

# Storage System for Large Scale Deep Learning Applications

Author: Yintung Chen (yc4377), Hanxiao Chai (hc3445)

## 1. Introduction

The primary objective of this project is to identify storage systems that can efficiently manage and serve data to GPUs for training deep learning models, particularly when dealing with petabyte-scale datasets. This involves assessing the I/O performance critical to ensuring that the computational power of GPUs is fully utilized without bottlenecks caused by data delivery delays.

This project explores the evolution and optimization of storage systems for large-scale deep learning applications, focusing particularly on the limitations of traditional storage methods like object storage and the development of a more efficient alternative, AIStore.

### 1-1. Motivation

Deep learning models, especially those trained on very large datasets, require high I/O throughput to ensure that GPUs are fed data at a rate that matches their computational speed. When datasets exceed the capacity of a single host, or when their size demands the use of a cluster of storage servers, a sophisticated storage system becomes crucial. In deep learning applications, the timely delivery of data to the GPU is essential for fully leveraging computational power. The efficiency of this data delivery impacts the overall performance of machine learning models, making the storage system a critical component of the infrastructure.

### 1-2. Challenges Addressed

Traditional distributed storage systems, such as HDFS (Hadoop Distributed File System), are known for their scalability. However, they often face significant challenges with the small-file problem. This issue is particularly critical in deep learning, where datasets frequently comprise a vast number of small files, such as images, audio clips, and text documents. Additionally, these systems typically exhibit high latencies that fail to meet the throughput demands of deep learning tasks, thereby substantially hindering performance.

Object storage, while generally performing better than older systems such as HDFS, still struggles with the network overhead and falls short in terms of providing the necessary bandwidth and low-latency access required to fully utilize the computational power of GPUs. These limitations manifest in slower data retrieval times, which can delay training processes and affect overall model performance.

To address these challenges, the project introduces AIStore, a Kubernetes-based distributed storage solution designed as a fast tier in front of traditional object storage providers. AIStore aims to enhance data access speed and reduce latency by employing direct data transfers, parallel reading through object replication across nodes and disks, and a scalable architecture that supports a high

number of storage drives without performance degradation.

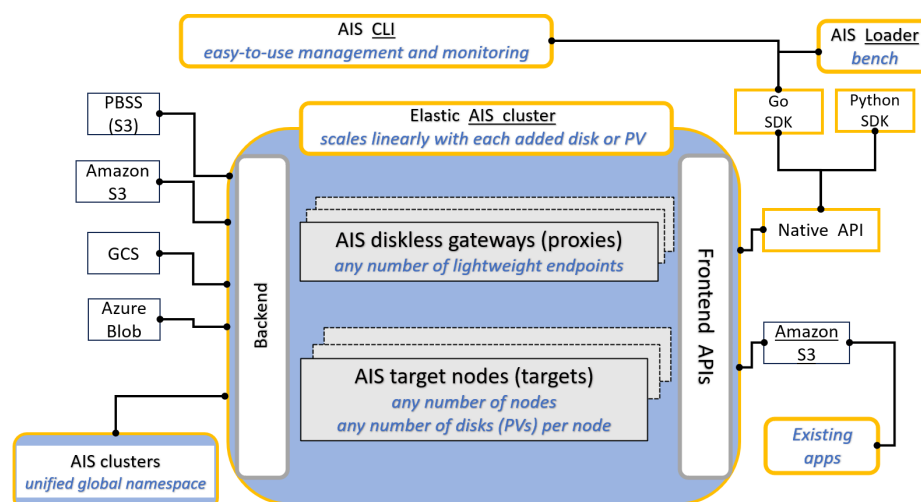
### 1-3. State-of-the-art Solution: AIStore

This introduction sets the groundwork for a detailed discussion on how AIStore represents a significant advancement over traditional storage solutions, promising near-optimal GPU utilization and efficiency improvements across various deep learning workflows. Through this project, we aim to demonstrate that AIStore can overcome the prevalent challenges faced by existing storage systems for large scale deep learning, providing a scalable, high-performance solution that meets the demanding needs of modern deep learning applications.

