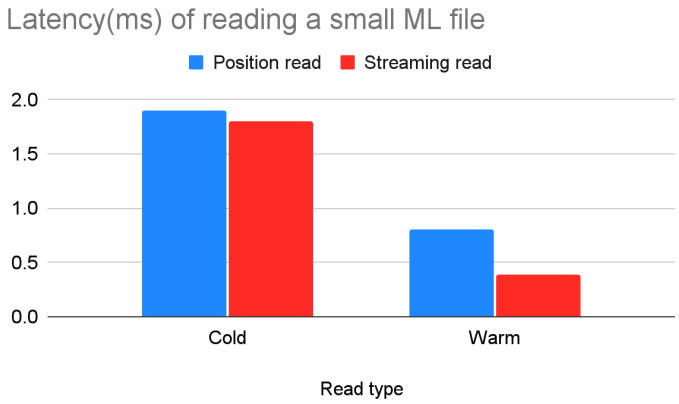


Figure 4: *Single Cloud Data Access Pattern of Training on Unstructured Dataset*

Training on Structured Dataset

When accessing structured data (such as Parquet or ORC), we found the data access pattern to be primarily small, random reads. This type of read pattern resulted in position reads outperforming streaming reads when we ran read operations with 4 threads on a production ML dataset, for both warm and cold reads when reading large ML structured datasets.

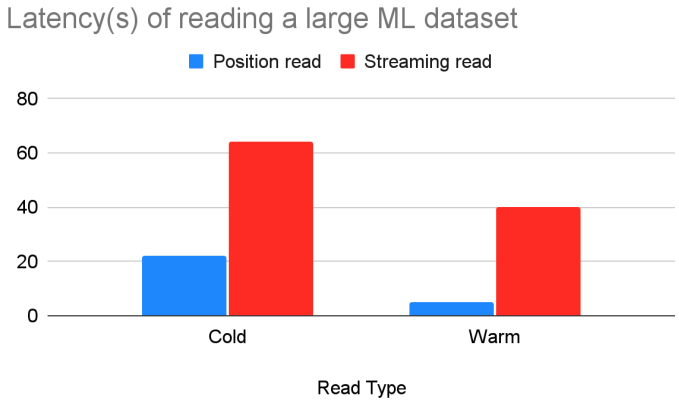


Figure 5: *Single Cloud Data Access Pattern of Training on Structured Dataset*

2.2.4 Multi-cloud/Multi-region Data Access Pattern

In some cases, the machine learning pipeline stages may be distributed across regions or clouds. For example, data ingestion may be preprocessed in one region, models retrained in another, and inferencing performed in one or more other regions.

The motivation behind choosing a multi-region, multi-cloud approach is cost, performance, and service capabilities. An organization may want to leverage the most cost-efficient cloud resources available. Also, inference stages often need to be placed geographically close to end users to reduce latency. Finally, some cloud providers offer specialized resources or services that others do not. For example, Google Cloud offers TPUs, and AWS provides SageMaker.

