Technical Report

NetApp StorageGRID Data Lake for Autonomous Driving Workloads

Solution Design

David Arnette, NetApp
September 2020 | TR-4851

Abstract

This document demonstrates the use of NetApp® StorageGRID® object storage as a data repository and management system for machine learning (ML) and deep learning (DL) software development. This paper describes the data flow and requirements in autonomous vehicle software development and the StorageGRID features that streamline the data lifecycle. This solution applies to any multistage data pipeline workflow that is typical in ML and DL development processes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Executive Summary

Autonomous vehicle (AV) software development requires massive quantities of data that are generated in geographically diverse locations. Bringing those data assets together for labeling, processing, and model training is one of the biggest challenges that engineers face when they are developing advanced driver assistance and AV capabilities. As the quantity of data continues to grow into tens and hundreds of exabytes, traditional storage systems cannot scale cost-effectively to reach that capacity. And because of limits in their capacity and addressable space, these systems also impose operational complexity.

To overcome these limitations, object storage is quickly becoming the storage platform of choice to manage these vast quantities of data. Object storage can provide nearly unlimited storage capacity in a single namespace and offers geographically distributed storage options to ease data movement from one location to another. And with simplified access semantics, object storage makes it easy to access data.

The NetApp® StorageGRID® enterprise-grade, software-defined object storage system delivers all these capabilities and integrates seamlessly with the other products that are needed for a complete data pipeline solution. This solution design guide demonstrates the use of StorageGRID object storage as the primary data repository and backbone for an AV software development data pipeline. It includes information about the high-level workflows that are used in the development of deep learning (DL) models for AV and advanced driver assistance systems (ADAS) use. It also includes guidance for using the StorageGRID data lake for the various stages of the AV development workflow.

The target audience for this solution guide includes the following groups:

- Infrastructure architects who design solutions for the development of AI models and software for automotive use cases, including AV.
- Data engineers who want to streamline data management tasks throughout the AV software development cycle.
- Executive decision makers who are interested in achieving the fastest time to market from AI initiatives.

2 Solution Overview

2.1 Autonomous Driving Use Case Summary

Autonomous vehicles have incredible potential to improve roadway safety and efficiency. Deep neural networks (DNNs) and convolutional neural networks (CNNs) enable this autonomy. Components include computer vision models to interpret image and sensor data, prediction models to estimate object positions and movements, and recommendation models to suggest or to execute specific actions. The National Highway Traffic Safety Administration rates levels of autonomy on a scale of 0 to 5:

- **Level 0.** No automation. Zero autonomy; the driver performs all driving tasks.
- **Level 1.** Driver assistance. The vehicle is controlled by the driver, but some driving-assistance features might be included in the vehicle design.
- **Level 2.** Partial automation. The vehicle has combined automated functions, such as acceleration and steering, but the driver must always remain engaged with the driving task and monitor the environment.
- **Level 3.** Conditional automation. The driver must be present but is not required to monitor the environment. The driver must always be ready to take control of the vehicle with notice.
- **Level 4.** High automation. The vehicle can perform all driving functions under certain conditions. The driver can have the option to control the vehicle.
- **Level 5.** Full automation. The vehicle can perform all driving functions under all conditions. The driver can have the option to control the vehicle.

Each level of autonomy from L2 and above requires the development of multiple AI models, each having complex neural network architectures and the need for a very large training dataset. To quantify these numbers and to provide a directional estimate, consider the following:

- One survey car driving 8 hours per day, 250 days per year, can create up to 2,000 hours of video data per year.
- With five cameras at 30 frames per second, this video data equates to 1 billion images per year.
- Using 2-megapixel camera resolution (approximately 2MB per image), each car can generate about 1TB of data per hour, or 2PBs per year.
- Roughly one-third of this data is cleansed and labeled to become useful for model training.
- Best practices require a few million images for the training of simple networks and 5 million to 8 million images to train highly complex networks.

This initial volume of data represents one of the biggest challenges in the development of AV and ADAS systems. Many AV development organizations are working on techniques to minimize the amount of raw data that is moved from edge locations to processing data centers. However, because of the overall amount of data that is required for neural network training, massive amounts of data are still involved.

Development of AI software for autonomous driving is an iterative workflow process that involves several steps:

- **Data collection** is achieved by a fleet of survey cars that collect real-life data by using onboard cameras and sensors such as radar and lidar.
- **Data factory processing** includes data ingestion and transcoding, metadata management and processing, and active learning.
- **Model training** uses labeled datasets to train neural networks for a specific task.
- **Simulation and Replay** involve both component-level testing on real data and the simulation of driving scenarios and conditions, using simulations to validate model behavior under certain conditions.

Each of these processes has different data access and performance requirements, which are described in more detail in section 4, “Data Pipeline Workflow Validation.”

2.2 Data Challenges

The data and accelerated computing requirements for achieving higher levels of autonomy are monumental. Data grows exponentially over the life of a project, and time is a critical factor in this process. Businesses want to make effective use of developer resources and to deliver new capabilities to market in a competitive time frame. The infrastructure in this development process must be able to reduce the time that is involved in each step and scale as the data and the organization grow, without hindering the effectiveness of individual developers.

In addition to AV software development, the data lake often serves other parts of the business. As large manufacturing organizations adopt digital transformation and next-generation manufacturing processes, data flows from many locations into the lake and is served to many other processes from the lake. Existing processes may require POSIX file system access capabilities, and newer cloud-native applications typically use Simple Storage Service (S3) for data access. Some data consumers, such as the labeling process that is described in section 4.2, have minimal storage performance requirements, but model training demands the highest levels of performance.

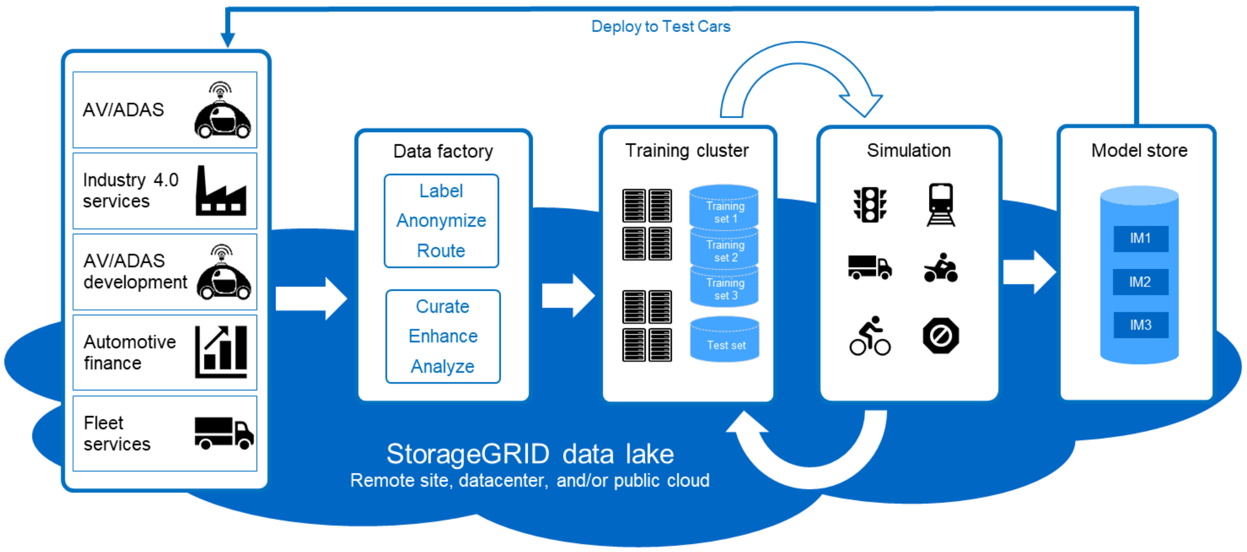
These requirements define the need for a solution that offers massive storage capacity with the ability to tailor cost models to match data lifecycle stages and the ability to provide access through multiple protocols. The solution also must be able to manage with minimal user intervention of data movement between existing systems, between locations or cost tiers, or into special-purpose environments. In the end, such a solution typically requires multiple storage types that are based on specific access and performance requirements and also requires software integration to minimize administrative overhead and to maximize resource utilization.