AIStore distinguishes itself in the landscape of distributed storage solutions by directly addressing the challenges highlighted in previous research, offering an innovative system tailored for the demands of deep learning tasks. Built upon a Kubernetes-based architecture, AIS is inherently scalable, enabling dynamic adjustment to computational and storage needs without manual intervention. This scalability is essential for handling the voluminous and variable demands typical of deep learning environments.

Specifically, AIStore enhances its capabilities as a distributed storage solution by functioning as a fast-tier cache in front of traditional backend object storage providers. It efficiently populates itself on demand (through *cold GET* or *prefetch*), acting as a local cluster cache that can swiftly respond to data retrieval requests. This feature is particularly beneficial for environments where data access speed is crucial. Additionally, AIStore's architecture allows it to boost read performance significantly by facilitating parallel reads across replicated objects on multiple target nodes and disks. This approach not only minimizes the overhead associated with metadata but also ensures that the system's read performance scales almost linearly with the addition of more storage drives. Consequently, AIStore provides a marked improvement over direct object storage access via HTTP, offering faster data delivery that is critical for optimizing the performance of deep learning tasks.

Furthermore, AIStore leverages the inherent flexibility of underlying object storage protocols. It enhances this with a high-performance interface that does not sacrifice I/O throughput, crucial for the demanding read operations of deep learning tasks. By combining these features—scalability through Kubernetes and robust I/O performance—AIStore presents a comprehensive solution that resolves many of the limitations identified in earlier systems, thereby setting a new standard for storage in AI-driven applications.



## 2. Related Works

The paper "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era" [1] underscores the pivotal role of large-scale labeled data in advancing deep learning in vision. The authors examine the impact of scaling up training datasets significantly—by 10x or 100x—on the performance of vision tasks. Utilizing the JFT-300M dataset, which contains over 300 million images with noisy labels across thousands of categories, the study reveals several key findings: First, it confirms that larger datasets can substantially enhance the performance of vision models logarithmically with increasing data volume. Second, the study highlights the enduring value of representation learning, where enhanced base models significantly improve performance across multiple vision tasks. Finally, the research challenges the community to not overlook the significance of data quantity in training models and advocates for efforts towards building even larger datasets. This evidence strongly supports the importance of efficient data delivery mechanisms in training deep learning models, a core focus of our project.

Another paper "PHDFS: Optimizing I/O performance of HDFS in deep learning cloud computing platform" [2] demonstrates an innovative approach to tackle the small file problem in HDFS by grouping small files based on their correlation to form some fixed batches of training data, thus enhancing I/O performance in deep learning tasks. In our project, we acknowledge the necessity of managing large volumes of small files efficiently, as evidenced by the PHDFS approach, which significantly reduces access times and enhances frame processing rates. This solution, however, imposes limitations on the random shuffling of data—a critical process for deep learning models to prevent overfitting and ensure generalized learning. We shift our focus towards maintaining the flexibility of data handling that deep learning models require, particularly the ability to perform random shuffling effectively. Our project aims to identify a storage solution that not only optimizes I/O performance by addressing the small file issue but also preserves, or even enhances, the random access capabilities essential for deep learning, ensuring that the performance improvements in data handling translate directly into superior model training outcomes.

In demonstrating the adaptability of object storage as a foundational layer for specialized systems, the "Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores" [3] paper is particularly instructive. Delta Lake leverages the underlying object storage provided by cloud platforms to offer ACID transaction guarantees, scalable metadata handling, and schema enforcement—features crucial for reliable data manipulation in analytics and machine learning operations. This approach exemplifies how base object storage can be effectively augmented to meet specific application needs, mirroring our project's use of AIStore to enhance object storage capabilities specifically for deep learning tasks. By layering additional functionalities tailored to the unique requirements of deep learning workflows, AIStore aims to optimize data access and management, improving the efficiency and performance of deep learning models in production environments.