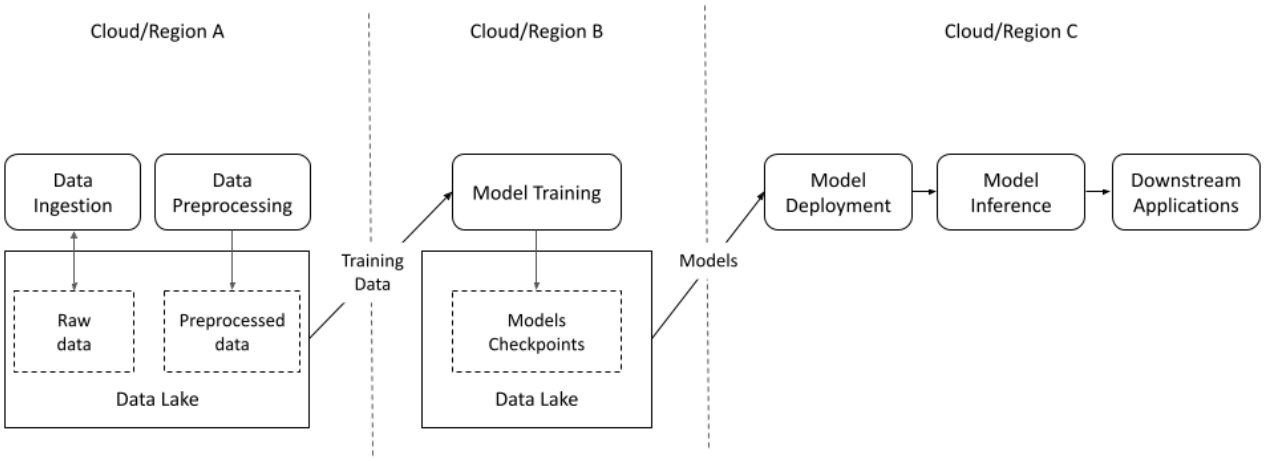


Figure 6: *Multi-cloud/Multi-region Data Access [5]*

The above figure illustrates the scenario of data access architecture of multi-cloud/region [5].

From cloud A to cloud B, the data access patterns of transfer training data are periodic, read-only access to training data. The model training speed is usually bottlenecked by fetching remote training data.

From cloud B to cloud C, the data access patterns are periodic, read-only access to models. Note that one trained model is usually shared by several model inference processes, informed by downstream applications. When the model size is large, like large language models, data access throughput is important.

2.3 Factors to Consider

Given the understanding of data access patterns and requirements, when building an architecture to optimize data access, you need to consider the performance, scalability, and reliability of the solution. This will ensure that you are able to deliver the full value of your AI infrastructure investments.

The solution to data access should support the following:

- High performance and throughput for ML workloads
- Dataset management, including load/unload/update of data from the data lake
- Cloud-native capabilities, such as multi-tenancy, scalability, and elasticity
- Eliminate data redundancy to avoid managing multiple copies of data
- Reduced dependency on specialized networking hardware
- Flexibility to place compute anywhere, regardless of the location of the data
- Agnostic to cloud service providers to avoid vendor lock-in
- Future-proofing to adapt to advancements in storage and computation technologies
- Security, including consistent authentication and authorization

Alluxio provides a solution that meets all of these requirements. It can connect machine learning engines with different storage systems and virtualize the data across regions and clouds. This allows your organization to access and manage data from different sources in a unified way. Alluxio is an architecture optimized for on-demand access to data, so you can get out of its way and access it to the right place at the right time.

Alluxio offers the following values:

- Automatically load / unload / update data from your existing data lake.
- Faster access to training data informed by data access patterns.
- Maintain optimal data access with high data throughput to keep the GPU fully utilized.
- Deploy models faster and provides high concurrency model serving to inference nodes.
- Increase the productivity of the data engineering team by eliminating the need to manage data copies.
- Reduce cloud storage API and egress costs, such as the cost of S3 GET requests, data transfer costs, etc.

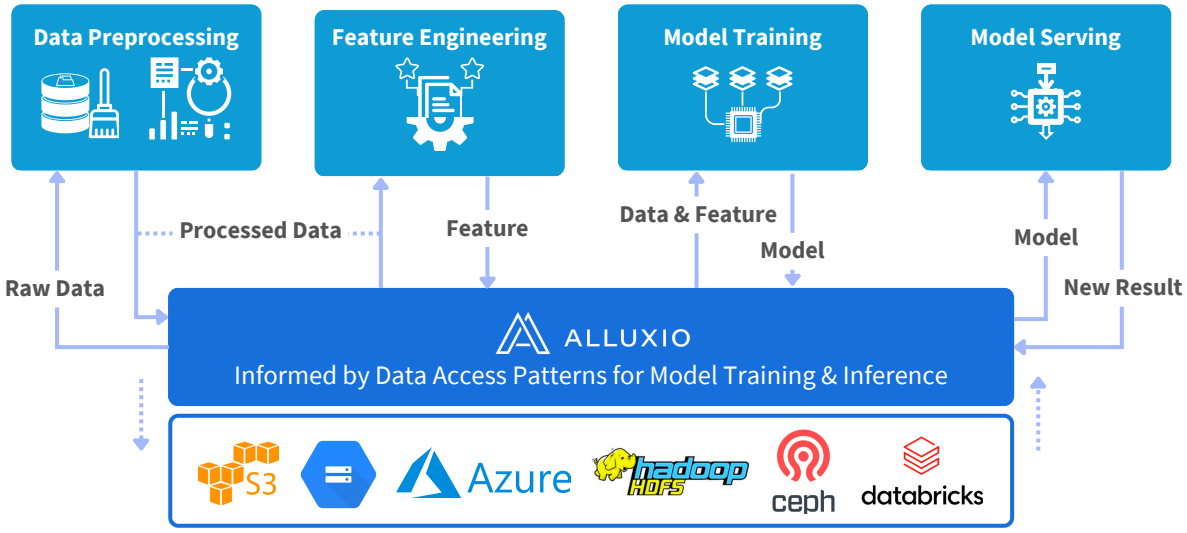


Figure 7: *Alluxio-powered Data Access Across the ML Pipeline*

3. Optimizing Data Access in AI/ML Platform – Architecture and Benchmark

3.1 Architecture Overview

In this section, we will focus primarily on the model training and model serving (deployment) phases of the ML pipeline because they are the most resource intensive stages. The following is a reference architecture for model training and serving with Alluxio.

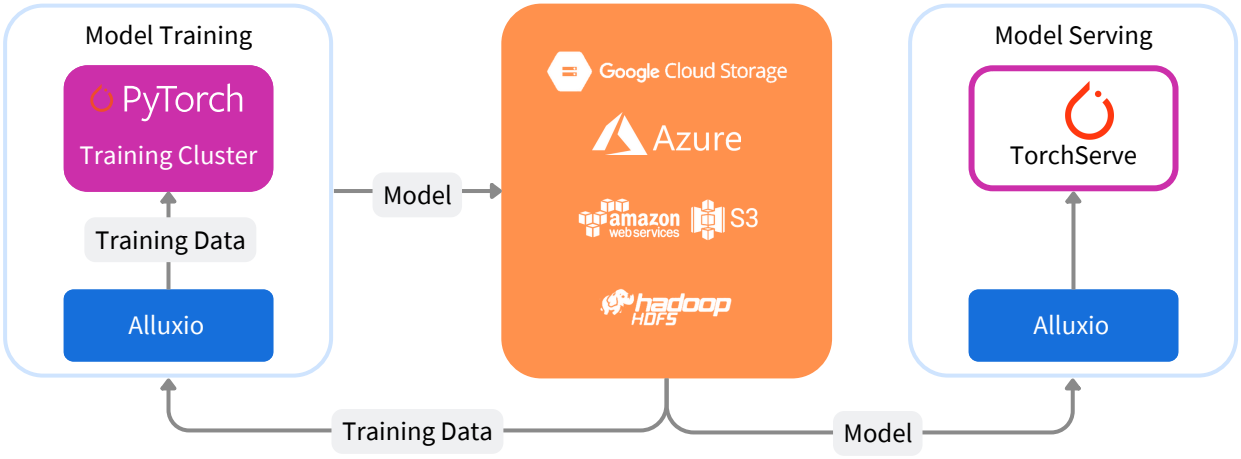


Figure 8: *Architecture for Model Training and Serving with Alluxio*

In this reference architecture, the training data is stored within a centralized data storage platform, such as AWS S3 or GCS (Google Cloud Storage). To facilitate seamless provisioning of the training data to the model training cluster, Alluxio is employed. ML training frameworks, including PyTorch, TensorFlow, scikit-learn, and XGBoost, are executed on CPU/GPU/TPU clusters. These frameworks utilize the training data to generate ML models, which are subsequently stored in a centralized model storage repository.

For the model serving stage, dedicated serving/inference clusters are utilized, employing frameworks such as TorchServe, TensorFlow Serving, Triton, and KFServing. These serving clusters leverage Alluxio to retrieve the models from the model storage repository. Once loaded, the serving clusters handle incoming queries, execute the necessary inference jobs, and return the computed results.

Both the training and serving environments are based on Kubernetes, which facilitates easier scalability and reproducibility of the infrastructure.

3.2 Performance and GPU Utilization Benchmarks for Alluxio-powered Model Training

3.2.1 Deep Learning Algorithm and Dataset

ResNet (Residual Neural Network) is a widely used deep learning model that has gained popularity in the field of computer vision. It builds upon the fundamental architecture of convolutional neural networks (CNNs) but effectively addresses the issue of vanishing gradients, enabling better training performance and improved accuracy. In the context of the reference architecture, we plan to employ ResNet in conjunction with the ImageNet dataset for image classification tasks, serving as an example use case in the computer vision domain.

3.2.2 Deployment and Test Setup

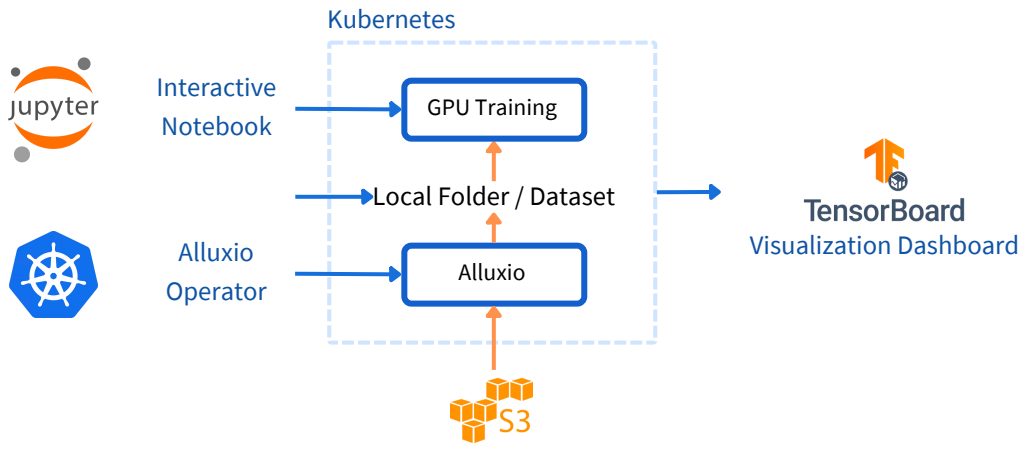


Figure 9: *Benchmark Deployment and Test Setup*

Test Setup Summary

- Alluxio - Kubernetes
- GPU server - AWS EC2/Kubernetes
- Deep learning algorithm (CV) - ResNet (one of the most popular CV algorithms)
- Deep learning framework - PyTorch
- Dataset - ImageNet (subset - ~35k images, each is ~100kB - 200kB)
- Dataset storage - S3 (single region)
- Mounting - FUSE
- Visualization - TensorBoard
- Code execution - Jupyter notebook

Baseline

- S3-FUSE

3.2.3 Benchmark Results

Training Performance Benchmark Results

Based on the results of Resnet-50, 3 epochs performance benchmark, Alluxio is **5 times faster** than S3-FUSE. In general, increased data access performance reduces the overall time for model training.

	Alluxio	S3 - FUSE
Total Training Time (3 epochs)	**17 minutes**	**85 minutes**

Table 2: *Computer Vision Training Performance Benchmark Results: Alluxio vs. S3-FUSE*

GPU Utilization Benchmark Results

With Alluxio, GPU utilization improved significantly. Alluxio has reduced data loading time from 82% to 1%, resulting in GPU utilization increasing **from 17% to 93%**.

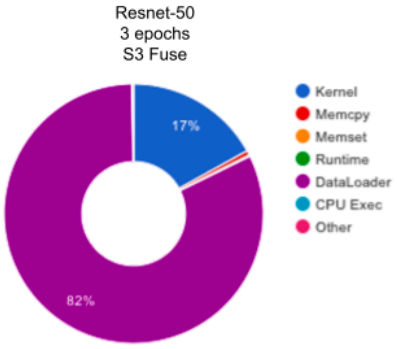
Before using Alluxio

> 80% of Total Time is Spent on DataLoader

Result in low GPU Utilization Rate (<20%)

GPU Summary	
Name	Tesla T4
Memory	14.62GB
Compute Capability	7.5
GPU Utilization	**16.96%**
Est. SM Efficiency	16.91%
Est. Achieved Occupancy	68.75%
Kernel Time using Tensor Cores	0.0%

Category	Time Duration (us)	Percentage (%)
Average Step Time	1,763,649,145	100
Kernel	299,168,905	16.96
Memcpy	10,521,722	0.6
Memset	39,459	0
Runtime	3,043,169	0.17
DataLoader	**1,446,068,956**	**81.99**
CPU Exec	1,570,076	0.09
Other	3,245,858	0.18



After using Alluxio

Reduce Data Loader Rate from 82% to 1%

Increase GPU Utilization Rate from 17% to 93%

GPU Summary	
Name	Tesla T4
Memory	14.62GB
Compute Capability	7,5
GPU Utilization	**93.29%**
Est. SM Efficiency	92.98%
Est. Achieved Occupancy	68.03%
Kernel Time using Tensor Cores	0.0%

Category	Time Duration (us)	Percentage (%)
Average Step Time	334,274,946	100
Kernel	311,847,023	93.29
Memcpy	10,500,126	3.14
Memset	43,946	0.01
Runtime	3,899,241	1.17
DataLoader	**3,343,301**	**1**
CPU Exec	1,648,391	0.49
Other	2,992,918	0.9

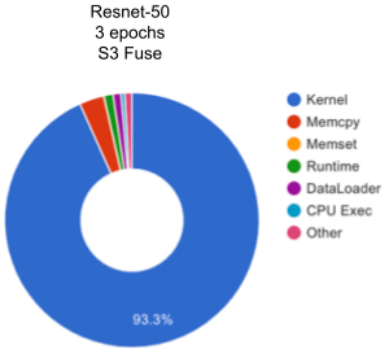


Figure 10: *Computer Vision Training GPU Utilization Benchmark Results: Alluxio vs. S3-FUSE*

4. Real-world Scenarios

4.1 FinTech Giant, AliPay, Speeds up Large-Scale CV Training on Billions of Small Files

AliPay is the world's largest mobile payment platform, serving over 1.3 billion users and 80 million merchants. In order to provide its users with the best possible experience, Alipay relies on machine learning models to power a variety of features, such as fraud detection, risk assessment, and personalized recommendations.

However, as AliPay's user base and transaction volume grew, the company began to experience challenges with model training. The disparity between computation and storage performance was causing model training to be slow and inefficient. Additionally, the high cost of specialized hardware was putting a strain on Alipay's budget.

To address these challenges, AliPay began using Alluxio, a unified data access layer that can accelerate machine learning workloads. Alluxio provides a high-performance cache that sits between the compute and storage layers, reducing latency and improving throughput. This allows AliPay to train models on commodity hardware, which is much more cost-effective than specialized hardware.

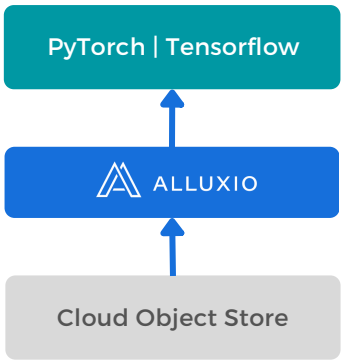


Figure 11: *AliPay Model Training Architecture with Alluxio*

In addition to improving performance, Alluxio also simplifies data management for AliPay. Alluxio eliminates the need to maintain data copies by providing on-demand data access. This frees data engineers to focus on other tasks, such as optimizing model performance.

As a result of using Alluxio, AliPay has seen significant improvements in the speed and efficiency of its model training. The company has also reduced its infrastructure costs and freed up data engineers to focus on more strategic tasks. Learn more here >>.

> After attempting various methods to address our challenges, only Alluxio is able to meet our requirements for large-scale AI training. Alluxio has significantly enhanced our AI training jobs for our businesses in various domains.
>
> *— Chuanying Chen, Senior Software Engineer at AliPay*

4.2 Top Online Content Community, Zhihu, Accelerated LLM Training and Deployment with 90% GPU Utilization

Zhihu (NYSE: ZH) is China's leading online content community with 400 million users, 100 million MAU, and 54 billion monthly views. Zhihu trains custom large language models (LLMs) to power its search and recommendation features. To develop LLMs, Zhihu needed a high-performance data access layer to access data from multiple clouds efficiently.

The Zhihu team faced several challenges building a high-performance data access layer for LLMs. First, they needed to find a way to access data from multiple clouds efficiently. Second, they needed to ensure that the data access layer was scalable to meet the growing demands of LLM training and deployment. Third, they needed to ensure the data access layer was reliable and could withstand unexpected failures.

The Zhihu team chose to use Alluxio as the high-performance data access layer for LLMs. Alluxio provides an acceleration service for large-scale data access. Alluxio acts as a unified acceleration solution for large-scale data access to model training and deployment.

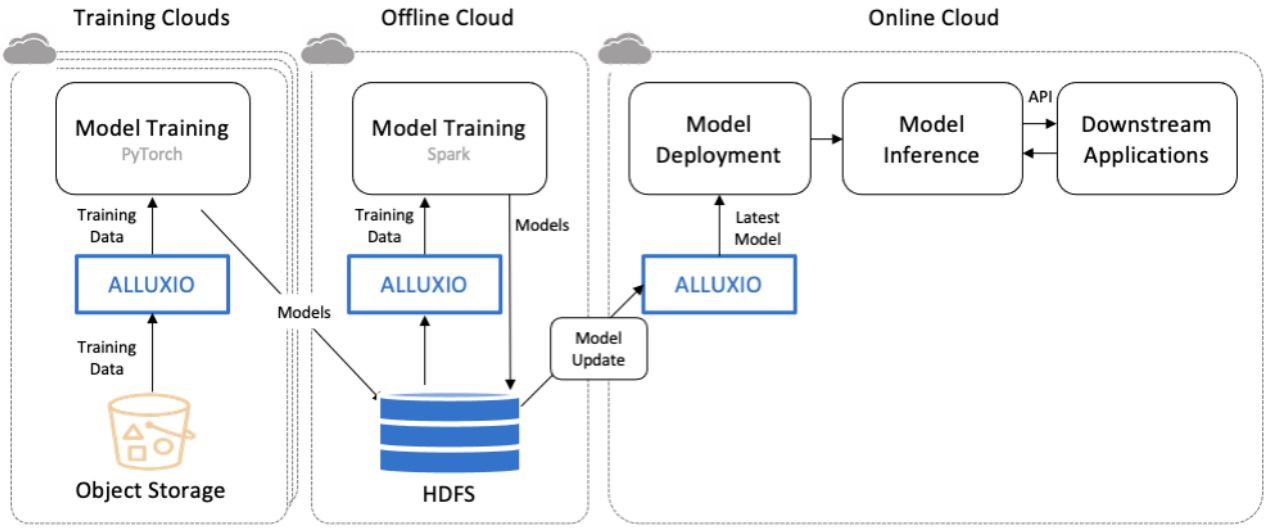


Figure 12: *Zhihu Multi-cloud LLM Pipeline with Alluxio*

After adopting Alluxio, Zhihu saw significant performance, scalability, and reliability improvements. They could train LLMs 2-3 times faster and deploy updated models every minute instead of hours or days. They also saw a 50% reduction in infrastructure costs. Read the full story here >>.

> We choose Alluxio as the high-performance data access layer to tackle our technical challenges. As a result, we've achieved a 90% GPU utilization, 50% reduced infrastructure and operations costs, and accelerated model deployment and update times from several hours to minutes.
>
> *— Mengyu Hu, Software Engineer in the data platform team at Zhihu*

5. Summary

Advancements in AI/ML are simultaneously unlocking opportunities for innovation and presenting challenges for data access. The increasing number of cloud and multicloud environments exacerbates the complexities that must be considered when designing for AI/ML architecture.

Data access plays a vital role in delivering the performance, scale, and mobility of your AI workload needs. Data is everywhere, and there is growing complexity associated with AI platforms. Challenges are related to GPU scarcity, cost, and the multitude of large datasets in silos, which require a solid data and AI platform architecture. As you have seen, Alluxio addresses data access challenges for AI to bridge any model training or deployment demands to any storage in any cloud.

Our experts understand how to architect the end-to-end machine learning pipeline. **Book a meeting** to learn more about solutions tailored to your organization's AI/ML needs.

References

[1] Gartner, “2023 Gartner CIO survey”
[2] Gartner, “2021 Gartner AI in Organizations Survey”
[3] AI and compute, https://openai.com/research/ai-and-compute
[4] Amazon EC2 P4 Instances, https://aws.amazon.com/ec2/instance-types/p4/
[5] Hojin Park, Andy Lu, Greg Ganger, George Amvrosiadis: Multi-region/cloud data sharing scenarios, https://docs.google.com/document/d/1g6tjaFCEjAjGf5-lyRCK-juMHYiXw91le-N8Uu1MCTQ/edit?usp=sharing

Authors of This White Paper

Hope Wang, Developer Advocate at Alluxio
Beinan Wang, Senior Staff Software Engineer at Alluxio
Chunxu Tang, Research Scientist at Alluxio
Lu Qiu, Machine Learning Engineer at Alluxio
Shawn Sun, Software Engineer at Alluxio
Shouwei Chen, Open-source Product Manager at Alluxio
Chenjia Guo, Marketing Coordinator & Analyst at Alluxio

About Alluxio

Alluxio, the developer of the open source data platform, makes it easy to manage your data and serve it from any storage to any compute engine in any environment — on premise, in the cloud, or across clouds. By removing complexities and toil from managing and accessing data infrastructure, Alluxio accelerates and future-proofs your data strategy, delivering performant, accessible, cost-effective, resilient, and secure data applications that power improved outcomes, at any scale. To learn more, contact info@alluxio.com or follow us on Linkedin, or Twitter.