2.3 StorageGRID Data Lake and Pipeline Solution Overview

NetApp StorageGRID offers a cost-effective and scalable way to store the massive quantity of initial data and provides the ability to input data in one location and retrieve it for processing at another. Object storage also allows the association of rich metadata, enabling detailed data management policies for retention and protection throughout the data and model's lifecycle.

With these capabilities, organizations can build a data lake that spans multiple locations and/or deployment modalities to transform the data lake into a data pipeline that can flow in any direction simultaneously. Data that is ingested at remote locations can be automatically moved for further processing or archiving. Data that is created in a data center can be seamlessly propagated across remote or public cloud resources to support any processing or workflow need. Figure 1 shows a high-level view of the combination data lake and pipeline as they are used in an AV/ADAS software development workflow.

Figure 1) AV/ADAS development process.



3 Technology

3.1 StorageGRID

NetApp StorageGRID is a software-defined, object-based storage solution that supports industry-standard object APIs, including the S3 API and the OpenStack Swift API. StorageGRID provides secure, durable storage for unstructured data at scale. Integrated, metadata-driven lifecycle management policies optimize where data lives throughout its life. To reduce costs, content is placed in the right location, at the right time, and on the right storage tier. StorageGRID is composed of globally distributed, redundant, heterogeneous nodes, which can be integrated with both existing and next-generation client applications.

Key advantages of the StorageGRID system include the following:

- **Massively scalable and easy-to-use global data repository for unstructured data.**
- **Hybrid cloud–enabled system.** Policy-based information lifecycle management (ILM) stores objects to public clouds, including AWS and Microsoft Azure. StorageGRID platform services enable content replication, event notification, and metadata searching on public clouds.

- **Flexible data protection for durability and availability.** Data can be protected by using replication and layered erasure coding. At-rest and in-flight data verification confirms integrity for long-term retention.
- **Support for multiple storage tenant accounts** to segregate the objects that are stored on the system by different entities.
- **Support for software- or hardware-based deployment.** StorageGRID can be deployed on any combination of the following:
  - Virtual machines running in VMware.
  - Docker containers on Linux hosts.
  - StorageGRID engineered appliances: Storage appliances provide object storage. Service appliances provide grid administration and load-balancing services.
- **Federated identity management.** StorageGRID integrates with Active Directory, OpenLDAP, or Oracle Directory Services for user authentication. It supports single sign-on (SSO) by using the Security Assertion Markup Language 2.0 (SAML 2.0) standard to exchange authentication and authorization data between StorageGRID and Active Directory Federation Services (AD FS).

For more information, see NetApp StorageGRID on netapp.com.

3.2 ONTAP AI

The NetApp ONTAP® AI proven architecture, powered by NVIDIA DGX™ systems and NetApp cloud-connected storage systems, was developed and verified by NetApp and NVIDIA. NetApp ONTAP AI tightly integrates DGX systems and NetApp AFF all-flash storage systems and state-of-the-art networking. By eliminating design complexity and guesswork, NetApp ONTAP AI simplifies AI deployments. Customers can start small and grow their systems without interruption while intelligently managing data from the edge to the core to the cloud and back.

ONTAP AI provides the neural network training infrastructure that is used in this validation. Data from the object store is cached locally in an ONTAP NFS export that is mounted on the DGX-1 systems. With this approach, data can be accessed at extremely high speed by multiple GPU servers simultaneously without having to stage the data multiple times.

For more information about ONTAP AI, see NetApp Verified Architectures NVA-1121, NVA-1135, and NVA-1138.

3.3 NetApp AI Control Plane

NetApp AI Control Plane is a full-stack AI data and experiment management solution for data scientists and data engineers. The performance and capabilities of ONTAP combined with Kubernetes and Kubeflow allows organizations to define and to implement AI development workflows that incorporate the near-instant creation of data and model baselines for traceability and versioning. AI Control Plane also enables IT operators to seamlessly replicate data across sites and regions and to swiftly provision Jupyter Notebook workspaces with access to massive datasets. The Kubeflow pipeline feature of AI Control Plane can be used to automate the entire workflow process that is described in Section 4. For more information about the NetApp AI Control Plane, see TR-4798.

3.4 Hardware Requirements

The hardware requirements for this solution vary by site, based on the specific requirements for storage capacity and object throughput performance at each site. StorageGRID offers physical and virtual appliance options to meet any capacity, performance, and footprint requirements.

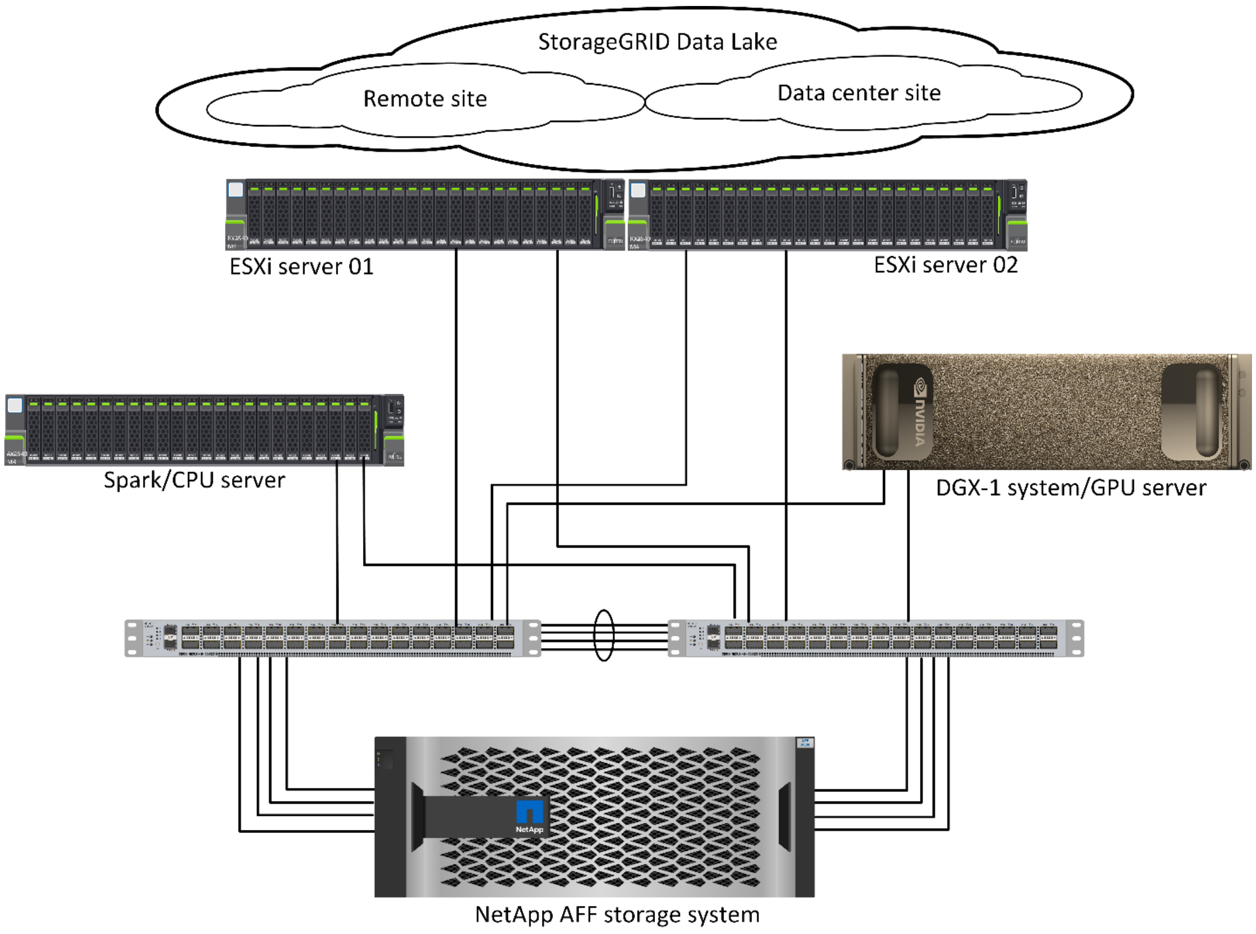
Remote sites require enough capacity and performance to store the daily ingest of raw data from survey cars and to hold that data for some time. Typically, to meet capacity and ingest performance requirements, a minimum installation of three medium-capacity hardware nodes such as StorageGRID