### 3. Experiment Design and Evaluation

Our design focuses on developing a robust methodology to monitor, evaluate, and analyze the data transmission rate from storage to memory. This evaluation is pivotal for ensuring that the data feeding into GPUs for computation achieves the maximum possible throughput, thus addressing potential bottlenecks in storage-read capabilities that could impair the overall training efficiency.

#### 3-1. Measurement on Storage Efficiency

To accurately monitor and evaluate the data transmission rate from storage to memory, the system will deploy the [Flexible I/O Tester](#) (fio) — an I/O performance measurement tool. By configuring fio to simulate deep learning workloads, the system will systematically measure and log system-level I/O performance across various storage solutions, including object storage and AIStore. This step is crucial to assess how swiftly data moves across the network and how effectively it is made available to GPUs for processing, highlighting potential bottlenecks in the storage-to-computation pipeline.

Following the initial I/O performance assessment, this project will implement Roofline Modeling to analyze and visualize the compute potential and performance bottlenecks of the GPUs during deep learning tasks. Roofline models provide a graphical representation of how computational performance, measured in FLOPs, is affected by memory bandwidth and computational ceilings. This model will help identify if the observed I/O throughput is sufficient or if it lags behind the GPUs' capability to process data, indicating room for optimization in data delivery systems to fully leverage GPU performance.

#### 3-2. The Experiment Dataset: 36TB ImageNet

For all experiments outlined in this project, we will utilize the official ImageNet dataset. This dataset comprises approximately 36TB of image data and is hosted within a bucket on Google Cloud Storage. This collection of images provides a robust benchmark to assess the effectiveness of our storage solutions under realistic data-intensive scenarios. The ImageNet dataset's size and complexity make it an ideal candidate to test both the I/O efficiency and the overall data delivery performance.

amlc\_imagenet\_prod

Location

us (multiple regions in United States)

Storage class

Standard

Public access

Not public

Protection

Soft Delete

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

INVENTORY REPORTS

OPERATIONS

Folder browser

▼

amlc\_imagenet\_prod

⋮

▼

train/

⋮

▶

n01484850/

⋮

▶

n01491361/

⋮

▶

n01494475/

⋮

▶

n01496331/

⋮

▶

n01514668/

⋮

▶

n01514859/

⋮

▶

n01530575/

⋮

▶

n01531178/

⋮

▶

n01532829/

⋮

▶

n01537544/

⋮

▶

n01558993/

⋮

▶

n01560419/

⋮

▶

n01580077/

⋮

▶

n01582220/

⋮

▶

n01592084/

⋮

Buckets

>

amlc\_imagenet\_prod

>

train

>

n01484850

📄

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

TRANSFER DATA

MANAGE HOLDS

EDIT RETENTION

DOWNLOAD

DELETE

Filter by name prefix only

























Filter

Filter objects and folders

Show

Live objects only

⌵

| <input type="checkbox"/> | Name   | Size     | Type       | Created                  | Storage class |   |
|--------------------------|--|----------|------------|--------------------------|---------------|---|
| <input type="checkbox"/> |  n01484850_10016.JPEG | 97.9 KB  | image/jpeg | Apr 21, 2024, 6:53:50 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10036.JPEG | 86.5 KB  | image/jpeg | Apr 21, 2024, 6:53:28 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10073.JPEG | 71.2 KB  | image/jpeg | Apr 21, 2024, 6:53:10 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10085.JPEG | 144.8 KB | image/jpeg | Apr 21, 2024, 6:52:08 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10086.JPEG | 120 KB   | image/jpeg | Apr 21, 2024, 6:52:07 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10134.JPEG | 74.4 KB  | image/jpeg | Apr 21, 2024, 6:53:03 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10139.JPEG | 92.9 KB  | image/jpeg | Apr 21, 2024, 6:51:56 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10160.JPEG | 23 KB    | image/jpeg | Apr 21, 2024, 6:54:23 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10263.JPEG | 94.8 KB  | image/jpeg | Apr 21, 2024, 6:52:38 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10336.JPEG | 96.1 KB  | image/jpeg | Apr 21, 2024, 6:54:26 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10370.JPEG | 130.1 KB | image/jpeg | Apr 21, 2024, 6:54:29 PM | Standard      |  |
| <input type="checkbox"/> |  n01484850_10385.JPEG | 106.2 KB | image/jpeg | Apr 21, 2024, 6:53:56 PM | Standard      |  |

### ImageNet dataset hosted in google cloud storage a bucket

It's important to note that parts of this dataset have been intentionally duplicated for convenience in setting up the experiment. Since the primary focus of this project is on assessing I/O performance and not the accuracy of any deep learning model, such alterations are unlikely to influence the outcomes of our performance evaluations. This approach ensures that our results accurately reflect the efficiency of the storage solutions in handling large-scale data operations, independent of the data content itself.

## 3-3. Object Storage Performance Experiment

This section of the project focuses on evaluating the read performance of object storage systems, specifically how they manage data retrieval tasks within a cloud environment. Our investigation will primarily assess the bandwidth and overall read performance of fetching objects from a bucket using basic HTTP requests. This method will provide a foundational understanding of how effectively data is retrieved from an object storage system.

After establishing the baseline read performance, we will extend our examination to a practical application by supplying a deep learning training task with data directly from an object storage bucket. Utilizing roofline modeling, we will analyze whether the efficiency of data provisioning is a limiting factor in fully leveraging GPU computational capabilities. This dual approach allows us to not only measure raw performance metrics but also to observe the impact of storage performance on real-world deep learning workloads, potentially identifying bottlenecks that could impede the effective utilization of GPU resources.

### 3-3-1. Read Performance Experiment Setup

In this segment of the experiment, we focus on evaluating the bandwidth and overall read performance of an object storage system. To accomplish this, we employ gcsfuse, a tool that enables mounting Google Cloud Storage buckets as POSIX-compliant file systems. This approach allows us to interact with cloud storage

using straightforward methods to assess performance.

For the test, the ImageNet training dataset, approximately 36TB in size, will be mounted onto a virtual machine using gcsfuse. We will then conduct a series of read operations to measure the I/O performance of this setup through the file system interface. The objective is to determine the efficiency and speed with which data can be retrieved from the cloud object storage, simulating real-world access patterns commonly encountered in deep learning workflows.

```
yc4377@ais-minikube ~ (0.2s)
gcsfuse --implicit-dirs --debug_gcs amlc_imagenet_prod mount-folder/ > /dev/null

yc4377@ais-minikube ~ (3.765s)
ls -al mount-folder/train/n01484850 | head -n 10
total 126124
-rw-r--r-- 1 yc4377 yc4377 100205 Apr 21 22:53 n01484850_10016.JPEG
-rw-r--r-- 1 yc4377 yc4377 88557 Apr 21 22:53 n01484850_10036.JPEG
-rw-r--r-- 1 yc4377 yc4377 72905 Apr 21 22:53 n01484850_10073.JPEG
-rw-r--r-- 1 yc4377 yc4377 148298 Apr 21 22:52 n01484850_10085.JPEG
-rw-r--r-- 1 yc4377 yc4377 122907 Apr 21 22:52 n01484850_10086.JPEG
-rw-r--r-- 1 yc4377 yc4377 76147 Apr 21 22:53 n01484850_10134.JPEG
-rw-r--r-- 1 yc4377 yc4377 95158 Apr 21 22:51 n01484850_10139.JPEG
-rw-r--r-- 1 yc4377 yc4377 23505 Apr 21 22:54 n01484850_10160.JPEG
-rw-r--r-- 1 yc4377 yc4377 97091 Apr 21 22:52 n01484850_10263.JPEG
```