SG5760 is necessary. Virtual nodes can be used to increase throughput or to add capacity with the existing hardware. For sites that require a smaller footprint, StorageGRID SG5712 nodes and/or virtual nodes can be used for capacity along with a caching mechanism like Varnish Software to improve performance.

Because of the variety of tasks that are performed in a data center, those sites have much higher performance and throughput requirements. Lower-performance nodes such as SG5760 are well-suited for high-density cold archive storage. SG6060 nodes offer high performance and very high capacity. And for the most demanding throughput requirements, StorageGRID SGF6024 nodes offer flash memory performance. ILM policies can manage data across these hardware tiers so that data automatically resides in the most cost-effective storage option while pipeline performance requirements are met.

This validation was performed with an entirely virtual grid that was deployed in a VMware virtual infrastructure. The other processes in this solution were validated by using a physical Docker host to run CPU-specific tasks. The ONTAP AI infrastructure was used to perform the model training tasks that are described in section 4.3, "Model Training." Figure 2 shows the environment that was used for this validation. This configuration represents the minimum possible StorageGRID configuration that is necessary to demonstrate functionality. Because StorageGRID supports physical and virtual nodes in the same grid, this configuration can be deployed for testing or for proof-of-concept purposes. It can then seamlessly grow into an exabyte-scale production object store environment simply by adding nodes, sites, and ILM policies and by removing any unnecessary virtual nodes.

Figure 2) Data lake validation environment.



3.5 Software Requirements

Table 1 lists the software components that were used in the validation of this solution.

Table 1) Software requirements.

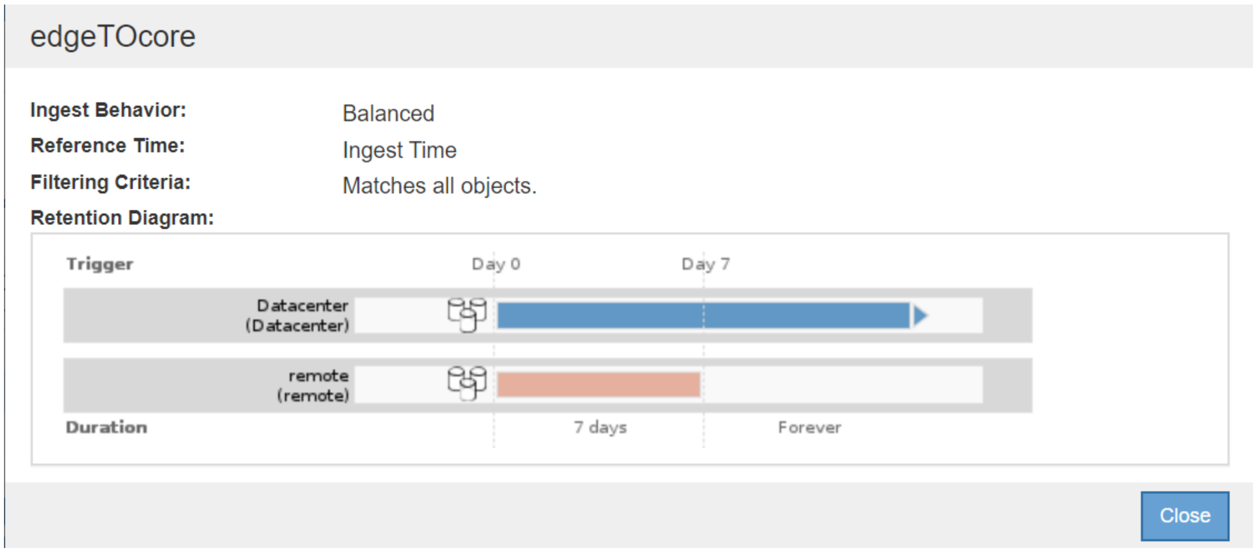
Software	Version or Other Information
StorageGRID object storage software	11.3
Elasticsearch	7.6.2
AWS Boto3 S3 software development kit (SDK)	1.13.15
Spark software	PySpark 2.4.5

3.6 Test Environment

A remote site consisting of three virtual storage nodes and a load-balancer node was deployed for the ingest location. The data center site consisted of three virtual storage nodes, a load balancer, and the StorageGRID admin node. This setup provides a virtual multisite environment to demonstrate the data management capabilities of the StorageGRID data pipeline solution.

Each site was configured into a separate storage pool by using erasure-coding data protection. An ILM rule was configured to match any data that was ingested at the remote site. An ILM policy was activated by using this rule to create a second copy of each object in the data center immediately upon ingest. The policy specified that new objects were to be kept in the remote location for only 7 days, but objects in the data center site were to be kept indefinitely. See Figure 3. In a production pipeline, additional policies would be activated. Policies would move data to other locations as necessary for processing, archive data to cold storage nodes after a specified time, or delete objects after a retention period has expired.

Figure 3) StorageGRID ILM policy.



edgeTOcore

Ingest Behavior:	Balanced
Reference Time:	Ingest Time
Filtering Criteria:	Matches all objects.
Retention Diagram:	

StorageGRID integrates with the Elasticsearch index to provide high-performance search capabilities for StorageGRID objects in the repository, and Kibana can be used for visual interaction with the search index. For this validation, a single Elasticsearch instance was used, but Elasticsearch uses a scale-up architecture that can grow to support any size of deployment.

The individual processes that were used in this validation were all run in Docker containers, both on the CPU host and on the DGX-1 system in the ONTAP AI training cluster. Docker containers provide complete portability of workloads and enable the automation of this entire process by using the NetApp AI Control Plane.