*Utilize gcsfuse to mount a bucket as POSIX file system*

### 3-3-2. Read Performance Evaluation: Flexible I/O Tester

We will utilize the [Flexible I/O Tester](#) (fio) to measure the basic I/O performance of the object storage system under a simulated workload reflective of real-world I/O tasks. The full configuration files of our fio test can be found in our submission, and the following table contains the result of fio experiments across different configurations:

| Block Size\Read Pattern | Sequential Read | Random Read |
|-------------------------|-----------------|-------------|
| 1MB                     | 307 MB/s        | 254 MB/s    |
| 10MB                    | 296 MB/s        | 306 MB/s    |

*Object storage read throughput evaluation on different block size and read pattern*

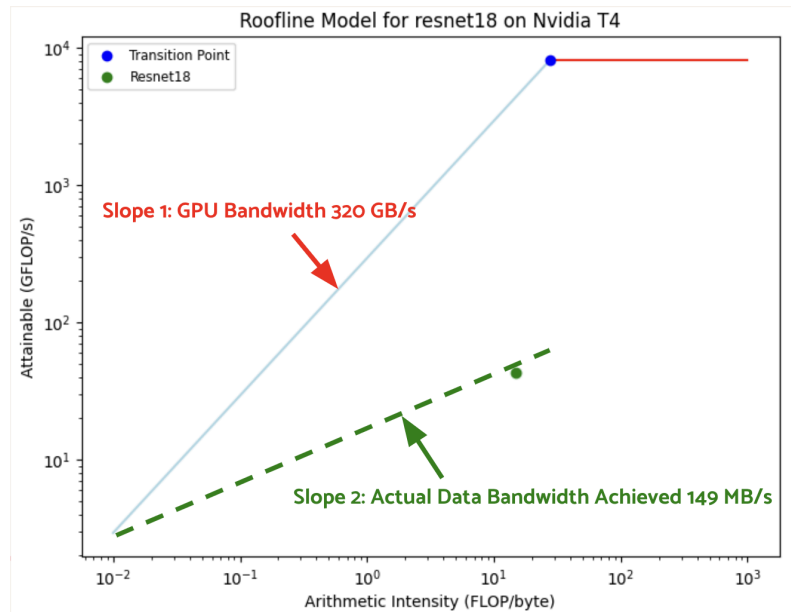
### 3-3-3. Overall Performance Evaluation: Roofline Modeling

The experiment involved training a ResNet18 model using the ImageNet dataset. We utilized sample PyTorch code to train the ResNet18 model for one epoch. The training process was managed through a combination of script executions and performance monitoring tools. We used NVIDIA's Nsight Compute to record data throughput and FLOPs, along with Python's time library to monitor the time taken for the training process.

Execution command:

```
ncu --profile-from-start off --set roofline -f -o resnet18 --metrics
"dram__bytes.sum,smsp__sass_thread_inst_executed_op_fadd_pred_on.sum,smsp__sass_thread_inst_executed_op_fmml_pred_on.sum,smsp__sass_thread_inst_executed_op_ffma_pred_on.sum"
--target-processes all python main.py -a resnet18 --gpu 0 --epochs 1 -b 16 ~/mount-folder/ImageNet
```





*Roofline chart result of ResNet18 training by gcsfuse*

During the experiments, the performance of the ResNet18 model on the NVIDIA Tesla T4 GPU was studied using the Roofline Model. The NVIDIA Tesla T4 GPU, as detailed in the roofline chart from NVIDIA Nsight Compute and confirmed by NVIDIA's official specifications, boasts a bandwidth of approximately 320 GB/s. This substantial figure is starkly higher than the read performance metrics we observed for object storage. Additionally, during our deep learning training task, the actual data bandwidth achieved—represented by a point on the roofline chart at roughly 149 MB/s—correlates closely with our previous measurements using the Flexible I/O Tester. These findings reinforce our initial hypothesis: the read performance of object storage is a significant bottleneck in optimizing the overall performance of GPUs in large dataset deep learning tasks.

This analysis highlights the direct impact of storage I/O performance on machine learning efficiency. The ability of Object Storage to handle large datasets justifies its use. The insights gained from the Roofline Model indicate that Object Storage's I/O performance still does not ideally match the bandwidth requirements needed to maximize GPU utilization. These findings underscore the necessity for storage solutions that can better align with the computational demands of GPUs, leading to the exploration and implementation of AIStore as a potentially more compatible system for deep learning environments.

### 3-4. AIStore Performance Experiment

In this section, we conducted experiments using the same ImageNet dataset stored in an object storage, which served as the backend for our AIStore cluster.

#### 3-4-1. AIStore Kubernetes Cluster Setup

For the kubernetes cluster, we used the Google Kubernetes Engine (GKE) and the terraform [deploy script](#) provided by AIS with the following setting:

##### Cluster 1:

- 3 worker nodes
- 3 gateways pods, 3 targets pods
- 24 storage disks, total raw

##### Cluster 2:

- 10 worker nodes
- 10 gateways pods, 10 targets pods
- 80 storage disks, total raw capacity

capacity of 72 TB

of 800 TB

### 3-4-2. AIStore Cluster Read Performance Experiment

In this segment of our experiment, we configured 6 aisloader clients, each equipped with 50 workers, resulting in a total of 300 workers. This configuration was designed to emulate the I/O workload of frontend clients retrieving data within the cluster, effectively simulating the data access patterns typical of deep learning tasks.

| File Size\Get from | GCS Cold GET (first time) | GCS Warm GET |
|--------------------|---------------------------|--------------|
| 1MB                | 1.15 GB/s                 | 17.66 GB/s   |
| 10MB               | 2.42 GB/s                 | 17.73 GB/s   |

*AIStore in **Cluster 1** read throughput evaluation on different file size and stage*

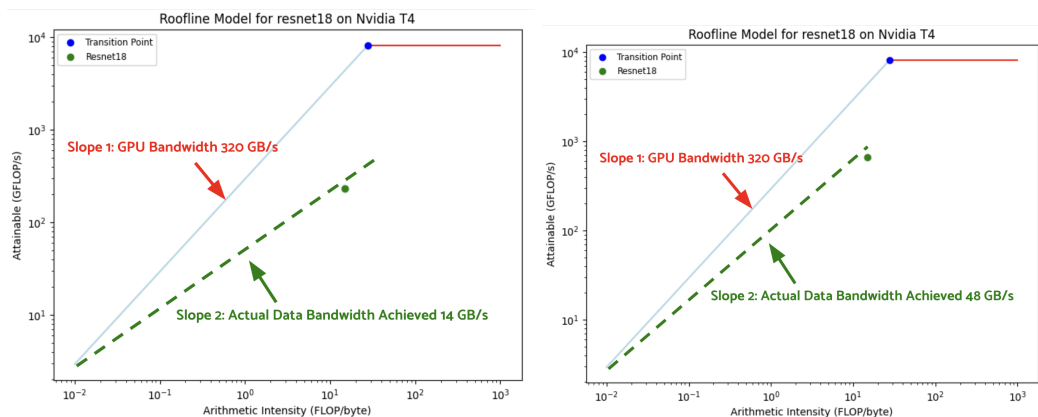
| File Size\Get from | GCS Cold GET (first time) | GCS Warm GET |
|--------------------|---------------------------|--------------|
| 1MB                | 2.98 GB/s                 | 51.49 GB/s   |
| 10MB               | 3.19 GB/s                 | 52.72 GB/s   |

*AIStore in **Cluster 2** read throughput evaluation on different file size and stage*

The experimental results demonstrate that the read performance of AIStore increases with the addition of more worker/target nodes to request and aggregate data from object storage. Notably, a setup with 3 nodes achieved a throughput of 17.73 GB/s, while a 10-node configuration reached 52.72 GB/s. This consistent per-node throughput across various setups indicates that AIStore's performance scales linearly with the addition of target nodes, affirming its scalability.