4 Data Pipeline Workflow Validation

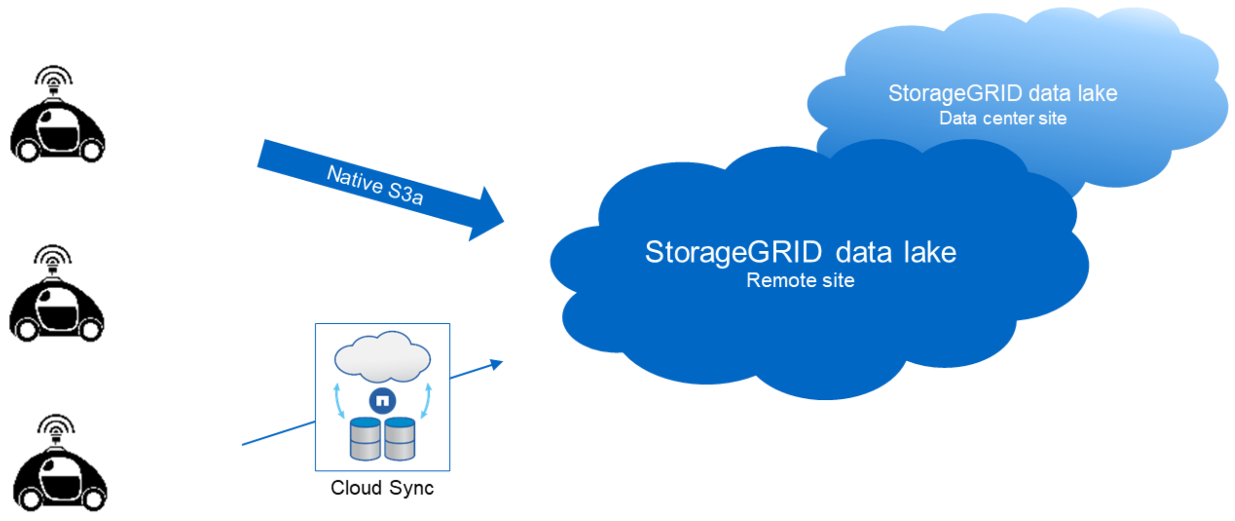
4.1 Data Ingest

The data pipeline starts wherever data is created, whether it is in a remote location or in a corporate data center. In the AV data pipeline, most of the data is generated in survey vehicles that drive many hours each day to capture as many real-world scenarios as possible. When the vehicles return to the garage, that data must be moved from the vehicles to a location that enables the remaining steps in the AV software development cycle. The amount of data that the survey vehicles create is often overwhelming. The initial ingest and movement of data from the remote locations to the primary data center have proven to be among the biggest challenges in AV software development.

NetApp StorageGRID can be deployed in various ways at a remote location to provide an ingest point for the data pipeline. Hardware and software options help customers achieve the performance and capacity levels that they need to support operations that are specific to each site. The remote sites operate independently for the purpose of ingesting and storing data, but they also participate in the ILM policies that are established for the entire grid. As data is ingested to the grid, metadata tags can be applied to indicate location, date, or other features. These tags can then be used to manage the movement of data through the pipeline and to enable searching for specific types of data.

Data can be ingested into the grid at the remote location in various ways, depending on the applications and survey car configurations. Many new applications use the S3 protocol natively and can upload data directly to the grid through standard HTTP API calls. Other applications rely on standard file systems to store captured data, in which case the data must be copied into the object store by using a tool of some kind. NetApp offers data movers, such as the NetApp Cloud Sync service and NetApp XCP Migration Tool, that can be used to upload data to an S3 object store. Several other open-source tools and SDKs are also available. Figure 4 shows the shows some possible ingest options for the first phase of the process.

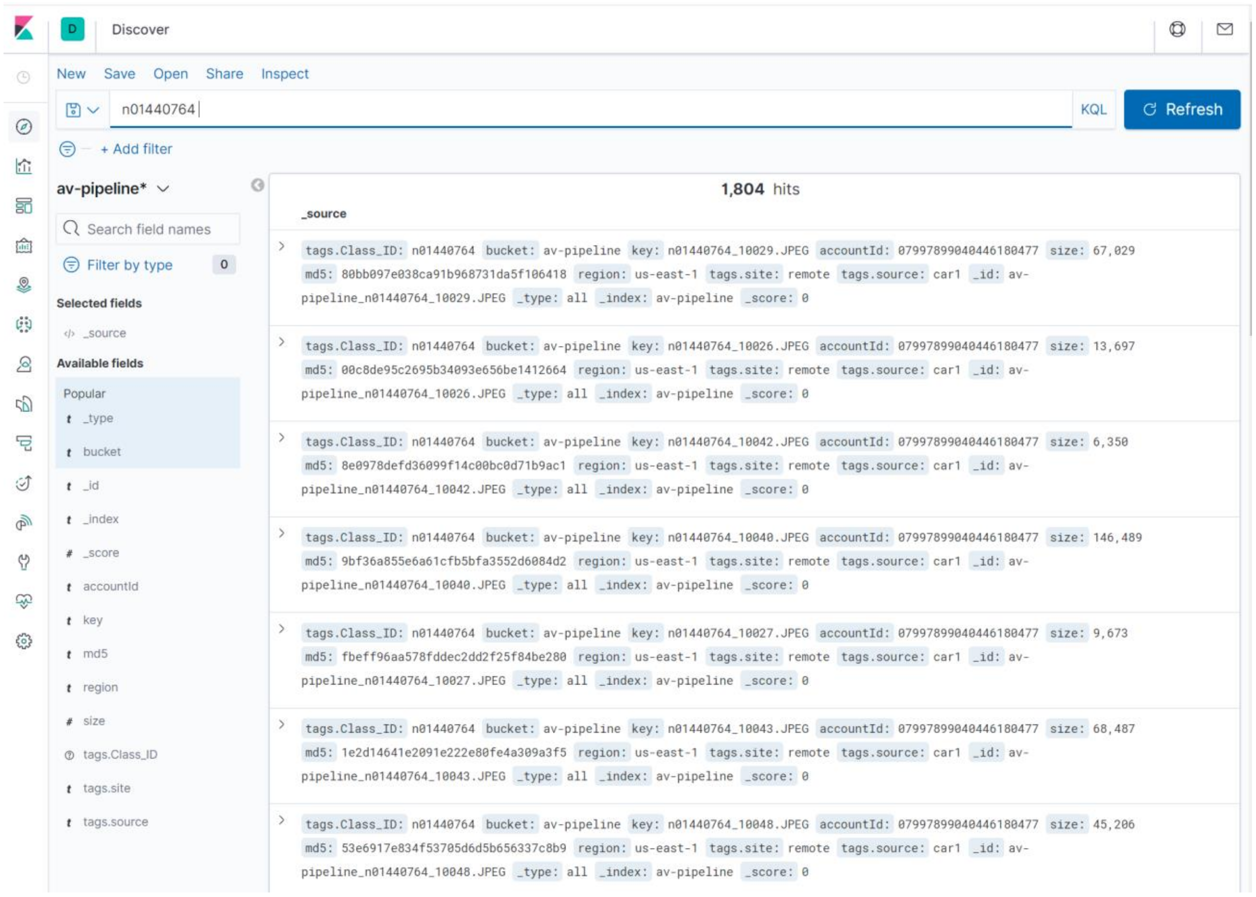
Figure 4) AV pipeline—data ingest.



For this validation, image files from the ImageNet dataset were uploaded to represent raw imagery that had been captured from a survey car. The ImageNet text annotation files were also uploaded separately to represent other data types that might be captured, such as radar or lidar. The data was uploaded by using a Python script that used the AWS Boto3 S3 SDK to transfer files through the S3 protocol. As the images and text files were uploaded, the objects were also tagged with the location name and a source ID. The active ILM policy was configured to maintain a copy of data at the local site for 7 days and to create another copy at the data center site to be retained indefinitely. Figure 3 shows the ILM policy that was configured for this validation.

The screenshot in Figure 5 shows an Elasticsearch query indicating that the ingested data is present in the grid and has the expected metadata tags. Similar queries are used to identify specific data to be processed in later phases of the pipeline.

Figure 5) Elasticsearch index of StorageGRID objects.



1,804 hits

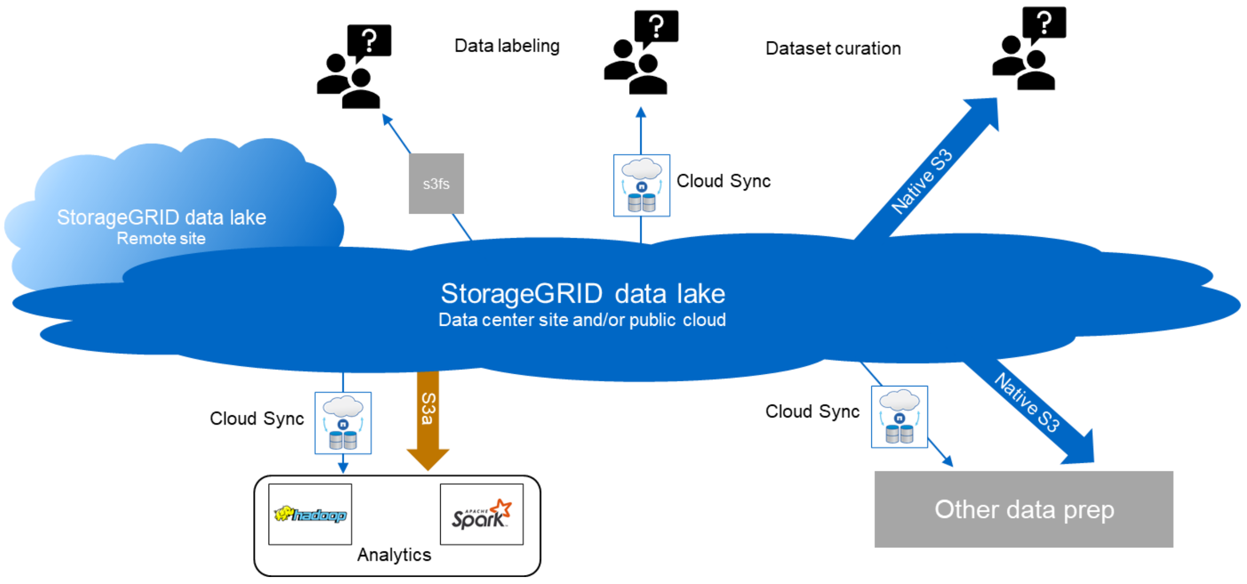
_source

```
tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10029.JPEG accountId: 07997899040446180477 size: 67,029
md5: 80bb097e038ca91b968731da5f106418 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10029.JPEG _type: all _index: av-pipeline _score: 0

tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10026.JPEG accountId: 07997899040446180477 size: 13,697
md5: 00c8de95c2695b34093e656be1412664 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10026.JPEG _type: all _index: av-pipeline _score: 0

tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10042.JPEG accountId: 07997899040446180477 size: 6,350
md5: 8e0978defd36099f14c00bc0d71b9ac1 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10042.JPEG _type: all _index: av-pipeline _score: 0

tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10040.JPEG accountId: 07997899040446180477 size: 146,489
md5: 9bf36a855e6a61cfb5bfa3552d6084d2 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10040.JPEG _type: all _index: av-pipeline _score: 0

tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10027.JPEG accountId: 07997899040446180477 size: 9,673
md5: fbeff96aa578fddec2dd2f25f84be280 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10027.JPEG _type: all _index: av-pipeline _score: 0

tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10043.JPEG accountId: 07997899040446180477 size: 68,487
md5: 1e2d14641e2091e222e80fe4a309a3f5 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10043.JPEG _type: all _index: av-pipeline _score: 0

tags.Class_ID: n01440764 bucket: av-pipeline key: n01440764_10048.JPEG accountId: 07997899040446180477 size: 45,206
md5: 53e6917e834f53705d6d5b656337c8b9 region: us-east-1 tags.site: remote tags.source: car1 _id: av-
pipeline_n01440764_10048.JPEG _type: all _index: av-pipeline _score: 0
```

4.2 Data Factory Processing

After ingest, data in the AV development cycle might go through several possible processing tasks before it is ready for model training. The data flows through what can be called a data factory, where the input is raw data from the survey cars and the output is curated data that is usable by subsequent steps in the AV development cycle. The data factory may be in a single location, or there may be users or parts of the process distributed across multiple locations. Figure 6 shows a few of the many possible tasks that might happen in the virtual data factory and how those steps integrate with the object storage pipeline.

Figure 6) AV pipeline—data factory.

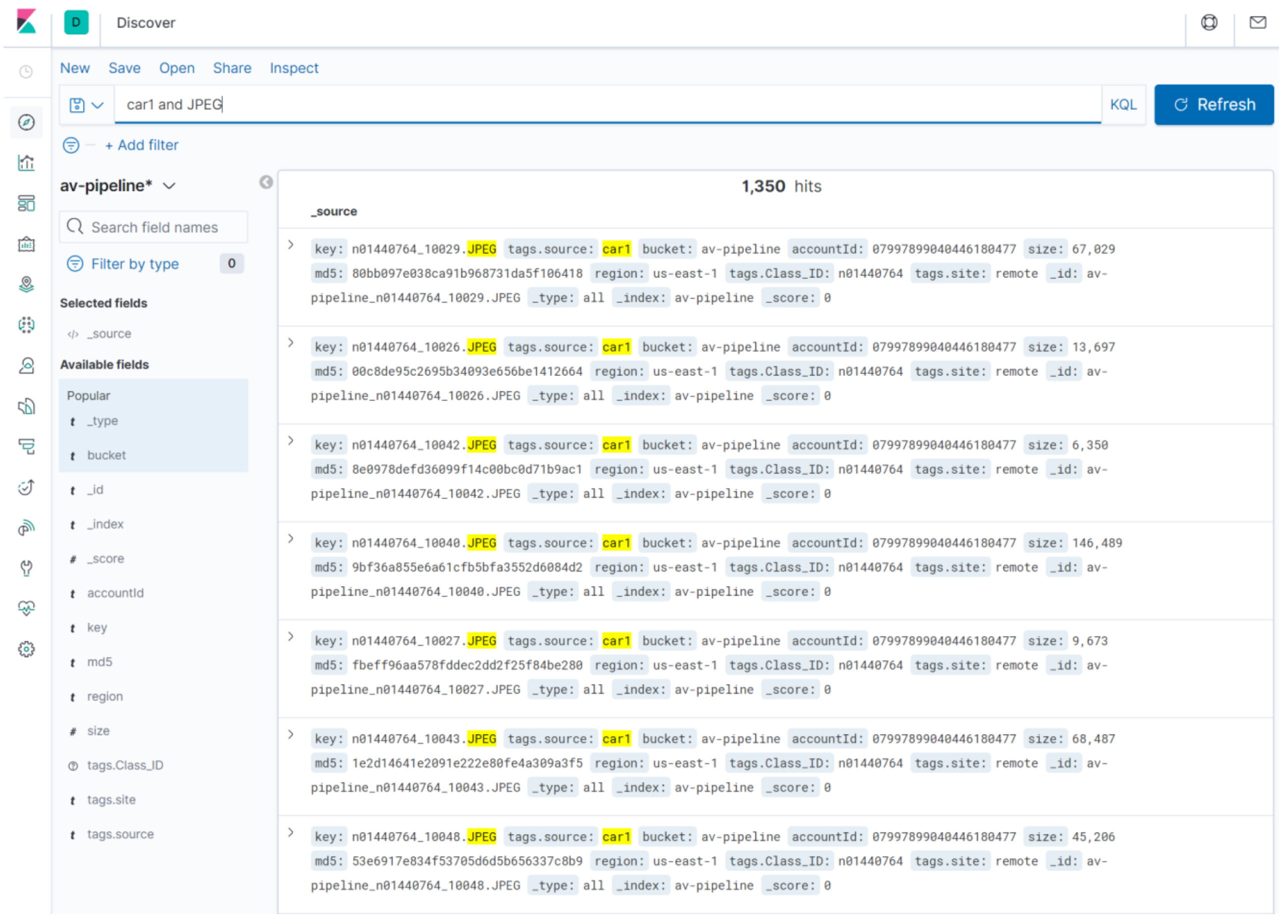


Data Labeling and Annotation

One critical task in the data factory is the labeling of image data for environmental conditions and object identification. This task is usually performed by people, and performance is not a critical requirement. Data is usually accessed directly from the object store by using an application that supports S3 natively. Another option is to use a file translation layer such as s3fs on the client system to provide traditional POSIX access to the object data. Generally, this process can be carried out without transferring any data. The application that is used simply needs a way to access the source data and to upload any annotation or curation data back to the repository.

Figure 7 shows an Elasticsearch query that searches for objects with the `source` tag applied during the ingest phase.

Figure 7) StorageGRID object tagging.



```
car1 and JPEG
```

```
1,350 hits
_source
key: n01440764_10029.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 67,029
md5: 80bb097e038ca91b968731da5f106418 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10029.JPEG _type: all _index: av-pipeline _score: 0

key: n01440764_10026.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 13,697
md5: 00c8de95c2695b34093e656be1412664 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10026.JPEG _type: all _index: av-pipeline _score: 0

key: n01440764_10042.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 6,350
md5: 8e0978defd36099f14c00bc0d71b9ac1 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10042.JPEG _type: all _index: av-pipeline _score: 0

key: n01440764_10040.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 146,489
md5: 9bf36a855e6a61cfb5bfa3552d6084d2 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10040.JPEG _type: all _index: av-pipeline _score: 0

key: n01440764_10027.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 9,673
md5: fbeff96aa578fddec2dd2f25f84be280 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10027.JPEG _type: all _index: av-pipeline _score: 0

key: n01440764_10043.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 68,487
md5: 1e2d14641e2091e222e80fe4a309a3f5 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10043.JPEG _type: all _index: av-pipeline _score: 0

key: n01440764_10048.JPEG tags.source: car1 bucket: av-pipeline accountId: 07997899040446180477 size: 45,206
md5: 53e6917e834f53705d6d5b656337c8b9 region: us-east-1 tags.Class_ID: n01440764 tags.site: remote _id: av-
pipeline_n01440764_10048.JPEG _type: all _index: av-pipeline _score: 0
```

To simulate a data-labeling process, another Python script on a user workstation was used to perform the preceding Elasticsearch query. The query identified the objects to be labeled and confirmed the mapping of images and annotation data. Then it tagged the validated data with a new tag that designated the data as labeled. This tag is used in the dataset curation phase to identify labeled data and to create a usable dataset.

Data Analytics

Another common task in the data factory is the analysis of nonimage data to aid in the selection of data for specific purposes or models. This type of analysis has traditionally been performed with Hadoop, but because of significant architectural limitations, many customers are turning to Apache Spark for this use case. Spark delivers a more dynamic capability to provide data analytics, with less infrastructure, and can interface directly with an S3 object store and load data into memory for high-speed processing.

To simulate an analytics process as part of the object storage pipeline, a standalone Spark word-count job was run against the annotation files that were uploaded during the ingest phase. The annotation files are simple XML files that contain X and Y coordinates for a bounding box around a specific object in an image, along with some other text notations. Another Elasticsearch query was performed to identify the specific objects to be analyzed by using the ingest tag and file type. That list of objects was passed to a containerized standalone Spark word-count job, which loaded the objects into memory and performed the requested analysis. The results were returned as output and were uploaded back to the object store as a separate file. Figure 8 shows the output of the test script, including the list of files returned by the search query, the results of the word-count job, and the file that was uploaded back to the object store.

Figure 8) Example Spark job from a StorageGRID data lake.

```
./run.sh

Starting pyspark container to execute wordcount.py
Searching annotation files for keywords 'pose', 'truncated', and 'difficult'


Started docker container spark-master

ELK query found 530 annotation files for class_id n01484850 in bucket av-pipeline

20/06/03 18:25:48 WARN NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).


[Stage 0:=====================================================>(529 + 1) / 530]

[Stage 1:=====================================================>(529 + 1) / 530]

Saving results
Results saved to bucket av-pipeline as class_id_n01484850_posecount.txt
Confirming successful save of results file:
{
    "hits": {
        "hits": [
            {
                "_score": 11.963264,
                "_type": "all",
                "_id": "av-pipeline_class_id_n01484850_posecount.txt",
                "_source": {
                    "region": "us-east-1",
                    "bucket": "av-pipeline",
                    "key": "class_id_n01484850_posecount.txt",
                    "size": 19996,
                    "accountId": "07997899040446180477",
                    "md5": "fbd95fcdb7adc4dedabd43a84bd6db0e"
                },
                "_index": "av-pipeline"
            }
        ],
        "total": {
            "relation": "eq",
            "value": 1
        },
        "max_score": 11.963264
    },
    "_shards": {
        "successful": 1,
        "failed": 0,
        "skipped": 0,
        "total": 1
    },
    "took": 64,
    "timed_out": false
}
```

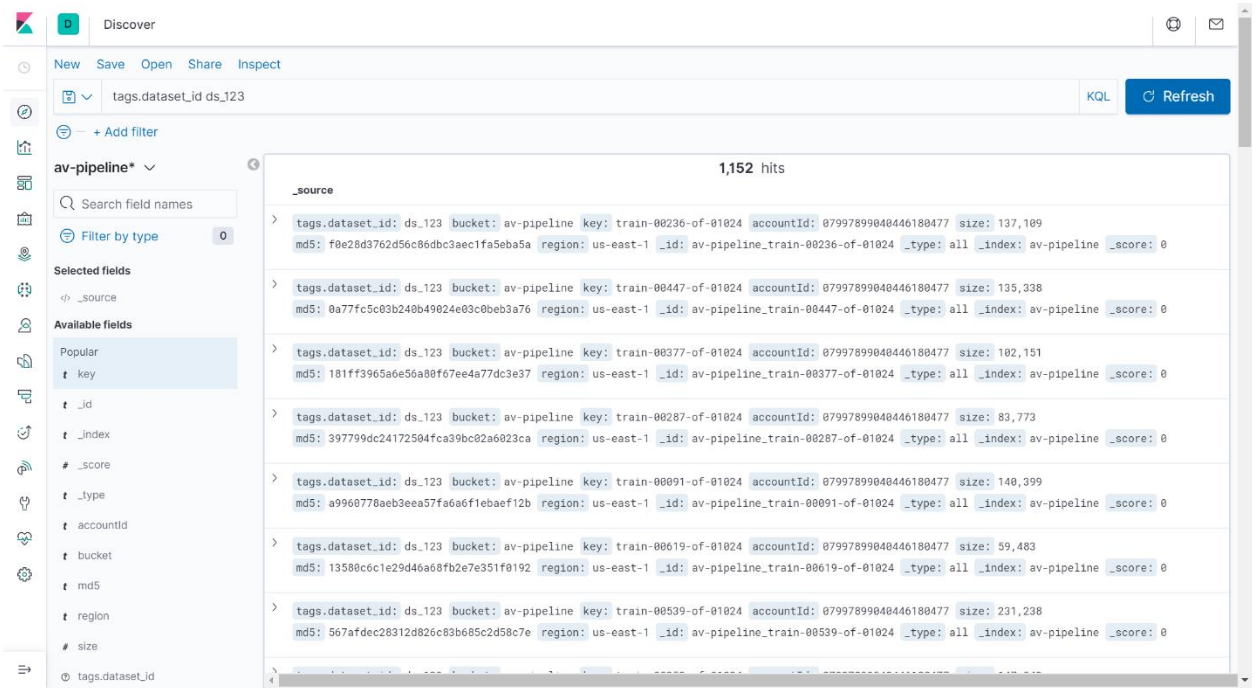
Customers with existing Hadoop environments can use NetApp Cloud Sync to move data from StorageGRID into an NFS share that is exported by the Hadoop Distributed File System (HDFS) cluster.

Dataset Curation

One of the last steps in the data factory processing is the selection of data into specific datasets for model training. This step involves the assembly of specific scenes, conditions, objects, annotations, or anything else required as the direct input for model training, as well as any processing into a usable format.

To complete this simulation of the data factory process, the ImageNet dataset images and annotations were processed into the TFRecord format for use in model training. A final Python script was used to query the Elasticsearch index for the data that was tagged as labeled during the previous step and to download it by using S3 APIs to a local directory. The images and annotations were processed into TFRecord files and then were uploaded back to the object store with a dataset ID tag. Figure 9 shows the results of a final ELK query on the dataset ID that was applied by the TFRecord processing script.

Figure 9) Dataset object tags.



```
tags.dataset_id ds_123
```

```
1,152 hits
_source
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00236-of-01024 accountId: 07997899040446180477 size: 137,109
md5: f0e28d3762d56c86dbc3aec1fa5eba5a region: us-east-1 _id: av-pipeline_train-00236-of-01024 _type: all _index: av-pipeline _score: 0
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00447-of-01024 accountId: 07997899040446180477 size: 135,338
md5: 0a77fc5c03b240b49024e03c0beb3a76 region: us-east-1 _id: av-pipeline_train-00447-of-01024 _type: all _index: av-pipeline _score: 0
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00377-of-01024 accountId: 07997899040446180477 size: 102,151
md5: 181ff3965a6e56a80f67ee4a77dc3e37 region: us-east-1 _id: av-pipeline_train-00377-of-01024 _type: all _index: av-pipeline _score: 0
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00287-of-01024 accountId: 07997899040446180477 size: 83,773
md5: 397799dc24172504fca39bc02a6023ca region: us-east-1 _id: av-pipeline_train-00287-of-01024 _type: all _index: av-pipeline _score: 0
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00091-of-01024 accountId: 07997899040446180477 size: 140,399
md5: a9960778aeb3eea57fa6a6f1ebaef12b region: us-east-1 _id: av-pipeline_train-00091-of-01024 _type: all _index: av-pipeline _score: 0
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00619-of-01024 accountId: 07997899040446180477 size: 59,483
md5: 13580c6c1e29d46a68fb2e7e351f0192 region: us-east-1 _id: av-pipeline_train-00619-of-01024 _type: all _index: av-pipeline _score: 0
tags.dataset_id: ds_123 bucket: av-pipeline key: train-00539-of-01024 accountId: 07997899040446180477 size: 231,238
md5: 567afdec28312d826c83b685c2d58c7e region: us-east-1 _id: av-pipeline_train-00539-of-01024 _type: all _index: av-pipeline _score: 0
```

As noted previously, these processes can occur in a single location or in multiple locations. In any case, the tags that are applied in each step can be used to trigger an ILM policy to move data to other locations if necessary or to other grid storage nodes in the same location. This feature can provide automatic archiving of data as needed or movement of data to higher-performance nodes as required for later processing. In addition, at any point in this workflow, data can be moved to a write once, read many (WORM) bucket. This move preserves the data in an immutable form for regulatory compliance or deletes it, because after processing, the raw inputs are no longer needed. StorageGRID supports regular and WORM buckets within the same grid, and the same ILM policies and rules can apply regardless of which bucket a specific object resides in.

4.3 Model Training

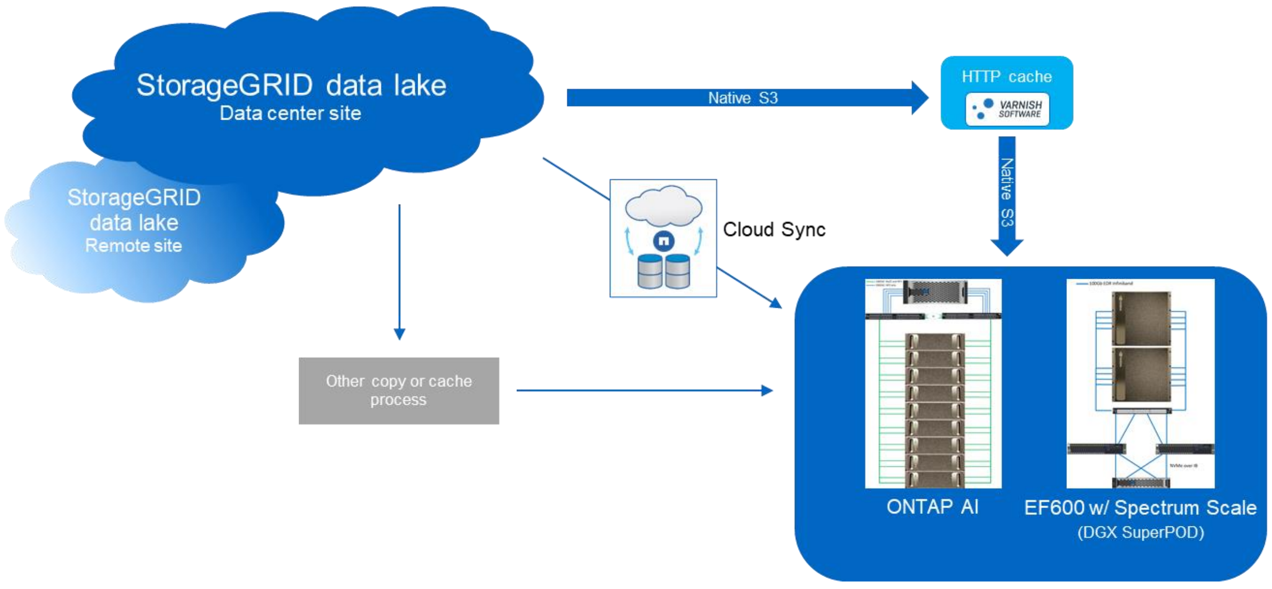
DNN model training requires the highest performance of any step in the AV software development process. The massive size of datasets and the high throughput that is required by the large GPU clusters in AV software development need more performance than object storage can typically provide. To support these requirements, NetApp offers two infrastructure solutions for high-performance training:

- The NetApp ONTAP AI reference architecture.
- The NVIDIA DGX SuperPOD architecture with NetApp EF600 all-flash arrays and the IBM Spectrum Scale file system.

Regardless of which performance storage layer is used, data is moved from the object store data lake into the training cluster. Figure 10 shows how data is moved by using a data mover such as NetApp

Cloud Sync, an S3 caching process such as the Varnish HTTP cache, or another automated copy process through the S3 API.

Figure 10) AV pipeline—model training.



To complete the training phase of this AV data pipeline example, the Elasticsearch index was queried for data with the dataset ID that was specified during the data curation step. That data was downloaded to the ONTAP AI training cluster. A TensorFlow training job was then executed against the dataset. Figure 11 shows the output of the sample training process. The use of shared NFS volumes to store training data allows data movement processes to be handled by other data factory servers while delivering high-performance volume access to the training cluster, optimizing data center resources.

Figure 11) Model training example.

```
root@gpu01:/# python3 download_train.py

Identified 1152 objects in ds_123, downloading to NFS mount point /mnt/av-pipeline/ds_123

Executing Tensorflow training job on 4 GPU

================
== TensorFlow ==
================

NVIDIA Release 20.01-tf1 (build 9338186)
TensorFlow Version 1.15.0

Container image Copyright (c) 2019, NVIDIA CORPORATION.  All rights reserved.
Copyright 2017-2019 The TensorFlow Authors.  All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION.  All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or
file.

NOTE: MOFED driver for multi-node communication was not detected.
      Multi-node communication performance may be reduced.

================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 1 -H localhost:1 bash -c 'sync; echo 1 > /proc/sys/vm/drop_caches'
=========================================
mpirun -allow-run-as-root -np 4 -H localhost:4 -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x
LD_LIBRARY_PATH -x PATH python /netapp/ten
sorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=resnet50 --
```

```
batch_size=256 --device=gpu --force_gpu_compat       ible=True --num_intra_threads=1 --
num_inter_threads=48 --variable_update=horovod --batch_group_size=20 --num_batches=500 --
nodistortions -       -num_gpus=1 --data_format=NCHW --use_fp16=True --use_tf_layers=False --
data_name=imagenet --use_datasets=True --datasets_parallel_interlea       ve_cycle_length=10 --
datasets_sloppy_parallel_interleave=False --data_dir=/mnt/av-pipeline/ds_123/ --num_mounts=1 --
mount_prefix=/mnt/moun       t_%d --datasets_prefetch_buffer_size=2000 --
datasets_use_prefetch=True --datasets_num_private_threads=4 --horovod_device=gpu > /tmp/202006
05_105318_tensorflow_horovod_rdma_resnet50_gpu_4_256_b500_imagenet_nodistort_fp16_r10_m1_nockpt.t
xt 2>&1
Total images/sec = 4272.3825
```

For more information about the model training phase of the AV software development cycle, see TR-4799 NetApp ONTAP AI Reference Architecture for Autonomous Driving Workloads.

In a production development environment, the resulting model is evaluated against other datasets for validation and regression testing. If the new model performs acceptably, the final model is then committed back to the object store. In this validation case, the model was uploaded to the WORM bucket along with a small metadata file containing the dataset ID and training process logs. Both of these files were tagged with an experiment ID to associate them together and to make retrieval easy. The specific data objects in the dataset could also be committed to the WORM bucket, if this action had not already been carried out in an earlier step. The use of StorageGRID to provide both WORM and standard object storage in the same infrastructure enables customers to integrate traceability and repeatability in a single pipeline and at any step of the process as required.

Data lifecycle policies can be applied differently at any stage of the process. For example, they can enable cold archiving of artifacts from any step, removal of data committed to the WORM repository for space efficiency, or movement to other locations for other processing or uses.

4.4 Simulation and Replay

Simulation and replay involve both component-level testing on real data and the simulation of driving scenarios and conditions to validate model behavior. AI-powered autonomous vehicles must be able to respond accurately to diverse driving, traffic, and road situations, such as emergency vehicles, pedestrians, animals, and a virtually infinite number of other variations, obstacles, and environmental factors. Many of these conditions are either too dangerous to capture and test in the real world or are so uncommon that examples are not captured quickly enough for effective model training. Simulation and testing are critical to validate the full hardware and software stack before the AV platform is deployed on the road in a test vehicle.

Simulation is the process of using precaptured images or video as the basis for synthetically creating additional scenarios. By using GPU rendering techniques, more elements or environmental factors can be added to the data stream that is sent to the various car sensors. These additions expand the number of conditions under which a given base data object can be used. The complex neural networks that are required must be exposed to as many examples of every potential scenario as possible. Therefore, this use case expansion significantly decreases the number of specific conditional data samples that I are needed to achieve higher levels of autonomy. From a storage perspective, simulation does not impose significant storage performance requirements, because after a single base scenario is loaded, the process is almost entirely compute focused. Dozens of variations might be generated in real time during a testing cycle before another base scenario is loaded.

Replay involves the use of existing datasets of sensor-specific data to validate the performance of newly trained models and systems. Reference datasets are used to compare new models to the performance of existing models for regression testing, and new datasets might be used to evaluate the performance of existing models against new scenarios or conditions. Replay generally imposes more significant performance requirements, because data is streamed in its existing form to hardware and software stacks. For hardware-in-the-loop (HIL) environments the data is typically streamed in real time, because the sensors must “perceive” the incoming data as if the hardware were on the road. Software-in-the-loop (SIL) testing removes the sensor requirements and significantly accelerates the process. In the SIL case, data can be streamed directly from the object store by using high-performance storage nodes, or it can be staged to the processing environments by using one of the methods described in Section 4 above.

5 Conclusion

The data pipeline for AV and ADAS software development goes through many steps before a trained model is produced, and these steps impose several different data access, performance, and retention requirements. In addition, the quantity of data that is involved means that the storage system must be very cost-effective at a massive scale. NetApp StorageGRID provides highly robust and efficient data protection and archiving capabilities with automated data lifecycle management for managing unstructured data at the largest scales. With its user- and data-level security features, NetApp StorageGRID combined with flexible and integrated search and process workflow capabilities is the optimal solution for the data lake repository for AV and ADAS software development. StorageGRID is also a superior solution for any other enterprise data lake use case.

Acknowledgments

The author would like to acknowledge the contributions of key NetApp team members: Robert Nagy, Aron Klein, Steven Pruchniewski, and Erik Mulder. Sincere appreciation and thanks go to all these individuals, who provided insight and expertise that greatly assisted in the creation of this paper.

Where to Find Additional Information

To learn more about the information that is described in this document, review the following resources:

- NetApp StorageGRID
  https://www.netapp.com/us/products/data-management-software/object-storage-grid-sds.aspx
- NetApp ONTAP AI
  https://www.netapp.com/us/products/ontap-ai.aspx
- Elasticsearch
  https://www.elastic.co/
- PySpark
  https://spark.apache.org/docs/2.4.5/
  https://spark.apache.org/docs/2.4.5/api/python/index.html
- AWS BOTO S3 SDK
  https://aws.amazon.com/sdk-for-python/

Version History

Version	Date	Document Version History
Version 1.0	September 2020	Initial release.

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.