### 3-4-3. Roofline Modeling: AIStore Cluster as Storage Provider

In this part, we utilized identical sample PyTorch code to train a ResNet18 model for one epoch, mirroring the approach used in our previous experiments.



*Roofline chart result of AIStore Cluster 1*

*Roofline chart result of AIStore Cluster 2*

The roofline charts from our experiments illustrate the significant improvements in I/O throughput achieved by incrementally adding nodes and storage drives to an AIStore cluster. As we expand the cluster, the increase in data delivery



bandwidth suggests that with sufficient scaling, AIStore's throughput could potentially match or exceed the bandwidth capabilities of modern GPUs. This prospect highlights AIStore's capacity to efficiently manage and serve very large datasets for deep learning tasks. By continuously enhancing the cluster with additional resources, AIStore demonstrates its potential as an optimal storage solution capable of adapting to the intensive demands of large-scale deep learning environments, ensuring that data delivery does not become a bottleneck in maximizing GPU utilization.

## 4. Conclusion, Discussion, and Future Works

This project set out to explore efficient data storage solutions capable of supporting the high I/O demands of deep learning applications, particularly those processing large-scale datasets. The investigations focused on two main storage solutions: traditional object storage and the innovative AIStore system, examining their capabilities in terms of read performance and impact on deep learning model training efficiency.

### 4-1. Key Findings in this Project

1. **Object Storage Reconsidered:** Despite its advantages in scalability and general applicability, traditional object storage should not be viewed solely as an end solution for high-performance needs. While it is capable of managing large datasets, our experiments have shown that object storage alone may not always satisfy the demanding data delivery speeds required for maximizing GPU utilization in deep learning tasks. However, the core strength of object storage lies in its flexibility, which makes it an ideal foundational layer upon which additional functionalities can be built. This attribute allows systems like AIStore to enhance object storage by integrating more specialized features tailored to specific performance requirements. Essentially, AIStore capitalizes on the inherent scalability and flexibility of object storage to meet the more stringent I/O demands of modern AI applications, showcasing a strategic enhancement rather than a replacement of traditional storage methods.
2. **Linear Scalability of Distributed System:** AIStore, with its Kubernetes-based architecture, addressed many of these limitations. By enhancing object storage with features like on-demand data populating and efficient data retrieval through its frontend APIs, AIStore proved to significantly improve I/O performance. The scalability tests further indicated that AIStore's throughput potentially matches or exceeds that of current GPU capabilities, suggesting a solution that can scale with the increasing demands of AI applications.

### 4-2. Discussion

The experimental results underscore the importance of storage system architecture in the context of deep learning. The ability of AIStore to outperform traditional object storage models in both speed and efficiency highlights the need for specialized storage solutions in the era of big data and complex machine learning models. Furthermore, the integration of AIStore with existing cloud infrastructure presents a compelling case for its adoption in industry settings where data volume

and velocity are continually increasing.

#### 4-3. Future Works

Looking forward, several areas require further exploration:

1. **Expanded Scalability Tests:** While initial results are promising, larger-scale deployments of AIStore should be tested to confirm its scalability and to identify any potential bottlenecks in even larger distributed environments.
2. **Cross-Platform Compatibility:** Testing AIStore's compatibility and performance across different cloud and under various network conditions would provide a clearer picture of its adaptability and robustness in diverse IT ecosystems.
3. **Longitudinal Studies:** Long-term studies on the maintenance, upgrade, and operational cost implications of deploying AIStore versus traditional storage solutions would provide deeper insights into its practical benefits and challenges.

In conclusion, AIStore presents a viable and potent solution to the challenges posed by the data-intensive requirements of modern deep learning applications. Its ability to evolve and integrate with the latest advancements in cloud technology and AI makes it a critical component in the toolkit of AI researchers and industry practitioners aiming to harness the full potential of their computational resources.

#### 5. References

- [1] Revisiting Unreasonable Effectiveness of Data in Deep Learning Era.
- [2] PHDFS: Optimizing I/O performance of HDFS in deep learning cloud computing platform
- [3] Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores