Artificial Intelligence
NetApp Solutions
NetApp
April 03, 2024
# Table of Contents

NetApp Artificial Intelligence Solutions .................................................. 1
AI Converged Infrastructures ................................................................. 1
Data Pipelines, Data Lakes and Management ........................................... 15
Use Cases ............................................................................................... 157
NetApp Artificial Intelligence Solutions

AI Converged Infrastructures

NetApp AFF A400 with Lenovo ThinkSystem SR670 V2 for AI and ML Model Training

TR-4810: NetApp AFF A400 with Lenovo ThinkSystem SR670 V2 for AI and ML Model Training

Sathish Thyagarajan, David Arnette, NetApp
Mircea Troaca, Lenovo

This solution presents a mid-range cluster architecture using NetApp storage and Lenovo servers optimized for artificial intelligence (AI) workloads. It is meant for small- to medium-sized enterprises for which most compute jobs are single node (single or multi-GPU) or distributed over a few computational nodes. This solution aligns with most day-to-day AI training jobs for many businesses.

This document covers testing and validation of a compute and storage configuration consisting of eight-GPU Lenovo SR670V2 servers, a mid-range NetApp AFF A400 storage system and 100GbE interconnect switch. To measure the performance, we used ResNet50 with the ImageNet dataset, a batch size of 408, half precision, CUDA, and cuDNN. This architecture provides an efficient and cost-effective solution for small and medium-sized organizations just starting out with AI initiatives that require the enterprise-grade capabilities of NetApp ONTAP cloud-connected data storage.

Target audience

This document is intended for the following audiences:

• Data scientists, data engineers, data administrators, and developers of AI systems
• Enterprise architects who design solutions for the development of AI models
• Data scientists and data engineers who are looking for efficient ways to achieve deep learning (DL) and machine learning (ML) development goals
• Business leaders and OT/IT decision makers who want to achieve the fastest possible time to market for AI initiatives

Solution architecture

This solution with Lenovo ThinkSystem servers and NetApp ONTAP with AFF storage is designed to handle AI training on large datasets using the processing power of GPUs alongside traditional CPUs. This validation demonstrates high performance and optimal data management with a scale-out architecture that uses either one, two, or four Lenovo SR670 V2 servers alongside a single NetApp AFF A400 storage system. The following figure provides an architectural overview.

This NetApp and Lenovo solution offers the following key benefits:

• Highly efficient and cost-effective performance when executing multiple training jobs in parallel
• Scalable performance based on different numbers of Lenovo servers and different models of NetApp
storage controllers
• Robust data protection to meet low recovery point objectives (RPOs) and recovery time objectives (RTOs) with no data loss
• Optimized data management with snapshots and clones to streamline development workflows

Technology overview

This section introduces the major components of this solution in greater detail.

NetApp AFF systems

NetApp AFF storage systems enable businesses to meet enterprise storage requirements with industry-leading performance, superior flexibility, cloud integration, and best-in-class data management. Designed specifically for flash, AFF systems help accelerate, manage, and protect business-critical data.

NetApp AFF A400 is a mid-range NVMe flash storage system that includes the following features:

• Maximum effective capacity: ~20PB
• Maximum scale-out: 2-24 nodes (12 HA pairs)
• 25GbE and 16Gb FC host support
• 100GbE RDMA over Converged Ethernet (RoCE) connectivity to NVMe expansion storage shelves
• 100GbE RoCE ports can be used for host network attachment if NVMe shelves aren’t attached
• Full 12Gbps SAS connectivity expansion storage shelves
• Available in two configurations:
  ◦ Ethernet: 4x 25Gb Ethernet (SFP28) ports
  ◦ Fiber Channel: 4x 16Gb FC (SFP+) ports
• 100% 8KB random read @.4 ms 400k IOPS

NetApp AFF A250 features for entry level AI/ML deployments include the following:

• Maximum effective capacity: 35PB
• Maximum scale out: 2-24 nodes (12 HA pairs)
• 440k IOPS random reads @1ms
• Built on the latest NetApp ONTAP release ONTAP 9.8 or later
• Two 25Gb Ethernet ports for HA and cluster interconnect

NetApp also offers other storage systems, such as the AFF A800 and AFF A700 that provide higher performance and scalability for larger-scale AI/ML deployments.

NetApp ONTAP

ONTAP 9, the latest generation of storage management software from NetApp, enables businesses to modernize infrastructure and transition to a cloud-ready data center. Leveraging industry-leading data management capabilities, ONTAP enables the management and protection of data with a single set of tools, regardless of where that data resides. Data can also be moved freely to wherever it’s needed: the edge, the core, or the cloud. ONTAP 9 includes numerous features that simplify data management, accelerate and protect critical data, and future-proof infrastructure across hybrid cloud architectures.

Simplify data management

Data management is crucial to enterprise IT operations so that appropriate resources are used for applications and datasets. ONTAP includes the following features to streamline and simplify operations and reduce the total cost of operation:

• **Inline data compaction and expanded deduplication.** Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity. This applies to data stored locally and data tiered to the cloud.

• **Minimum, maximum, and adaptive quality of service (QoS).** Granular QoS controls help maintain performance levels for critical applications in highly shared environments.

• **ONTAP FabricPool.** This feature automatically tiers cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID object storage.

Accelerate and protect data

ONTAP delivers superior levels of performance and data protection and extends these capabilities in the following ways:

• **Performance and lower latency.** ONTAP offers the highest possible throughput at the lowest possible latency.

• **Data protection.** ONTAP provides built-in data protection capabilities with common management across all platforms.

• **NetApp Volume Encryption.** ONTAP offers native volume-level encryption with both onboard and external key management support.

Future-proof infrastructure

ONTAP 9 helps meet demanding and constantly changing business needs:

• **Seamless scaling and nondisruptive operations.** ONTAP supports the nondisruptive addition of capacity to existing controllers as well as to scale-out clusters. Customers can upgrade to the latest technologies, such as NVMe and 32Gb FC, without costly data migrations or outages.

• **Cloud connection.** ONTAP is the most cloud-connected storage management software, with options for
software-defined storage (ONTAP Select) and cloud-native instances (NetApp Cloud Volumes Service) in all public clouds.

- **Integration with emerging applications.** ONTAP offers enterprise-grade data services for next-generation platforms and applications such as OpenStack, Hadoop, and MongoDB by using the same infrastructure that supports existing enterprise apps.

**NetApp FlexGroup volumes**

Training datasets are typically a collection of potentially billions of files. Files can include text, audio, video, and other forms of unstructured data that must be stored and processed to be read in parallel. The storage system must store many small files and must read those files in parallel for sequential and random I/O.

A FlexGroup volume (the following figure) is a single namespace made up of multiple constituent member volumes that is managed and acts like a NetApp FlexVol volume to storage administrators. Files in a FlexGroup volume are allocated to individual member volumes and are not striped across volumes or nodes. They enable the following capabilities:

- Up to 20 petabytes of capacity and predictable low latency for high-metadata workloads
- Up to 400 billion files in the same namespace
- Parallelized operations in NAS workloads across CPUs, nodes, aggregates, and constituent FlexVol volumes

![NetApp FlexGroup](image)

**Lenovo ThinkSystem portfolio**

Lenovo ThinkSystem servers feature innovative hardware, software, and services that solve customers’ challenges today and deliver an evolutionary, fit-for-purpose, modular design approach to address tomorrow’s challenges. These servers capitalize on best-in-class, industry-standard technologies coupled with differentiated Lenovo innovations to provide the greatest possible flexibility in x86 servers.

Key advantages of deploying Lenovo ThinkSystem servers include the following:
• Highly scalable, modular designs that grow with your business
• Industry-leading resilience to save hours of costly unscheduled downtime
• Fast flash technologies for lower latencies, quicker response times, and smarter data management in real time

In the AI area, Lenovo is taking a practical approach to helping enterprises understand and adopt the benefits of ML and AI for their workloads. Lenovo customers can explore and evaluate Lenovo AI offerings in Lenovo AI Innovation Centers to fully understand the value for their particular use case. To improve time to value, this customer-centric approach gives customers proofs of concept for solution development platforms that are ready to use and optimized for AI.

**Lenovo SR670 V2**

The Lenovo ThinkSystem SR670 V2 rack server delivers optimal performance for accelerated AI and high-performance computing (HPC). Supporting up to eight GPUs, the SR670 V2 is suited for the computationally intensive workload requirements of ML, DL, and inference.
With the latest scalable Intel Xeon CPUs that support high-end GPUs (including the NVIDIA A100 80GB PCIe 8x GPU), the ThinkSystem SR670 V2 delivers optimized, accelerated performance for AI and HPC workloads.

Because more workloads use the performance of accelerators, the demand for GPU density has increased. Industries such as retail, financial services, energy, and healthcare are using GPUs to extract greater insights and drive innovation with ML, DL, and inference techniques.

The ThinkSystem SR670 V2 is an optimized, enterprise-grade solution for deploying accelerated HPC and AI workloads in production, maximizing system performance while maintaining data center density for supercomputing clusters with next-generation platforms.

Other features include:

- Support for GPU direct RDMA I/O in which high-speed network adapters are directly connected to the GPUs to maximize I/O performance.
• Support for GPU direct storage in which NVMe drives are directly connected to the GPUs to maximize storage performance.

MLPerf

MLPerf is the industry-leading benchmark suite for evaluating AI performance. In this validation, we used its image-classification benchmark with MXNet, one of the most popular AI frameworks. The MXNet_benchmarks training script was used to drive AI training. The script contains implementations of several popular conventional models and is designed to be as fast as possible. It can be run on a single machine or run in distributed mode across multiple hosts.

Test plan

In this validation, we performed image recognition training as specified by MLPerf v2.0. Specifically, we trained the ResNet v2.0 model with the ImageNet dataset until we reached an accuracy of 76.1%. The main metric is the time to reach the desired accuracy. We also report training bandwidth in images per second to better judge scale-out efficiency.

The primary test case evaluated multiple independent training processes (one per node) running concurrently. This simulates the main use case, a shared system used by multiple data scientists. The second test case evaluated scale-out efficiency.

Test results

The following table summarizes the results for all tests performed for this solution.

<table>
<thead>
<tr>
<th>Test description</th>
<th>Results summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image recognition training: multiple concurrent jobs</td>
<td>Highly efficient performance. All jobs ran at full speed even when the cluster was fully used. The NetApp storage systems delivered training performance comparable to local SSD storage while enabling easy sharing of data between servers.</td>
</tr>
<tr>
<td>Image recognition training: scale out</td>
<td>Highly efficient for up to four nodes. At that point, scale out was less efficient but still feasible. Using a higher-speed computational network improves scalability. The NetApp storage system delivered training performance comparable to local SSD storage while enabling easy sharing of data between servers.</td>
</tr>
</tbody>
</table>

Test configuration

This section describes the tested configurations, the network infrastructure, the SR670 V2 server, and the NetApp storage provisioning details.

Solution architecture

We used the solution components listed in the following table for this validation.
### Solution components

<table>
<thead>
<tr>
<th>Lenovo ThinkSystem servers</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Two SR670 V2 servers each with eight NVIDIA A100 80GB GPU cards</td>
</tr>
<tr>
<td></td>
<td>• Each server contains 2 Intel Xeon Platinum 8360Y CPUs (28 physical cores) and 1TB RAM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linux (Ubuntu – 20.04 with CUDA 11.8)</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• NetApp ONTAP 9.10.1 software</td>
</tr>
<tr>
<td></td>
<td>• 24x 960GB SSDs</td>
</tr>
<tr>
<td></td>
<td>• NFS protocol</td>
</tr>
<tr>
<td></td>
<td>• 1 interface group (ifgrp) per controller, with four logical IP addresses for mount points</td>
</tr>
</tbody>
</table>

In this validation, we used ResNet v2.0 with the ImageNet basis set as specified by MLPerf v2.0. The dataset is stored in a NetApp AFF storage system with the NFS protocol. The SR670s were connected to the NetApp AFF A400 storage system over a 100GbE switch.

ImageNet is a frequently used image dataset. It contains almost 1.3 million images for a total size of 144GB. The average image size is 108KB.

The following figure depicts the network topology of the tested configuration.

![Network Topology Diagram](image)

### Storage controller

The following table lists the storage configuration.
The /a400-100g folder contains the dataset used for ResNet validation.

Test procedure and detailed results

This section describes the detailed test procedure results.

Image recognition training using ResNet in ONTAP

We ran the ResNet50 benchmark with one and two SR670 V2 servers. This test used the MXNet 22.04-py3 NGC container to run the training.

We used the following test procedure in this validation:

1. We cleared the host cache before running the script to make sure that data was not already cached:

   ```
   sync ; sudo /sbin/sysctl vm.drop_caches=3
   ```

2. We ran the benchmark script with the ImageNet dataset in server storage (local SSD storage) as well as on the NetApp AFF storage system.

3. We validated network and local storage performance using the `dd` command.

4. For the single-node run, we used the following command:

   ```
python train_imagenet.py --gpus 0,1,2,3,4,5,6,7 --batch-size 408 --kv -store horovod --lr 10.5 --mom 0.9 --lr-step-epochs pow2 --lars-eta 0.001 --label-smoothing 0.1 --wd 5.0e-05 --warmup-epochs 2 --eval-period 4 --eval-offset 2 --optimizer sgdfastlars --network resnet-v1b-stats-fl --num-layers 50 --num-epochs 37 --accuracy-threshold 0.759 --seed 27081 --dtype float16 --disp-batches 20 --image-shape 4,224,224 --fuse-bn-relu 1 --fuse-bn-relu-gp 1 --bn-group 1 --min-random-area 0.05 --max-random-area 1.0 --conv-algo 1 --force-tensor-core 1 --input-layout NHWC --conv -layout NHWC --batchnorm-layout NHWC --pooling-layout NHWC --batchnorm-mom 0.9 --batchnorm-eps 1e-5 --data-train /data/train.rec --data-train-idx /data/train.idx --data-val /data/val.rec --data-val-idx /data/val.idx --dalio-dont-use-mmap 0 --dalio-hw-decoder-load 0 --dalio-prefetch-queue 5 --dalio-nvjpeg-memory-padding 256 --input-batch-multiplier 1 --dalio-threads 6 --dalio-cache-size 0 --dalio-roi-decode 1 --dalio-preallocate-width 5980 --dalio-preallocate-height 6430 --dalio-tmp-buffer-hint 35556328 --dalio-decoder-buffer-hint 1315942 --dalio-crop -buffer-hint 165581 --dalio-normalize-buffer-hint 441549 --profile 0 --e2e-cuda-graphs 0 --use-dali

5. For the distributed runs, we used the parameter server’s parallelization model. We used two parameter servers per node, and we set the number of epochs to be the same as for the single-node run. We did this because distributed training often takes more epochs due to imperfect synchronization between processes. The different number of epochs can skew comparisons between single-node and distributed cases.

Data read speed: Local versus network storage

The read speed was tested by using the `dd` command on one of the files for the ImageNet dataset. Specifically, we ran the following commands for both local and network data:

```
sync ; sudo /sbin/sysctl vm.drop_caches=3 dd if=/a400-100g/netapp-ra/resnet/data/preprocessed_data/train.rec of=/dev/null bs=512k count=2048
```

Results (average of 5 runs):
Local storage: 1.7 GB/s Network storage: 1.5 GB/s.

Both values are similar, demonstrating that the network storage can deliver data at a rate similar to local storage.

Shared use case: Multiple, independent, simultaneous jobs

This test simulated the expected use case for this solution: multi-job, multi-user AI training. Each node ran its own training while using the shared network storage. The results are displayed in the following figure, which shows that the solution case provided excellent performance with all jobs running at essentially the same speed as individual jobs. The total throughput scaled linearly with the number of nodes.
These graphs present the runtime in minutes and the aggregate images per second for compute nodes that used eight GPUs from each server on 100 GbE client networking, combining both the concurrent training model and the single training model. The average runtime for the training model was 35 minutes and 9 seconds. The individual runtimes were 34 minutes and 32 seconds, 36 minutes and 21 seconds, 34 minutes and 37 seconds, 35 minutes and 25 seconds, and 34 minutes and 31 seconds. The average images per second for the training model were 22,573, and the individual images per second were 21,764; 23,438; 22,556; 22,564; and 22,547.

Based on our validation, one independent training model with a NetApp data runtime was 34 minutes and 54 seconds with 22,231 images/sec. One independent training model with a local data (DAS) runtime was 34 minutes and 21 seconds with 22,102 images/sec. During those runs the average GPU utilization was 96%, as
observed on nvidia-smi. Note that this average includes the testing phase, during which GPUs were not used, while CPU utilization was 40% as measured by mpstat. This demonstrates that the data delivery rate is sufficient in each case.

**Architecture adjustments**

The setup used for this validation can be adjusted to fit other use cases.

**CPU Adjustments**

We used a Skylake Intel Xeon Platinum 8360Y processor for this validation, as recommended by Lenovo. We expect that the equivalent Cascade Lake CPU, an Intel Xeon Gold 6330 processor, would deliver similar performance because this workload is not CPU bound.

**Storage Capacity Increase**

Based on your storage capacity needs, you can increase the share storage (NFS volume) on demand, provided that you have the additional disk shelves and controller models. You can do this from the CLI or from the NetApp web interface of the storage controller as the admin user.

**Conclusion**

The NetApp and Lenovo solution validated here is a flexible scale-out architecture that is ideal for entry into mid-level enterprise AI.

NetApp storage delivers the same or better performance as local SSD storage and offers the following benefits to data scientists, data engineers, and IT decision makers:

- Effortless sharing of data between AI systems, analytics, and other critical business systems. This data sharing reduces infrastructure overhead, improves performance, and streamlines data management across the enterprise.

- Independently scalable compute and storage to minimize costs and improve resource utilization.

- Streamlined development and deployment workflows using integrated snapshots and clones for instantaneous and space-efficient user workspaces, integrated version control, and automated deployment.

- Enterprise-grade data protection for disaster recovery and business continuance.

**Acknowledgments**

- Karthikeyan Nagalingam, Technical Marketing Engineer, NetApp
- Jarrett Upton, Admin, AI Lab Systems, Lenovo

**Where to find additional information**

To learn more about the information described in this document, refer to the following documents and/or websites:

- NetApp All Flash Arrays product page

- NetApp AFF A400 page
  https://docs.netapp.com/us-en/ontap-systems/a400/index.html
NetApp AI with NVIDIA

Overview of ONTAP AI converged infrastructure solutions from NetApp and NVIDIA.

NetApp ONTAP AI with NVIDIA DGX A100 Systems

• Design Guide
• Deployment Guide

NetApp ONTAP AI with NVIDIA DGX A100 Systems and Mellanox Spectrum Ethernet Switches

• Design Guide
• Deployment Guide

NVA-1151-DESIGN: NetApp ONTAP AI with NVIDIA DGX A100 systems design guide

David Arnette and Sung-Han Lin, NetApp

NVA-1151-DESIGN describes a NetApp Verified Architecture for machine learning and artificial intelligence workloads using NetApp AFF A800 storage systems, NVIDIA DGX A100 systems, and NVIDIA Mellanox network switches. It also includes benchmark test results for the architecture as implemented.

NVA-1151-DEPLOY: NetApp ONTAP AI with NVIDIA DGX A100 systems design guide

NVA-1151-DEPLOY: NetApp ONTAP AI with NVIDIA DGX A100 systems

David Arnette, NetApp

NVA-1151-DEPLOY includes storage system deployment instructions for a NetAppVerified Architecture (NVA) for machine learning (ML) and artificial intelligence (AI) workloads using NetApp AFF A800 storage systems, NVIDIA DGX A100 systems, and NVIDIA Mellanox network switches. It also includes instructions for running validation benchmark tests after deployment is complete.
NVA-1153-DESIGN: NetApp ONTAP AI with NVIDIA DGX A100 systems and Mellanox Spectrum Ethernet switches

David Arnette and Sung-Han Lin, NetApp

NVA-1153-DESIGN describes a NetApp Verified Architecture for machine learning (ML) and artificial intelligence (AI) workloads using NetApp AFF A800 storage systems, NVIDIA DGX A100 systems, and NVIDIA Mellanox Spectrum SN3700V 200Gb Ethernet switches. This design features RDMA over Converged Ethernet (RoCE) for the compute cluster interconnect fabric to provide customers with a completely ethernet-based architecture for high-performance workloads. This document also includes benchmark test results for the architecture as implemented.

NVA-1153-DEPLOY: NetApp ONTAP AI with NVIDIA DGX A100 systems and Mellanox Spectrum Ethernet switches

David Arnette, NetApp

NVA-1153-DEPLOY includes storage-system deployment instructions for a NetApp Verified Architecture for machine learning (ML) and artificial intelligence (AI) workloads using NetApp AFF A800 storage systems, NVIDIA DGX A100 systems, and NVIDIA Mellanox Spectrum SN3700V 200Gb Ethernet switches. It also includes instructions for executing validation benchmark tests after deployment is complete.

NetApp EF-Series AI with NVIDIA

Overview of EF-Series AI converged infrastructure solutions from NetApp and NVIDIA.

EF-Series AI with NVIDIA DGX A100 Systems and BeeGFS

- Design Guide
- Deployment Guide
- BeeGFS Deployment Guide

NVA-1156-DESIGN: NetApp EF-Series AI with NVIDIA DGX A100 systems and BeeGFS

Abdel Sadek, Tim Chau, Joe McCormick and David Arnette, NetApp

NVA-1156-DESIGN describes a NetApp Verified Architecture for machine learning (ML) and artificial intelligence (AI) workloads using NetApp EF600 NVMe storage systems, the BeeGFS parallel file system, NVIDIA DGX A100 systems, and NVIDIA Mellanox Quantum QM8700 200Gbps IB switches. This design features 200Gbps InfiniBand (IB)
for the storage and compute cluster interconnect fabric to provide customers with a completely IB-based architecture for high-performance workloads. This document also includes benchmark test results for the architecture as implemented.

NVA-1156-DESIGN: NetApp EF-Series AI with NVIDIA DGX A100 systems and BeeGFS

NVA-1156-DEPLOY: NetApp EF-Series AI with NVIDIA DGX A100 systems and BeeGFS

Abdel Sadek, Tim Chau, Joe McCormick, and David Arnette, NetApp

This document describes a NetApp Verified Architecture for machine learning (ML) and artificial intelligence (AI) workloads using NetApp EF600 NVMe storage systems, the ThinkParQ BeeGFS parallel file system, NVIDIA DGX A100 systems, and NVIDIA Mellanox Quantum QM8700 200Gbps InfiniBand (IB) switches. This document also includes instructions for executing validation benchmark tests after the deployment is complete.

NVA-1156-DEPLOY: NetApp EF-Series AI with NVIDIA DGX A100 systems and BeeGFS

TR-4859: Deploying IBM spectrum scale with NetApp E-Series storage - Installation and validation

Chris Seirer, NetApp

TR-4859 describes the process of deploying a full parallel file system solution based on IBM’s Spectrum Scale software stack. TR-4859 is designed to provide details on how to install Spectrum Scale, validate the infrastructure, and manage the configuration.

TR-4859: Deploying IBM spectrum scale with NetApp E-Series storage - Installation and validation

TR-4815: NetApp AFF A800 and Fujitsu Server PRIMERGY GX2570 M5 for AI and ML model training workloads

David Arnette, NetApp
Takashi Oishi, Fujitsu

This solution focuses on a scale-out architecture to deploy artificial intelligence systems with NetApp storage systems and Fujitsu servers. The solution was validated with MLperf v0.6 model-training benchmarks using Fujitsu GX2570 servers and a NetApp AFF A800 storage system.

TR-4815: NetApp AFF A800 and Fujitsu Server PRIMERGY GX2570 M5 for AI and ML model training workloads

Data Pipelines, Data Lakes and Management

AWS FSx for NetApp ONTAP (FSxN) for MLOps

Author(s):
Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

This section delves into the practical application of AI infrastructure development, providing an end-to-end walkthrough of constructing an MLOps pipeline using FSxN. Comprising three comprehensive examples, it guides you to meet your MLOps needs via this powerful data management platform.

These articles focus on:

1. Part 1 - Integrating AWS FSx for NetApp ONTAP (FSxN) as a private S3 bucket into AWS SageMaker
2. Part 2 - Leveraging AWS FSx for NetApp ONTAP (FSxN) as a Data Source for Model Training in SageMaker
3. Part 3 - Building A Simplified MLOps Pipeline (CI/CT/CD)

By the end of this section, you will have gained a solid understanding of how to use FSxN to streamline MLOps processes.

Part 1 - Integrating AWS FSx for NetApp ONTAP (FSxN) as a private S3 bucket into AWS SageMaker

**Author(s):**
Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

**Introduction**
Using SageMaker as an example, this page provides guidance on configuring FSxN as a private S3 bucket.

For more information about FSxN, please take a look at this presentation (Video Link)

**User Guide**

**Server creation**

Create a SageMaker Notebook Instance

1. Open AWS console. In the search panel, search SageMaker and click the service Amazon SageMaker.

![Amazon SageMaker](image)

2. Open the **Notebook instances** under Notebook tab, click the orange button **Create notebook instance**.
3. In the creation page, Enter the Notebook instance name
   Expand the Network panel
   Leave other entries default and select a VPC, Subnet, and Security group(s). (This VPC and Subnet will be used to create FSxN file system later)
   Click the orange button Create notebook instance at the bottom right.
Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn more

**Notebook instance settings**

- **Notebook instance name**
  - fmx-demo
  - Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

- **Notebook instance type**
  - ml.t3.medium

- **Elastic Inference**
  - Learn more
  - None

- **Platform identifier**
  - Learn more
  - Amazon Linux 2, Jupyter Lab

**Permissions and encryption**

- **IAM role**
  - Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.

- **Create role using the role creation wizard**

- **Root access - optional**
  - Enable - Give users root access to the notebook
  - Disable - Don’t give users root access to the notebook

- **Encryption key - optional**
  - Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

**Network - optional**

- **VPC - optional**
  - Default vpc-0df3956ab1fca2ec9 (172.31.0.0/16)

- **Subnet**
  - Choose a subnet in an availability zone supported by Amazon SageMaker.

- **Security group(s)**

**Direct internet access**

- Enable — Access the internet directly through Amazon SageMaker
- Disable — Access the internet through a VPC

**Git repositories - optional**

**Tags - optional**

Create notebook instance
Create an FSxN File System

1. Open AWS console. In the search panel, search Fsx and click the service **FSx**.

2. Click **Create file system**.

3. Select the first card **FSx for NetApp ONTAP** and click **Next**.
4. In the details configuration page.
   a. Select the **Standard create** option.

   b. Enter the **File system name** and the **SSD storage capacity**.
c. Make sure to use the **VPC** and **subnet** same to the **SageMaker Notebook** instance.
d. Enter the **Storage virtual machine** name and **Specify a password** for your SVM (storage virtual machine).
e. Leave other entries default and click the orange button **Next** at the bottom right.

f. Click the orange button **Create file system** at the bottom right of the review page.
5. It may take about **20-40 minutes** to spin up the FSx file system.

Server Configuration

ONTAP Configuration

1. Open the created FSx file system. Please make sure the status is **Available**.

2. Select the **Administration** tab and keep the **Management endpoint - IP address** and **ONTAP administrator username**.
3. Open the created **SageMaker Notebook instance** and click **Open JupyterLab**.

4. In the Jupyter Lab page, open a new **Terminal**.
5. Enter the ssh command ssh <admin user name>@<ONTAP server IP> to login to the FSxN ONTAP file system. (The user name and IP address are retrieved from the step 2) Please use the password used when creating the Storage virtual machine.

6. Execute the commands in the following order. We use fsxn-ontap as the name for the FSxN private S3 bucket name. Please use the storage virtual machine name for the -vserver argument.

```
vserver object-store-server create -vserver fsxn-svm-demo -object-store-server fsx_s3 -is-http-enabled true -is-https-enabled false
vserver object-store-server user create -vserver fsxn-svm-demo -user s3user
vserver object-store-server group create -name s3group -users s3user -policies FullAccess
vserver object-store-server bucket create fsxn-ontap -vserver fsxn-svm-demo -type nas -nas-path /vol1
```
7. Execute the below commands to retrieve the endpoint IP and credentials for FSxN private S3.

```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1
set adv
vserver object-store-server user show
```

8. Keep the endpoint IP and credential for future use.
Client Configuration

1. In SageMaker Notebook instance, create a new Jupyter notebook.

2. Use the below code as a work around solution to upload files to FSxN private S3 bucket. For a comprehensive code example please refer to this notebook. 

   fsxn_demo.ipynb

```python
# Setup configurations
# -------- Manual configurations --------
seed: int = 77  # Random
bucket_name: str = 'fsxn-ontap'  # The bucket name in ONTAP
aws_access_key_id = '<Your ONTAP bucket key id>'  # Please get this credential from ONTAP
aws_secret_access_key = '<Your ONTAP bucket access key>'  # Please get this credential from ONTAP
fsx_endpoint_ip: str = '<Your FSxN IP address>'  # Please get this IP address from FSXN
# -------- Manual configurations --------

# Workaround
## Permission patch
mkdir -p vol1
sudo mount -t nfs $fsx_endpoint_ip:/vol1 /home/ec2-user/SageMaker/vol1
sudo chmod 777 /home/ec2-user/SageMaker/vol1

## Authentication for FSxN as a Private S3 Bucket
aws configure set aws_access_key_id $aws_access_key_id
```
aws configure set aws_secret_access_key $aws_secret_access_key

## Upload file to the FSxN Private S3 Bucket
%
capture
local_file_path: str = <Your local file path>

aws s3 cp --endpoint-url http://$fsx_endpoint_ip /home/ec2-user
/SageMaker/$local_file_path s3://$bucket_name/$local_file_path

# Read data from FSxN Private S3 bucket
## Initialize a s3 resource client

import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize Fsxn S3 bucket object
# --- Start integrating SageMaker with FSXN ---
# This is the only code change we need to incorporate SageMaker with FSXN
s3_client: boto3.client = boto3.resource('s3',
    region_name = region_name,
    aws_access_key_id = aws_access_key_id,
    aws_secret_access_key = aws_secret_access_key,
    use_ssl = False,
    endpoint_url = f'http://$fsx_endpoint_ip',
    config = boto3.session.Config(
        signature_version = 's3v4',
        s3 = {'addressing_style': 'path'}
    )
)

# --- End integrating SageMaker with FSXN ---

## Read file byte content
bucket = s3_client.Bucket(bucket_name)

binary_data = bucket.Object(data.filename).get() ['Body']

This concludes the integration between FSxN and the SageMaker instance.

**Useful debugging checklist**

- Ensure that the SageMaker Notebook instance and FSxN file system are in the same VPC.
- Remember to run the `set dev` command on ONTAP to set the privilege level to **dev**.
FAQ (As of Sep 27, 2023)

Q: Why am I getting the error "An error occurred (NotImplemented) when calling the CreateMultipartUpload operation: The s3 command you requested is not implemented" when uploading files to FSxN?

A: As a private S3 bucket, FSxN supports uploading files up to 100MB. When using the S3 protocol, files larger than 100MB are divided into 100MB chunks, and the 'CreateMultipartUpload' function is called. However, the current implementation of FSxN private S3 does not support this function.

Q: Why am I getting the error "An error occurred (AccessDenied) when calling the PutObject operations: Access Denied" when uploading files to FSxN?

A: To access the FSxN private S3 bucket from a SageMaker Notebook instance, switch the AWS credentials to the FSxN credentials. However, granting write permission to the instance requires a workaround solution that involves mounting the bucket and running the 'chmod' shell command to change the permissions.

Q: How can I integrate the FSxN private S3 bucket with other SageMaker ML services?

A: Unfortunately, the SageMaker services SDK does not provide a way to specify the endpoint for the private S3 bucket. As a result, FSxN S3 is not compatible with SageMaker services such as Sagemaker Data Wrangler, Sagemaker Clarify, Sagemaker Glue, Sagemaker Athena, Sagemaker AutoML, and others.

Part 2 - Leveraging AWS FSx for NetApp ONTAP (FSxN) as a Data Source for Model Training in SageMaker

Author(s):
Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

Introduction

This tutorial offers a practical example of a computer vision classification project, providing hands-on experience in building ML models that utilize FSxN as the data source within the SageMaker environment. The project focuses on using PyTorch, a deep learning framework, to classify tire quality based on tire images. It emphasizes the development of machine learning models using FSxN as the data source in Amazon SageMaker.

What is FSxN

Amazon FSx for NetApp ONTAP is indeed a fully managed storage solution offered by AWS. It leverages NetApp’s ONTAP file system to provide reliable and high-performance storage. With support for protocols like NFS, SMB, and iSCSI, it allows seamless access from different compute instances and containers. The service is designed to deliver exceptional performance, ensuring fast and efficient data operations. It also offers high availability and durability, ensuring that your data remains accessible and protected. Additionally, the storage capacity of Amazon FSx for NetApp ONTAP is scalable, allowing you to easily adjust it according to your needs.

Prerequisite

Network Environment
FSxN (Amazon FSx for NetApp ONTAP) is an AWS storage service. It includes a file system running on the NetApp ONTAP system and an AWS-managed system virtual machine (SVM) that connects to it. In the provided diagram, the NetApp ONTAP server managed by AWS is located outside the VPC. The SVM serves as the intermediary between SageMaker and the NetApp ONTAP system, receiving operation requests from SageMaker and forwarding them to the underlying storage. To access FSxN, SageMaker must be placed within the same VPC as the FSxN deployment. This configuration ensures communication and data access between SageMaker and FSxN.

**Data Access**

In real-world scenarios, data scientists typically utilize the existing data stored in FSxN to build their machine learning models. However, for demonstration purposes, since the FSxN file system is initially empty after creation, it is necessary to manually upload the training data. This can be achieved by mounting FSxN as a volume to SageMaker. Once the file system is successfully mounted, you can upload your dataset to the mounted location, making it accessible for training your models within the SageMaker environment. This approach allows you to leverage the storage capacity and capabilities of FSxN while working with SageMaker for model development and training.

The data reading process involves configuring FSxN as a private S3 bucket. To learn the detailed configuration instructions, please refer to [Part 1 - Integrating AWS FSx for NetApp ONTAP (FSxN) as a private S3 bucket into AWS SageMaker](#)

**Integration Overview**
The workflow of using training data in FSxN to build a deep learning model in SageMaker can be summarized into three main steps: data loader definition, model training, and deployment. At a high level, these steps form the foundation of an MLOps pipeline. However, each step involves several detailed sub-steps for a comprehensive implementation. These sub-steps encompass various tasks such as data preprocessing, dataset splitting, model configuration, hyperparameter tuning, model evaluation, and model deployment. These steps ensure a thorough and effective process for building and deploying deep learning models using training data from FSxN within the SageMaker environment.

Step-by-Step Integration

Data Loader

In order to train a PyTorch deep learning network with data, a data loader is created to facilitate the feeding of data. The data loader not only defines the batch size but also determines the procedure for reading and preprocessing each record within the batch. By configuring the data loader, we can handle the processing of data in batches, enabling training of the deep learning network.

The data loader consists of 3 parts.

Preprocessing Function

```python
from torchvision import transforms

preprocess = transforms.Compose([      
    transforms.ToTensor(),
    transforms.Resize((224, 224)),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])
```

The above code snippet demonstrates the definition of image preprocessing transformations using the `torchvision.transforms` module. In this tutorial, the preprocess object is created to apply a series of transformations. Firstly, the `ToTensor()` transformation converts the image into a tensor representation. Subsequently, the `Resize 224,224` transformation resizes the image to a fixed size of 224x224 pixels. Finally, the `Normalize()` transformation normalizes the tensor values by subtracting the mean and dividing by the standard deviation along each channel. The mean and standard deviation values used for normalization are...
commonly employed in pre-trained neural network models. Overall, this code prepares the image data for further processing or input into a pre-trained model by converting it to a tensor, resizing it, and normalizing the pixel values.

**The PyTorch Dataset Class**

```python
import torch
from io import BytesIO
from PIL import Image

class FSxNImageDataset(torch.utils.data.Dataset):
    def __init__(self, bucket, prefix='', preprocess=None):
        self.image_keys = [s3_obj.key
                           for s3_obj in list(bucket.objects.filter(Prefix=prefix).all())]
        self.preprocess = preprocess

    def __len__(self):
        return len(self.image_keys)

    def __getitem__(self, index):
        key = self.image_keys[index]
        response = bucket.Object(key)

        label = 1 if key[13:].startswith('defective') else 0

        image_bytes = response.get()[:'Body'].read()
        image = Image.open(BytesIO(image_bytes))
        if image.mode == 'L':
            image = image.convert('RGB')

        if self.preprocess is not None:
            image = self.preprocess(image)

        return image, label
```

This class provides functionality to obtain the total number of records in the dataset and defines the method for reading data for each record. Within the `__getitem__` function, the code utilizes the boto3 S3 bucket object to retrieve the binary data from FSxN. The code style for accessing data from FSxN is similar to reading data from Amazon S3. The subsequent explanation delves into the creation process of the private S3 object bucket.

**FSxN as a private S3 repository**
To read data from FSxN in SageMaker, a handler is created that points to the FSxN storage using the S3 protocol. This allows FSxN to be treated as a private S3 bucket. The handler configuration includes specifying the IP address of the FSxN SVM, the bucket name, and the necessary credentials. For a comprehensive explanation on obtaining these configuration items, please refer to the document at Part 1 - Integrating AWS FSx for NetApp ONTAP (FSxN) as a private S3 bucket into AWS SageMaker.

In the example mentioned above, the bucket object is used to instantiate the PyTorch dataset object. The dataset object will be further explained in the subsequent section.
The PyTorch Data Loader

```python
from torch.utils.data import DataLoader
torch.manual_seed(seed)

# 1. Hyperparameters
batch_size = 64

# 2. Preparing for the dataset
dataset = FSxNImageDataset(bucket, 'dataset/tyre', preprocess=preprocess)

train, test = torch.utils.data.random_split(dataset, [1500, 356])
data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
```

In the example provided, a batch size of 64 is specified, indicating that each batch will contain 64 records. By combining the PyTorch Dataset class, the preprocessing function, and the training batch size, we obtain the data loader for training. This data loader facilitates the process of iterating through the dataset in batches during the training phase.

Model Training

```python
from torch import nn

class TyreQualityClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 32, (3, 3)),
            nn.ReLU(),
            nn.Conv2d(32, 32, (3, 3)),
            nn.ReLU(),
            nn.Conv2d(32, 64, (3, 3)),
            nn.ReLU(),
            nn.Flatten(),
            nn.Linear(64*(224-6)*(224-6), 2)
        )

    def forward(self, x):
        return self.model(x)
```
import datetime

num_epochs = 2
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = TyreQualityClassifier()
fn_loss = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

model.to(device)
for epoch in range(num_epochs):
    for idx, (X, y) in enumerate(data_loader):
        X = X.to(device)
        y = y.to(device)

        y_hat = model(X)

        loss = fn_loss(y_hat, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        print(f"Current Time: {current_time} - Epoch [{epoch+1}/{num_epochs}] - Batch [{idx + 1}] - Loss: {loss}", end='\r')

This code implements a standard PyTorch training process. It defines a neural network model called TyreQualityClassifier using convolutional layers and a linear layer to classify tire quality. The training loop iterates over data batches, computes the loss, and updates the model’s parameters using backpropagation and optimization. Additionally, it prints the current time, epoch, batch, and loss for monitoring purposes.

Model Deployment

Deployment
The code saves the PyTorch model to Amazon S3 because SageMaker requires the model to be stored in S3 for deployment. By uploading the model to Amazon S3, it becomes accessible to SageMaker, allowing for the deployment and inference on the deployed model.
def serialize(self, data):
    transformed_image = preprocess(data)
    tensor_image = torch.Tensor(transformed_image)

    serialized_data = io.BytesIO()
    torch.save(tensor_image, serialized_data)
    serialized_data.seek(0)
    serialized_data = serialized_data.read()

    return serialized_data

class TyreQualityPredictor(Predictor):
    def __init__(self, endpoint_name, sagemaker_session):
        super().__init__(endpoint_name,
                         sagemaker_session=sagemaker_session,
                         serializer=TyreQualitySerializer(),
                         deserializer=JSONDeserializer(),

        sagemaker_model = PyTorchModel(
            model_data=f's3://(bucket_name_default)/pytorch/{model_name}.tar.gz',
            role=sagemaker.get_execution_role(),
            framework_version='2.0.1',
            py_version='py310',
            predictor_cls=TyreQualityPredictor,
            entry_point='inference.py',
            source_dir='code',
        )

        timestamp = int(time.time())
        pytorch_endpoint_name = '{}-{}-{}'.format('tyre-quality-classifier', 'pt',
                                                   timestamp)
        sagemaker_predictor = sagemaker_model.deploy(
            initial_instance_count=1,
            instance_type='ml.p3.2xlarge',
            endpoint_name=pytorch_endpoint_name,
        )

This code facilitates the deployment of a PyTorch model on SageMaker. It defines a custom serializer, TyreQualitySerializer, which preprocesses and serializes input data as a PyTorch tensor. The TyreQualityPredictor class is a custom predictor that utilizes the defined serializer and a JSONDeserializer. The code also creates a PyTorchModel object to specify the model's S3 location, IAM role, framework version, and entry point for inference. The code generates a timestamp and constructs an endpoint name based on the
model and timestamp. Finally, the model is deployed using the deploy method, specifying the instance count, instance type, and generated endpoint name. This enables the PyTorch model to be deployed and accessible for inference on SageMaker.

Inference

```python
image_object = list(bucket.objects.filter('dataset/tyre'))[0].get()
image_bytes = image_object['Body'].read()

with Image.open(BytesIO(image_bytes)) as image:
    predicted_classes = sagemaker_predictor.predict(image)

print(predicted_classes)
```

This is the example of using the deployed endpoint to do the inference.

Part 3 - Building A Simplified MLOps Pipeline (CI/CT/CD)

Author(s):
Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

Introduction

In this tutorial, you will learn how to leverage various AWS services to construct a simple MLOps pipeline that encompasses Continuous Integration (CI), Continuous Training (CT), and Continuous Deployment (CD). Unlike traditional DevOps pipelines, MLOps requires additional considerations to complete the operational cycle. By following this tutorial, you will gain insights into incorporating CT into the MLOps loop, enabling continuous training of your models and seamless deployment for inference. The tutorial will guide you through the process of utilizing AWS services to establish this end-to-end MLOps pipeline.

Manifest

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data storage</td>
<td>AWS FSxN</td>
<td>Refer to Part 1 - Integrating AWS FSx for NetApp ONTAP (FSxN) as a private S3 bucket into AWS SageMaker.</td>
</tr>
<tr>
<td>Data science IDE</td>
<td>AWS SageMaker</td>
<td>This tutorial is based on the Jupyter notebook presented in Part 2 - Leveraging AWS FSx for NetApp ONTAP (FSxN) as a Data Source for Model Training in SageMaker.</td>
</tr>
<tr>
<td>Function to trigger the MLOps pipeline</td>
<td>AWS Lambda function</td>
<td>-</td>
</tr>
<tr>
<td>Cron job trigger</td>
<td>AWS EventBridge</td>
<td>-</td>
</tr>
<tr>
<td>Deep learning framework</td>
<td>PyTorch</td>
<td>-</td>
</tr>
<tr>
<td>AWS Python SDK</td>
<td>boto3</td>
<td>-</td>
</tr>
</tbody>
</table>
Prerequisite

- An pre-configured FSxN file system. This tutorial utilizes data stored in FSxN for the training process.
- A *SageMaker Notebook instance* that is configured to share the same VPC as the FSxN file system mentioned above.
- Before triggering the *AWS Lambda function*, ensure that the *SageMaker Notebook instance* is in *stopped* status.
- The *ml.g4dn.xlarge* instance type is required to leverage the GPU acceleration necessary for the computations of deep neural networks.

Architecture

![Architecture Diagram]

This MLOps pipeline is a practical implementation that utilizes a cron job to trigger a serverless function, which in turn executes an AWS service registered with a lifecycle callback function. The *AWS EventBridge* acts as the cron job. It periodically invokes an *AWS Lambda function* responsible for retraining and redeploying the model. This process involves spinning up the *AWS SageMaker Notebook* instance to perform the necessary tasks.

Step-by-Step Configuration

**Lifecycle configurations**

To configure the lifecycle callback function for the AWS SageMaker Notebook instance, you would utilize *Lifecycle configurations*. This service allow you to define the necessary actions to be performed during when spinning up the notebook instance. Specifically, a shell script can be implemented within the *Lifecycle configurations* to automatically shut down the notebook instance once the training and deployment processes are completed. This is a required configuration as the cost is one of the major consideration in MLOps.

It’s important to note that the configuration for *Lifecycle configurations* needs to be set up in advance. Therefore, it is recommended to prioritize configuring this aspect before proceeding with the other MLOps pipeline setup.

1. To set up a Lifecycle configurations, open the *Sagemaker* panel and navigate to *Lifecycle configurations* under the section *Admin configurations*. 
2. Select the **Notebook Instance** tab and click the **Create configuration** button.
3. Paste the below code to the entry area.

```
#!/bin/bash

set -e
sudo -u ec2-user -i <<'EOF'
# 1. Retraining and redeploying the model
NOTEBOOK_FILE=/home/ec2-user/SageMaker/tyre_quality_classification_local_training.ipynb
echo "Activating conda env"
source /home/ec2-user/anaconda3/bin/activate pytorch_p310
nohup jupyter nbconvert "$NOTEBOOK_FILE"
--ExecutePreprocessor.kernel_name=python --execute --to notebook &
nbconvert_pid=$!
conda deactivate

# 2. Scheduling a job to shutdown the notebook to save the cost
PYTHON_DIR='/home/ec2-user/anaconda3/envs/JupyterSystemEnv/bin/python3.10'
echo "Starting the autostop script in cron"
(crontab -l 2>/dev/null; echo "*/5 * * * * bash -c 'if ps -p $nbconvert_pid > /dev/null; then echo \"Notebook is still running.\" >> /var/log/jupyter.log; else echo \"Notebook execution completed.\" >> /var/log/jupyter.log; $PYTHON_DIR -c \"import boto3;boto3.client(\"sagemaker\")\".stop_notebook_instance(NotebookInstanceName=get_notebook_name())\" >> /var/log/jupyter.log; fi'") | crontab -
EOF
```
4. This script executes the Jupyter Notebook, which handles the retraining and redeployment of the model for inference. After the execution is complete, the notebook will automatically shut down within 5 minutes. To learn more about the problem statement and the code implementation, please refer to Part 2 - Leveraging AWS FSx for NetApp ONTAP (FSxN) as a Data Source for Model Training in SageMaker.

5. After the creation, navigate to Notebook instances, select the target instance, and click Update settings under Actions dropdown.
6. Select the created Lifecycle configuration and click Update notebook instance.
AWS Lambda serverless function

As mentioned earlier, the **AWS Lambda function** is responsible for spinning up the **AWS SageMaker Notebook instance**.

1. To create an **AWS Lambda function**, navigate to the respective panel, switch to the **Functions** tab, and click on **Create Function**.

2. Please file all required entries on the page and remember to switch the Runtime to **Python 3.10**.
3. Please verify that the designated role has the required permission AmazonSageMakerFullAccess and click on the Create function button.
4. Select the created Lambda function. In the code tab, copy and paste the following code into the text area. This code starts the notebook instance named fsxn-ontap.

```python
import boto3
import logging

def lambda_handler(event, context):
    client = boto3.client('sagemaker')
    logging.info('Invoking SageMaker')
    client.start_notebook_instance(NotebookInstanceName='fsxn-ontap')
    return {
        'statusCode': 200,
        'body': f'Starting notebook instance: {notebook_instance_name}'
    }
```
5. Click the **Deploy** button to apply this code change.

6. To specify how to trigger this AWS Lambda function, click on the Add Trigger button.
7. Select EventBridge from the dropdown menu, then click on the radio button labeled Create a new rule. In the schedule expression field, enter \texttt{rate(1 day)}, and click on the Add button to create and apply this new cron job rule to the AWS Lambda function.
After completing the two-step configuration, on a daily basis, the **AWS Lambda function** will initiate the **SageMaker Notebook**, perform model retraining using the data from the **FSxN** repository, redeploy the updated model to the production environment, and automatically shut down the **SageMaker Notebook instance** to optimize cost. This ensures that the model remains up to date.

This concludes the tutorial for developing an MLOps pipeline.

**Hybrid Multicloud MLOps with Domino Data Lab and NetApp**
Hybrid Multicloud MLOps with Domino Data Lab and NetApp

Mike Oglesby, NetApp

Organizations all over the world are currently adopting AI to transform their businesses and processes. Because of this, AI-ready compute infrastructure is often in short supply. Enterprises are adopting hybrid multicloud MLOps architectures in order to take advantage of available compute environments across different regions, data centers, and clouds - balancing cost, availability, and performance.

Domino Nexus, from Domino Data Lab, is a unified MLOps control plane that lets you run data science and machine learning workloads across any compute cluster — in any cloud, region, or on-premises. It unifies data science silos across the enterprise, so you have one place to build, deploy, and monitor models. Likewise, NetApp’s hybrid cloud data management capabilities enable you to bring your data to your jobs and workspaces, no matter where they are running. When you pair Domino Nexus with NetApp, you have the flexibility to schedule workloads across environments without having to worry about data availability. In other words, you have the ability to send your workloads and your data to the appropriate compute environment, enabling you to accelerate your AI deployments while navigating regulations around data privacy and sovereignty.

This solution demonstrates the deployment of a unified MLOps control plane incorporating an on-premises Kubernetes cluster and an Elastic Kubernetes Service (EKS) cluster running in Amazon Web Services (AWS).

Technology Overview

Domino Data Lab

Domino Data Lab powers model-driven businesses with its leading Enterprise AI platform trusted by over 20% of the Fortune 100. Domino accelerates the development and deployment of data science work while increasing collaboration and governance. With Domino, enterprises worldwide can develop better medicines, grow more productive crops, build better cars, and much more. Founded in 2013, Domino is backed by Coatue Management, Great Hill Partners, Highland Capital, Sequoia Capital and other leading investors.

Domino lets enterprises and their data scientists build, deploy and manage AI on a unified, end-to-end platform
— fast, responsibly and cost-effectively. Teams can access all of the data, tools, compute, models, and projects they need across any environment, so they can collaborate, reuse past work, track models in production to improve accuracy, standardize with best practices, and make AI responsible and governed.

- **Open and Flexible**: Access the broadest ecosystem of open source and commercial tools, and infrastructure, for the best innovations and no vendor lock-in.

- **System of Record**: Central hub for AI operations and knowledge across the enterprise, enabling best practices, cross-functional collaboration, faster innovation, and efficiency.

- **Integrated**: Integrated workflows and automation — built for enterprise processes, controls, and governance — satisfy your compliance and regulatory needs.

- **Hybrid Multicloud**: Run AI workloads close to your data anywhere — on-premises, hybrid, any cloud or multi-cloud — for lower cost, optimal performance and compliance.

---

**Domino Nexus**

Domino Nexus is a single pane of glass that lets you run data science and machine learning workloads across any compute cluster — in any cloud, region, or on-premises. It unifies data science silos across the enterprise, so you have one place to build, deploy, and monitor models.

**NetApp BlueXP**

NetApp BlueXP unifies all of NetApp’s storage and data services into a single tool that lets you build, protect, and govern your hybrid multicloud data estate. It delivers a unified experience for storage and data services across on-premises and cloud environments, and enables operational simplicity through the power of AIOps, with the flexible consumption parameters and integrated protection required for today’s cloud-led world.
ONTAP 9, the latest generation of storage management software from NetApp, enables businesses to modernize infrastructure and transition to a cloud-ready data center. Leveraging industry-leading data management capabilities, ONTAP enables the management and protection of data with a single set of tools, regardless of where that data resides. You can also move data freely to wherever it is needed: the edge, the core, or the cloud. ONTAP 9 includes numerous features that simplify data management, accelerate, and protect critical data, and enable next generation infrastructure capabilities across hybrid cloud architectures.

**Simplify data management**

Data management is crucial to enterprise IT operations and data scientists so that appropriate resources are used for AI applications and training AI/ML datasets. The following additional information about NetApp technologies is out of scope for this validation but might be relevant depending on your deployment.

ONTAP data management software includes the following features to streamline and simplify operations and reduce your total cost of operation:

- Inline data compaction and expanded deduplication. Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity. This applies to data stored locally and data tiered to the cloud.
- Minimum, maximum, and adaptive quality of service (AQoS). Granular quality of service (QoS) controls help maintain performance levels for critical applications in highly shared environments.
- NetApp FabricPool. Provides automatic tiering of cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID storage solution. For more information about FabricPool, see TR-4598: FabricPool best practices.

**Accelerate and protect data**

ONTAP delivers superior levels of performance and data protection and extends these capabilities in the following ways:

- Performance and lower latency. ONTAP offers the highest possible throughput at the lowest possible latency.
- Data protection. ONTAP provides built-in data protection capabilities with common management across all platforms.
- NetApp Volume Encryption (NVE). ONTAP offers native volume-level encryption with both onboard and External Key Management support.
- Multitenancy and multifactor authentication. ONTAP enables sharing of infrastructure resources with the highest levels of security.

**Future-proof infrastructure**

ONTAP helps meet demanding and constantly changing business needs with the following features:

- Seamless scaling and nondisruptive operations. ONTAP supports the nondisruptive addition of capacity to existing controllers and to scale-out clusters. Customers can upgrade to the latest technologies, such as NVMe and 32Gb FC, without costly data migrations or outages.
- Cloud connection. ONTAP is the most cloud-connected storage management software, with options for software-defined storage and cloud-native instances in all public clouds.
- Integration with emerging applications. ONTAP offers enterprise-grade data services for next generation platforms and applications, such as autonomous vehicles, smart cities, and Industry 4.0, by using the same
infrastructure that supports existing enterprise apps.

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP is a first-party, fully managed AWS service that provides highly reliable, scalable, high-performing, and feature-rich file storage built on NetApp’s popular ONTAP file system. FSx for ONTAP combines the familiar features, performance, capabilities, and API operations of NetApp file systems with the agility, scalability, and simplicity of a fully managed AWS service.

NetApp Astra Trident

Astra Trident enables consumption and management of storage resources across all popular NetApp storage platforms, in the public cloud or on premises, including ONTAP (AFF, FAS, Select, Cloud, Amazon FSx for NetApp ONTAP), Element software (NetApp HCI, SolidFire), Azure NetApp Files service, and Cloud Volumes Service on Google Cloud. Astra Trident is a Container Storage Interface (CSI) compliant dynamic storage orchestrator that natively integrates with Kubernetes.

Kubernetes

Kubernetes is an open source, distributed, container orchestration platform that was originally designed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF). Kubernetes enables the automation of deployment, management, and scaling functions for containerized applications, and is the dominant container orchestration platform in enterprise environments.

Amazon Elastic Kubernetes Service (EKS)

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service in the AWS cloud. Amazon EKS automatically manages the availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, managing application availability, storing cluster data, and other key tasks. With Amazon EKS, you can take advantage of all the performance, scale, reliability, and availability of AWS infrastructure, as well as integrations with AWS networking and security services.

Architecture

This solution combines Domino Nexus’ hybrid multicloud workload scheduling capabilities with NetApp data services to create a unified hybrid cloud MLOps platform. See the following table for details.

<table>
<thead>
<tr>
<th>Component</th>
<th>Name</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLOps Control Plane</td>
<td>Domino Enterprise AI Platform with Domino Nexus</td>
<td>AWS</td>
</tr>
<tr>
<td>MLOps Platform Compute Environments</td>
<td>Domino Nexus Data Planes</td>
<td>AWS, On-premises data center</td>
</tr>
<tr>
<td>On-premises Compute Platform</td>
<td>Kubernetes with NetApp Astra Trident</td>
<td>On-premises data center</td>
</tr>
<tr>
<td>Cloud Compute Platform</td>
<td>Amazon Elastic Kubernetes Service (EKS) with NetApp Astra Trident</td>
<td>AWS</td>
</tr>
<tr>
<td>On-premises Data Platform</td>
<td>NetApp storage appliance powered by NetApp ONTAP</td>
<td>On-premises data center</td>
</tr>
<tr>
<td>Cloud Data Platform</td>
<td>Amazon FSx for NetApp ONTAP</td>
<td>AWS</td>
</tr>
</tbody>
</table>
Initial Setup

This section describes the initial setup tasks that need to be performed in order to utilize Domino Nexus with NetApp data services in a hybrid environment incorporating an on-premises data center and AWS.

Prerequisites

Before you perform the steps that are outlined in this section, we assume that you have already performed the following tasks:

- You have already deployed and configured your on-premises NetApp ONTAP storage platform. For more information, refer to the NetApp product documentation.
- You have already provisioned an Amazon FSx for NetApp ONTAP instance in AWS. For more information, refer to the Amazon FSx for NetApp ONTAP product page.
- You have already provisioned a Kubernetes cluster in your on-premises data center. For more information, refer to the Domino admin guide.
- You have already provisioned an Amazon EKS cluster in AWS. For more information, refer to the Domino admin guide.
- You have installed NetApp Astra Trident in your on-premises Kubernetes cluster. Additionally, you have configured this Trident instance to use your on-premises NetApp ONTAP storage platform when provisioning and managing storage resources. For more information, refer to the NetApp Astra Trident documentation.
- You have installed NetApp Astra Trident in your Amazon EKS cluster. Additionally, you have configured this Trident instance to use your Amazon FSx for NetApp ONTAP instance when provisioning and managing storage resources. For more information, refer to the NetApp Astra Trident documentation.
- You must have bi-directional network connectivity between your on-premises data center and your Virtual Private Cloud (VPC) in AWS. For more details on the various options for implementing this, refer to the Amazon Virtual Private Network (VPN) documentation.
Install the Domino Enterprise AI Platform in AWS

To install the Domino Enterprise MLOps Platform in AWS, follow the instructions outlined in Domino admin guide. You must deploy Domino in the same Amazon EKS cluster that you previously provisioned. Additionally, NetApp Astra Trident must already be installed and configured in this EKS cluster, and you must specify a Trident-managed storage class as the shared storage class in your domino.yml install configuration file.

Refer to the Domino install configuration reference guide for details on how to specify a shared storage class in your domino.yml install configuration file.

Technical Report TR-4952 walks through the deployment of Domino in AWS with Amazon FSx for NetApp ONTAP and may be a useful reference for troubleshooting any issues that arise.

Enable Domino Nexus

Next, you must enable Domino Nexus. Refer to the Domino admin guide for details.

Deploy a Domino Data Plane in your On-premises Data Center

Next, you must deploy a Domino Data Plane in your on-premises data center. You must deploy this data plane in the on-premises Kubernetes cluster that you previously provisioned. Additionally, NetApp Astra Trident must already be installed and configured in this Kubernetes cluster. Refer to the Domino admin guide for details.

Expose Existing NetApp Volumes to Domino

This section describes the tasks that need to be performed in order to expose existing NetApp ONTAP NFS volumes to the Domino MLOps platform. These same steps apply both on-premises and in AWS.

Why Expose NetApp ONTAP Volumes to Domino?

Using NetApp volumes in conjunction with Domino provides the following benefits:

- You can execute workloads against extremely large datasets by taking advantage of NetApp ONTAP’s scale-out capabilities.
- You can execute workloads across multiple compute nodes without having to copy your data to the individual nodes.
- You can take advantage of NetApp’s hybrid multicloud data movement and sync capabilities in order to access your data across multiple data centers and/or clouds.
- You want to be able to quickly and easily create a cache of your data in a different data center or cloud.

Expose Existing NFS Volumes that were not Provisioned by Astra Trident

If your existing NetApp ONTAP NFS volume was not provisioned by Astra Trident, follow the steps outlined in this sub-section.

Create PV and PVC in Kubernetes

For on-premises volumes, create the PV and PVC in your on-premises Kubernetes cluster. For Amazon FSx for NetApp ONTAP volumes, create the PV and PVC in Amazon EKS.
First, you must create a persistent volume (PV) and persistent volume claim (PVC) in your Kubernetes cluster. To create the PV and PVC, use the NFS PV/PVC example from the Domino admin guide and update the values to reflect to your environment. Be sure to specify the correct values for the namespace, nfs.path, and nfs.server fields. Additionally, we recommend giving your PV and PVC unique names that represent that nature of the data that is stored on the corresponding ONTAP NFS volume. For example, if the volume contains images of manufacturing defects, you might name the PV, pv-mfg-defect-images, and the PVC, pvc-mfg-defect-images.

Register External Data Volume in Domino

Next, you must register an external data volume in Domino. To register an external data volume, refer to the instructions in the Domino admin guide. When registering the volume, be sure to select "NFS" from the 'Volume Type' drop-down menu. After selecting "NFS", you should see your PVC in the 'Available Volumes' list.

Expose Existing Volumes that were Provisioned by Astra Trident

If your existing volume was provisioned by Astra Trident, follow the steps outlined in this sub-section.
Edit Existing PVC

If your volume was provisioned by Astra Trident, then you already have a persistent volume claim (PVC) corresponding to your volume. In order to expose this volume to Domino, you must edit the PVC and add the following label to the list of labels in the `metadata.labels` field:

```
"dominodatalab.com/external-data-volume": "Generic"
```

Register External Data Volume in Domino

Next, you must register an external data volume in Domino. To register an external data volume, refer to the instructions in the Domino admin guide. When registering the volume, be sure to select "Generic" from the 'Volume Type' drop-down menu. After selecting "Generic", you should see your PVC in the 'Available Volumes' list.

Access the same Data Across Different Environments

This section describes the tasks that need to be performed in order to access the same data across different compute environments. In the Domino MLOps platform, compute environments are referred to "data planes." Follow the tasks outlined in this section if your data resides on a NetApp volume in one data plane, but you need to access it in another data plane. This type of scenario is often referred to as "bursting" or, when the destination environment is the cloud, "cloud bursting." This capability is often needed when dealing with constrained or over-subscribed compute resources. For example, if your on-premises compute cluster is over-subscribed, you may want to schedule workloads to the cloud where they can be started immediately.

There are two recommended options for accessing a NetApp volume that resides in a different data plane. These options are outlined in the sub-sections below. Choose one of these options depending on your specific requirements. The benefits and drawbacks of the two options are described in the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Benefits</th>
<th>Drawbacks</th>
</tr>
</thead>
</table>
| Option 1 - Cache | - Simpler workflow  
- Ability to cache a subset of data based on needs  
- Ability to write data back to source  
- No remote copy to manage | - Increased latency on initial data access as cache is hydrated. |
| Option 2 - Mirror | - Full copy of source volume  
- No increased latency due to cache hydration (after mirror operation is complete) | - Must wait for mirror operation to complete before accessing data  
- Must manage a remote copy  
- No ability to write back to source |

Option 1 - Create a Cache of a Volume that Resides in a Different Data Plane

With NetApp FlexCache technology, you can create a cache of a NetApp volume that resides in a different data plane. For example, if you have a NetApp volume in your on-premises data plane, and you need to access that volume in your AWS data plane, you can create a cache of the volume in AWS. This section outlines the tasks that need to be performed in order to create a cache of a NetApp volume that resides in a different data plane.
Create FlexCache Volume in Destination Environment

If the destination environment is your on-premises data center, you will create the FlexCache volume on your on-premises ONTAP system. If the destination environment is AWS, you will create the FlexCache volume on your Amazon FSx for NetApp ONTAP instance.

First, you must create a FlexCache volume in the destination environment.

We recommend using BlueXP to create the FlexCache volume. To create a FlexCache volume with BlueXP, follow the instructions outlined in the BlueXP volume caching documentation.

If you prefer not to use BlueXP, you can use ONTAP System Manager or the ONTAP CLI to create the FlexCache volume. To create a FlexCache volume with System Manager, refer to the instructions outlined in the ONTAP documentation. To create a FlexCache volume with the ONTAP CLI, refer to the instructions outlined in the ONTAP documentation.

If you wish to automate this process, you can use the BlueXP API, the ONTAP REST API, or the ONTAP Ansible collection.

System Manager is not available in Amazon FSx for NetApp ONTAP.

Expose FlexCache Volume to Domino

Next, you must expose the FlexCache volume to the Domino MLOps platform. To expose the FlexCache volume to Domino, follow the instructions outlined in the 'Expose Existing NFS Volumes that were not Provisioned by Astra Trident' sub-section of the 'Expose Existing NetApp Volumes to Domino' section of this solution.

Now, you will be able to mount the FlexCache volume when launching jobs and workspaces in the destination data plane as shown in the following screenshots.

Before Creating FlexCache Volume
After Exposing FlexCache Volume to Domino
Option 2 - Replicate a Volume that Resides in a Different Data Plane

With NetApp SnapMirror data replication technology, you can create a copy of a NetApp volume that resides in a different data plane. For example, if you have a NetApp volume in your on-premises data plane, and you need to access that volume in your AWS data plane, you can create a copy of the volume in AWS. This section outlines the tasks that need to be performed in order to create a copy of a NetApp volume that resides in a different data plane.

Create SnapMirror Relationship

First, you must create a SnapMirror relationship between your source volume and a new destination volume in the destination environment. Note that the destination volume will be created as part of the process of creating the SnapMirror relationship.
We recommend using BlueXP to create the SnapMirror relationship. To create a SnapMirror relationship with BlueXP, follow the instructions outlined in the BlueXP replication documentation.

If you prefer not to use BlueXP, you can use ONTAP System Manager or the ONTAP CLI to create the SnapMirror relationship. To create a SnapMirror relationship with System Manager, refer to the instructions outlined in the ONTAP documentation. To create a SnapMirror relationship with the ONTAP CLI, refer to the instructions outlined in the ONTAP documentation.

If you wish to automate this process, you can use the BlueXP API, the ONTAP REST API, or the ONTAP Ansible collection.

System Manager is not available in Amazon FSx for NetApp ONTAP.

Break SnapMirror Relationship

Next, you must break the SnapMirror relationship in order to activate the destination volume for data access. Wait until the initial replication is complete before performing this step.

You can determine whether or not the replication is complete by checking the mirror state in BlueXP, ONTAP System Manager, or the ONTAP CLI. When the replication is complete, the mirror state will be "snapmirrored".

We recommend using BlueXP to break the SnapMirror relationship. To break a SnapMirror relationship with BlueXP, follow the instructions outlined in the BlueXP replication documentation.

If you prefer not to use BlueXP, you can use ONTAP System Manager or the ONTAP CLI to break the SnapMirror relationship. To break a SnapMirror relationship with System Manager, refer to the instructions outlined in the ONTAP documentation. To break a SnapMirror relationship with the ONTAP CLI, refer to the instructions outlined in the ONTAP documentation.

If you wish to automate this process, you can use the BlueXP API, the ONTAP REST API, or the ONTAP Ansible collection.

Expose Destination Volume to Domino

Next, you must expose the destination volume to the Domino MLOps platform. To expose the destination volume to Domino, follow the instructions outlined in the 'Expose Existing NFS Volumes that were not Provisioned by Astra Trident' sub-section of the 'Expose Existing NetApp Volumes to Domino' section of this solution.

Now, you will be able to mount the destination volume when launching jobs and workspaces in the destination data plane as shown in the following screenshots.

Before Creating SnapMirror Relationship
After Exposing Destination Volume to Domino
Where to Find Additional Information

To learn more about the information described in this document, refer to the following documents and/or websites:

- Domino Data Lab
  https://domino.ai
- Domino Nexus
  https://domino.ai/platform/nexus
NVIDIA AI Enterprise with NetApp and VMware

NVIDIA AI Enterprise with NetApp and VMware

Mike Oglesby, NetApp

For IT architects and admins, AI tooling can be complicated and unfamiliar. Additionally, many AI platforms are not enterprise-ready. NVIDIA AI Enterprise, powered by NetApp and VMware, was created to deliver a streamlined, enterprise-class AI architecture.

NVIDIA AI Enterprise is an end-to-end, cloud-native suite of AI and data analytics software that is optimized, certified, and supported by NVIDIA to run on VMware vSphere with NVIDIA-Certified Systems. This software facilitates the simple and rapid deployment, management, and scaling of AI workloads in the modern hybrid cloud environment. NVIDIA AI Enterprise, powered by NetApp and VMware, delivers enterprise-class AI workload and data management in a simplified, familiar package.
Technology Overview

NVIDIA AI Enterprise

NVIDIA AI Enterprise is an end-to-end, cloud-native suite of AI and data analytics software that is optimized, certified, and supported by NVIDIA to run on VMware vSphere with NVIDIA-Certified Systems. This software facilitates the simple and rapid deployment, management, and scaling of AI workloads in the modern hybrid cloud environment.

NVIDIA GPU Cloud (NGC)

NVIDIA NGC hosts a catalog of GPU-optimized software for AI practitioners to develop their AI solutions. It also provides access to various AI services including NVIDIA Base Command for model training, NVIDIA Fleet Command to deploy and monitor models, and the NGC Private Registry for securely accessing and managing proprietary AI software. Also, NVIDIA AI Enterprise customers can request support through the NGC portal.

VMware vSphere

VMware vSphere is VMware’s virtualization platform, which transforms data centers into aggregated computing infrastructures that include CPU, storage, and networking resources. vSphere manages these infrastructures as a unified operating environment, and provides administrators with the tools to manage the data centers that participate in that environment.

The two core components of vSphere are ESXi and vCenter Server. ESXi is the virtualization platform where administrators create and run virtual machines and virtual appliances. vCenter Server is the service through which administrators manage multiple hosts connected in a network and pool host resources.
ONTAP 9, the latest generation of storage management software from NetApp, enables businesses to modernize infrastructure and transition to a cloud-ready data center. Leveraging industry-leading data management capabilities, ONTAP enables the management and protection of data with a single set of tools, regardless of where that data resides. You can also move data freely to wherever it is needed: the edge, the core, or the cloud. ONTAP 9 includes numerous features that simplify data management, accelerate, and protect critical data, and enable next generation infrastructure capabilities across hybrid cloud architectures.

Simplify data management

Data management is crucial to enterprise IT operations and data scientists so that appropriate resources are used for AI applications and training AI/ML datasets. The following additional information about NetApp technologies is out of scope for this validation but might be relevant depending on your deployment.

ONTAP data management software includes the following features to streamline and simplify operations and reduce your total cost of operation:

- Inline data compaction and expanded deduplication. Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity. This applies to data stored locally and data tiered to the cloud.
- Minimum, maximum, and adaptive quality of service (AQoS). Granular quality of service (QoS) controls help maintain performance levels for critical applications in highly shared environments.
- NetApp FabricPool. Provides automatic tiering of cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID storage solution. For more information about FabricPool, see TR-4598: FabricPool best practices.

Accelerate and protect data

ONTAP delivers superior levels of performance and data protection and extends these capabilities in the following ways:

- Performance and lower latency. ONTAP offers the highest possible throughput at the lowest possible latency.
- Data protection. ONTAP provides built-in data protection capabilities with common management across all platforms.
- NetApp Volume Encryption (NVE). ONTAP offers native volume-level encryption with both onboard and External Key Management support.
- Multitenancy and multifactor authentication. ONTAP enables sharing of infrastructure resources with the highest levels of security.

Future-proof infrastructure

ONTAP helps meet demanding and constantly changing business needs with the following features:

- Seamless scaling and nondisruptive operations. ONTAP supports the nondisruptive addition of capacity to existing controllers and to scale-out clusters. Customers can upgrade to the latest technologies, such as NVMe and 32Gb FC, without costly data migrations or outages.
- Cloud connection. ONTAP is the most cloud-connected storage management software, with options for software-defined storage (ONTAP Select) and cloud-native instances (NetApp Cloud Volumes Service) in all public clouds.
• Integration with emerging applications. ONTAP offers enterprise-grade data services for next generation platforms and applications, such as autonomous vehicles, smart cities, and Industry 4.0, by using the same infrastructure that supports existing enterprise apps.

**NetApp DataOps Toolkit**

The NetApp DataOps Toolkit is a Python-based tool that simplifies the management of development/training workspaces and inference servers that are backed by high-performance, scale-out NetApp storage. Key capabilities include:

• Rapidly provision new high-capacity JupyterLab workspaces that are backed by high-performance, scale-out NetApp storage.
• Rapidly provision new NVIDIA Triton Inference Server instances that are backed by enterprise-class NetApp storage.
• Near-instaneously clone high-capacity JupyterLab workspaces in order to enable experimentation or rapid iteration.
• Near-instaneously save snapshots of high-capacity JupyterLab workspaces for backup and/or traceability/baselining.
• Near-instaneously provision, clone, and snapshot high-capacity, high-performance data volumes.

**Architecture**

This solution builds upon a proven and familiar architecture featuring NetApp, VMware, and NVIDIA-Certified Systems. See the following table for details.

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI and Data Analytics Software</td>
<td>NVIDIA AI Enterprise for VMware</td>
</tr>
<tr>
<td>Virtualization Platform</td>
<td>VMware vSphere</td>
</tr>
<tr>
<td>Compute Platform</td>
<td>NVIDIA-Certified Systems</td>
</tr>
<tr>
<td>Data Management Platform</td>
<td>NetApp ONTAP</td>
</tr>
</tbody>
</table>
Initial Setup

This section describes the initial setup tasks that need to be performed in order to utilize NVIDIA AI Enterprise with NetApp and VMware.

Prerequisites

Before you perform the steps that are outlined in this section, we assume that you have already deployed VMware vSphere and NetApp ONTAP. Refer to the NVIDIA AI Enterprise Product Support Matrix for details on supported vSphere versions. Refer to the NetApp and VMware solution documentation for details on deploying VMware vSphere with NetApp ONTAP.
Install NVIDIA AI Enterprise Host Software

To install the NVIDIA AI Enterprise host software, follow the instructions outlined in sections 1-4 in the NVIDIA AI Enterprise Quick Start Guide.

Utilize NVIDIA NGC Software

This section describes the tasks that need to be performed in order to utilize NVIDIA NGC enterprise software within an NVIDIA AI Enterprise environment.

Setup

This section describes the initial setup tasks that need to be performed in order to utilize NVIDIA NGC enterprise software within an NVIDIA AI Enterprise environment.

Prerequisites

Before you perform the steps that are outlined in this section, we assume that you have already deployed the NVIDIA AI Enterprise host software by following the instructions outlined on the Initial Setup page.

Create an Ubuntu Guest VM with vGPU

First, you must create an Ubuntu 20.04 guest VM with vGPU. To create an Ubuntu 20.04 guest VM with vGPU, follow the instructions outlined in the NVIDIA AI Enterprise Deployment Guide.

Download and Install NVIDIA Guest Software

Next, you must install the required NVIDIA guest software within the guest VM that you created in the previous step. To download and install the required NVIDIA guest software within the guest VM, follow the instructions outlined in sections 5.1-5.4 in the NVIDIA AI Enterprise Quick Start Guide.

When performing the verification tasks outlined in section 5.4, you may need to use a different CUDA container image version tag as the CUDA container image has been updated since the writing of the guide. In our validation, we used 'nvidia/cuda:11.0.3-base-ubuntu20.04'.

Download AI/Analytics Framework Container(s)

Next, you must download needed AI or analytics framework container images from NVIDIA NGC so that they will be available within your guest VM. To download framework containers within the guest VM, follow the instructions outlined in the NVIDIA AI Enterprise Deployment Guide.

Install and Configure the NetApp DataOps Toolkit

Next, you must install the NetApp DataOps Toolkit for Traditional Environments within the guest VM. The NetApp DataOps Toolkit can be used to manage scale-out data volumes on your ONTAP system directly from the terminal within the guest VM. To install the NetApp DataOps Toolkit within the guest VM, perform the following tasks.

1. Install pip.
$ sudo apt update
$ sudo apt install python3-pip
$ python3 -m pip install netapp-dataops-traditional

2. Log out of the guest VM terminal and then log back in.
3. Configure the NetApp DataOps Toolkit. In order to complete this step, you will need API access details for your ONTAP system. You may need to obtain these from your storage admin.

```
$ netapp_dataops_cli.py config

Enter ONTAP management LIF hostname or IP address (Recommendation: Use SVM management interface): 172.22.10.10
Enter SVM (Storage VM) name: NVAIE-client
Enter SVM NFS data LIF hostname or IP address: 172.22.13.151
Enter default volume type to use when creating new volumes (flexgroup/flexvol) [flexgroup]:
Enter export policy to use by default when creating new volumes [default]:
Enter snapshot policy to use by default when creating new volumes [none]:
Enter unix filesystem user id (uid) to apply by default when creating new volumes (ex. '0' for root user) [0]:
Enter unix filesystem group id (gid) to apply by default when creating new volumes (ex. '0' for root group) [0]:
Enter unix filesystem permissions to apply by default when creating new volumes (ex. '0777' for full read/write permissions for all users and groups) [0777]:
Enter aggregate to use by default when creating new FlexVol volumes: aff_a400_01_NVME_SSD_1
Enter ONTAP API username (Recommendation: Use SVM account): admin
Enter ONTAP API password (Recommendation: Use SVM account):
Verify SSL certificate when calling ONTAP API (true/false): false
Do you intend to use this toolkit to trigger BlueXP Copy and Sync operations? (yes/no): no
Do you intend to use this toolkit to push/pull from S3? (yes/no): no
Created config file: '/home/user/.netapp_dataops/config.json'.
```

Create a Guest VM template

Lastly, you must create a VM template based on your guest VM. You will be able to use this template to quickly create guest VMs for utilizing NVIDIA NGC software.

To create a VM template based on your guest VM, log into VMware vSphere, right-click on the guest VM name, choose 'Clone', choose 'Clone to Template…', and then follow the wizard.
Example Use Case - TensorFlow Training Job

This section describes the tasks that need to be performed in order to execute a TensorFlow training job within an NVIDIA AI Enterprise environment.

Prerequisites

Before you perform the steps that are outlined in this section, we assume that you have already created a guest VM template by following the instructions outlined on the Setup page.

Create Guest VM from Template

First, you must create a new guest VM from the template that you created in the previous section. To create a new guest VM from your template, log into VMware vSphere, right-click on the template name, choose 'New VM from This Template...', and then follow the wizard.
Create and Mount Data Volume

Next, you must create a new data volume on which to store your training dataset. You can quickly create a new data volume using the NetApp DataOps Toolkit. The example command that follows shows the creation of a volume named 'imagenet' with a capacity of 2 TB.

$ netapp_dataops_cli.py create vol -n imagenet -s 2TB

Before you can populate your data volume with data, you must mount it within the guest VM. You can quickly mount a data volume using the NetApp DataOps Toolkit. The example command that follows shows the mounting of the volume that was created in the previous step.

$ sudo -E netapp_dataops_cli.py mount vol -n imagenet -m ~/imagenet

Populate Data Volume

After the new volume has been provisioned and mounted, the training dataset can be retrieved from the source location and placed on the new volume. This typically will involve pulling the data from an S3 or Hadoop data lake and sometimes will involve help from a data engineer.

Execute TensorFlow Training Job

Now, you are ready to execute your TensorFlow training job. To execute your TensorFlow training job, perform the following tasks.

1. Pull the NVIDIA NGC enterprise TensorFlow container image.

   $ sudo docker pull nvcr.io/nvaie/tensorflow-2-1:22.05-tf1-nvaie-2.1-py3

2. Launch an instance of the NVIDIA NGC enterprise TensorFlow container. Use the '-v' option to attach your data volume to the container.

   $ sudo docker run --gpus all -v ~/imagenet:/imagenet -it --rm nvcr.io/nvaie/tensorflow-2-1:22.05-tf1-nvaie-2.1-py3

3. Execute your TensorFlow training program within the container. The example command that follows shows the execution of an example ResNet-50 training program that is included in the container image.

   $ python ./nvidia-examples/cnn/resnet.py --layers 50 -b 64 -i 200 -u batch --precision fp16 --data_dir /imagenet/data

Where to Find Additional Information

To learn more about the information described in this document, refer to the following documents and/or websites:
TR-4851: NetApp StorageGRID data lake for autonomous driving workloads - Solution design

David Arnette, NetApp

TR-4851 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.

NetApp AI Control Plane

TR-4798: NetApp AI Control Plane

Mike Oglesby, NetApp

Companies and organizations of all sizes and across many industries are turning to artificial intelligence (AI), machine learning (ML), and deep learning (DL) to solve real-world problems, deliver innovative products and services, and to get an edge in an increasingly competitive marketplace. As organizations increase their use of AI, ML, and DL, they face many challenges, including workload scalability and data availability. This document demonstrates how you can address these challenges by using the NetApp AI Control Plane, a solution that pairs NetApp data management capabilities with popular open-source tools and frameworks.

This report shows you how to rapidly clone a data namespace. It also shows you how to seamlessly replicate data across sites and regions to create a cohesive and unified AI/ML/DL data pipeline. Additionally, it walks you through the defining and implementing of AI, ML, and DL training workflows that incorporate the near-instant
creation of data and model baselines for traceability and versioning. With this solution, you can trace every model training run back to the exact dataset that was used to train and/or validate the model. Lastly, this document shows you how to swiftly provision Jupyter Notebook workspaces with access to massive datasets.

Note: For HPC style distributed training at scale involving a large number of GPU servers that require shared access to the same dataset, or if you require/prefer a parallel file system, check out TR-4890. This technical report describes how to include NetApp’s fully supported parallel file system solution BeeGFS as part of the NetApp AI Control Plane. This solution is designed to scale from a handful of NVIDIA DGX A100 systems, up to a full blown 140 node SuperPOD.

The NetApp AI Control Plane is targeted towards data scientists and data engineers, and, thus, minimal NetApp or NetApp ONTAP® expertise is required. With this solution, data management functions can be executed using simple and familiar tools and interfaces. If you already have NetApp storage in your environment, you can test drive the NetApp AI Control plane today. If you want to test drive the solution but you do not have already have NetApp storage, visit cloud.netapp.com, and you can be up and running with a cloud-based NetApp storage solution in minutes. The following figure provides a visualization of the solution.
Concepts and Components

Artificial Intelligence

AI is a computer science discipline in which computers are trained to mimic the cognitive functions of the human mind. AI developers train computers to learn and to solve problems in a manner that is similar to, or even superior to, humans. Deep learning and machine learning are subfields of AI. Organizations are increasingly adopting AI, ML, and DL to support their critical business needs. Some examples are as follows:

- Analyzing large amounts of data to unearth previously unknown business insights
- Interacting directly with customers by using natural language processing
- Automating various business processes and functions

Modern AI training and inference workloads require massively parallel computing capabilities. Therefore, GPUs are increasingly being used to execute AI operations because the parallel processing capabilities of GPUs are vastly superior to those of general-purpose CPUs.

Containers

Containers are isolated user-space instances that run on top of a shared host operating system kernel. The adoption of containers is increasing rapidly. Containers offer many of the same application sandboxing benefits that virtual machines (VMs) offer. However, because the hypervisor and guest operating system layers that VMs rely on have been eliminated, containers are far more lightweight. The following figure depicts a visualization of virtual machines versus containers.

Containers also allow the efficient packaging of application dependencies, run times, and so on, directly with an application. The most commonly used container packaging format is the Docker container. An application that has been containerized in the Docker container format can be executed on any machine that can run Docker containers. This is true even if the application’s dependencies are not present on the machine because all dependencies are packaged in the container itself. For more information, visit the Docker website.

Kubernetes

Kubernetes is an open source, distributed, container orchestration platform that was originally designed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF). Kubernetes enables the
automation of deployment, management, and scaling functions for containerized applications. In recent years, Kubernetes has emerged as the dominant container orchestration platform. Although other container packaging formats and run times are supported, Kubernetes is most often used as an orchestration system for Docker containers. For more information, visit the Kubernetes website.

**NetApp Trident**

Trident is an open source storage orchestrator developed and maintained by NetApp that greatly simplifies the creation, management, and consumption of persistent storage for Kubernetes workloads. Trident, itself a Kubernetes-native application, runs directly within a Kubernetes cluster. With Trident, Kubernetes users (developers, data scientists, Kubernetes administrators, and so on) can create, manage, and interact with persistent storage volumes in the standard Kubernetes format that they are already familiar with. At the same time, they can take advantage of NetApp advanced data management capabilities and a data fabric that is powered by NetApp technology. Trident abstracts away the complexities of persistent storage and makes it simple to consume. For more information, visit the Trident website.

**NVIDIA DeepOps**

DeepOps is an open source project from NVIDIA that, by using Ansible, automates the deployment of GPU server clusters according to best practices. DeepOps is modular and can be used for various deployment tasks. For this document and the validation exercise that it describes, DeepOps is used to deploy a Kubernetes cluster that consists of GPU server worker nodes. For more information, visit the DeepOps website.

**Kubeflow**

Kubeflow is an open source AI and ML toolkit for Kubernetes that was originally developed by Google. The Kubeflow project makes deployments of AI and ML workflows on Kubernetes simple, portable, and scalable. Kubeflow abstracts away the intricacies of Kubernetes, allowing data scientists to focus on what they know best—data science. See the following figure for a visualization. Kubeflow has been gaining significant traction as enterprise IT departments have increasingly standardized on Kubernetes. For more information, visit the Kubeflow website.
Kubeflow Pipelines

Kubeflow Pipelines are a key component of Kubeflow. Kubeflow Pipelines are a platform and standard for defining and deploying portable and scalable AI and ML workflows. For more information, see the official Kubeflow documentation.

Jupyter Notebook Server

A Jupyter Notebook Server is an open source web application that allows data scientists to create wiki-like documents called Jupyter Notebooks that contain live code as well as descriptive text. Jupyter Notebooks are widely used in the AI and ML community as a means of documenting, storing, and sharing AI and ML projects. Kubeflow simplifies the provisioning and deployment of Jupyter Notebook Servers on Kubernetes. For more information on Jupyter Notebooks, visit the Jupyter website. For more information about Jupyter Notebooks within the context of Kubeflow, see the official Kubeflow documentation.

Apache Airflow

Apache Airflow is an open-source workflow management platform that enables programmatic authoring, scheduling, and monitoring for complex enterprise workflows. It is often used to automate ETL and data pipeline workflows, but it is not limited to these types of workflows. The Airflow project was started by Airbnb but has since become very popular in the industry and now falls under the auspices of The Apache Software Foundation. Airflow is written in Python, Airflow workflows are created via Python scripts, and Airflow is
designed under the principle of "configuration as code." Many enterprise Airflow users now run Airflow on top of Kubernetes.

**Directed Acyclic Graphs (DAGs)**

In Airflow, workflows are called Directed Acyclic Graphs (DAGs). DAGs are made up of tasks that are executed in sequence, in parallel, or a combination of the two, depending on the DAG definition. The Airflow scheduler executes individual tasks on an array of workers, adhering to the task-level dependencies that are specified in the DAG definition. DAGs are defined and created via Python scripts.

**NetApp ONTAP 9**

NetApp ONTAP 9 is the latest generation of storage management software from NetApp that enables businesses like yours to modernize infrastructure and to transition to a cloud-ready data center. With industry-leading data management capabilities, ONTAP enables you to manage and protect your data with a single set of tools regardless of where that data resides. You can also move data freely to wherever you need it: the edge, the core, or the cloud. ONTAP 9 includes numerous features that simplify data management, accelerate and protect your critical data, and future-proof your infrastructure across hybrid cloud architectures.

**Simplify Data Management**

Data management is crucial for your enterprise IT operations so that you can use appropriate resources for your applications and datasets. ONTAP includes the following features to streamline and simplify your operations and reduce your total cost of operation:

- **Inline data compaction and expanded deduplication.** Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity.

- **Minimum, maximum, and adaptive quality of service (QoS).** Granular QoS controls help maintain performance levels for critical applications in highly shared environments.

- **ONTAP FabricPool.** This feature provides automatic tiering of cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID object-based storage.

**Accelerate and Protect Data**

ONTAP delivers superior levels of performance and data protection and extends these capabilities with the following features:

- **High performance and low latency.** ONTAP offers the highest possible throughput at the lowest possible latency.

- **NetApp ONTAP FlexGroup technology.** A FlexGroup volume is a high-performance data container that can scale linearly to up to 20PB and 400 billion files, providing a single namespace that simplifies data management.

- **Data protection.** ONTAP provides built-in data protection capabilities with common management across all platforms.

- **NetApp Volume Encryption.** ONTAP offers native volume-level encryption with both onboard and external key management support.

**Future-Proof Infrastructure**

ONTAP 9 helps meet your demanding and constantly changing business needs:

- **Seamless scaling and nondisruptive operations.** ONTAP supports the nondisruptive addition of
capacity to existing controllers and to scale-out clusters. You can upgrade to the latest technologies, such as NVMe and 32Gb FC, without costly data migrations or outages.

- **Cloud connection.** ONTAP is one of the most cloud-connected storage management software, with options for software-defined storage (ONTAP Select) and cloud-native instances (NetApp Cloud Volumes Service) in all public clouds.

- **Integration with emerging applications.** By using the same infrastructure that supports existing enterprise apps, ONTAP offers enterprise-grade data services for next-generation platforms and applications such as OpenStack, Hadoop, and MongoDB.

**NetApp Snapshot Copies**

A NetApp Snapshot copy is a read-only, point-in-time image of a volume. The image consumes minimal storage space and incurs negligible performance overhead because it only records changes to files created since the last Snapshot copy was made, as depicted in the following figure.

Snapshot copies owe their efficiency to the core ONTAP storage virtualization technology, the Write Anywhere File Layout (WAFL). Like a database, WAFL uses metadata to point to actual data blocks on disk. But, unlike a database, WAFL does not overwrite existing blocks. It writes updated data to a new block and changes the metadata. It’s because ONTAP references metadata when it creates a Snapshot copy, rather than copying data blocks, that Snapshot copies are so efficient. Doing so eliminates the seek time that other systems incur in locating the blocks to copy, as well as the cost of making the copy itself.

You can use a Snapshot copy to recover individual files or LUNs or to restore the entire contents of a volume. ONTAP compares pointer information in the Snapshot copy with data on disk to reconstruct the missing or damaged object, without downtime or a significant performance cost.
NetApp FlexClone Technology

NetApp FlexClone technology references Snapshot metadata to create writable, point-in-time copies of a volume. Copies share data blocks with their parents, consuming no storage except what is required for metadata until changes are written to the copy, as depicted in the following figure. Where traditional copies can take minutes or even hours to create, FlexClone software lets you copy even the largest datasets almost instantaneously. That makes it ideal for situations in which you need multiple copies of identical datasets (a development workspace, for example) or temporary copies of a dataset (testing an application against a production dataset).

A Snapshot copy records only changes to the active file system since the last Snapshot copy.
NetApp SnapMirror Data Replication Technology

NetApp SnapMirror software is a cost-effective, easy-to-use unified replication solution across the data fabric. It replicates data at high speeds over LAN or WAN. It gives you high data availability and fast data replication for applications of all types, including business critical applications in both virtual and traditional environments. When you replicate data to one or more NetApp storage systems and continually update the secondary data, your data is kept current and is available whenever you need it. No external replication servers are required. See the following figure for an example of an architecture that leverages SnapMirror technology.

SnapMirror software leverages NetApp ONTAP storage efficiencies by sending only changed blocks over the network. SnapMirror software also uses built-in network compression to accelerate data transfers and reduce network bandwidth utilization by up to 70%. With SnapMirror technology, you can leverage one thin replication data stream to create a single repository that maintains both the active mirror and prior point-in-time copies, reducing network traffic by up to 50%.

*FlexClone copies share data blocks with their parents, consuming no storage except what is required for metadata.*
NetApp BlueXP Copy and Sync

BlueXP Copy and Sync is a NetApp service for rapid and secure data synchronization. Whether you need to transfer files between on-premises NFS or SMB file shares, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage, or IBM Cloud Object Storage, BlueXP Copy and Sync moves the files where you need them quickly and securely.

After your data is transferred, it is fully available for use on both source and target. BlueXP Copy and Sync can sync data on-demand when an update is triggered or continuously sync data based on a predefined schedule. Regardless, BlueXP Copy and Sync only moves the deltas, so time and money spent on data replication is minimized.

BlueXP Copy and Sync is a software as a service (SaaS) tool that is extremely simple to set up and use. Data transfers that are triggered by BlueXP Copy and Sync are carried out by data brokers. BlueXP Copy and Sync data brokers can be deployed in AWS, Azure, Google Cloud Platform, or on-premises.

NetApp XCP

NetApp XCP is client-based software for any-to-NetApp and NetApp-to-NetApp data migrations and file system insights. XCP is designed to scale and achieve maximum performance by utilizing all available system resources to handle high-volume datasets and high-performance migrations. XCP helps you to gain complete visibility into the file system with the option to generate reports.

NetApp XCP is available in a single package that supports NFS and SMB protocols. XCP includes a Linux binary for NFS data sets and a windows executable for SMB data sets.

NetApp XCP File Analytics is host-based software that detects file shares, runs scans on the file system, and provides a dashboard for file analytics. XCP File Analytics is compatible with both NetApp and non-NetApp systems and runs on Linux or Windows hosts to provide analytics for NFS and SMB-exported file systems.

NetApp ONTAP FlexGroup Volumes

A training dataset can be a collection of potentially billions of files. Files can include text, audio, video, and other forms of unstructured data that must be stored and processed to be read in parallel. The storage system
must store large numbers of small files and must read those files in parallel for sequential and random I/O.

A FlexGroup volume is a single namespace that comprises multiple constituent member volumes, as shown in the following figure. From a storage administrator viewpoint, a FlexGroup volume is managed and acts like a NetApp FlexVol volume. Files in a FlexGroup volume are allocated to individual member volumes and are not striped across volumes or nodes. They enable the following capabilities:

- FlexGroup volumes provide multiple petabytes of capacity and predictable low latency for high-metadata workloads.
- They support up to 400 billion files in the same namespace.
- They support parallelized operations in NAS workloads across CPUs, nodes, aggregates, and constituent FlexVol volumes.

![FlexGroup volume diagram]

Hardware and Software Requirements

The NetApp AI Control Plane solution is not dependent on this specific hardware. The solution is compatible with any NetApp physical storage appliance, software-defined instance, or cloud service, that is supported by Trident. Examples include a NetApp AFF storage system, Azure NetApp Files, NetApp Cloud Volumes Service, a NetApp ONTAP Select software-defined storage instance, or a NetApp Cloud Volumes ONTAP instance. Additionally, the solution can be implemented on any Kubernetes cluster as long as the Kubernetes version used is supported by Kubeflow and NetApp Trident. For a list of Kubernetes versions that are supported by Kubeflow, see the [official Kubeflow documentation](#). For a list of Kubernetes versions that are supported by Trident, see the [Trident documentation](#). See the following tables for details on the environment that was used to validate the solution.

<table>
<thead>
<tr>
<th>Infrastructure Component</th>
<th>Quantity</th>
<th>Details</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment jump host</td>
<td>1</td>
<td>VM</td>
<td>Ubuntu 20.04.2 LTS</td>
</tr>
<tr>
<td>Kubernetes master nodes</td>
<td>1</td>
<td>VM</td>
<td>Ubuntu 20.04.2 LTS</td>
</tr>
</tbody>
</table>
### Infrastructure Component

<table>
<thead>
<tr>
<th>Infrastructure Component</th>
<th>Quantity</th>
<th>Details</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes worker nodes</td>
<td>2</td>
<td>VM</td>
<td>Ubuntu 20.04.2 LTS</td>
</tr>
<tr>
<td>Kubernetes GPU worker nodes</td>
<td>2</td>
<td>NVIDIA DGX-1 (bare-metal)</td>
<td>NVIDIA DGX OS 4.0.5 (based on Ubuntu 18.04.2 LTS)</td>
</tr>
<tr>
<td>Storage</td>
<td>1 HA Pair</td>
<td>NetApp AFF A220</td>
<td>NetApp ONTAP 9.7 P6</td>
</tr>
</tbody>
</table>

### Software Component

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Airflow</td>
<td>2.0.1</td>
</tr>
<tr>
<td>Apache Airflow Helm Chart</td>
<td>8.0.8</td>
</tr>
<tr>
<td>Docker</td>
<td>19.03.12</td>
</tr>
<tr>
<td>Kubeflow</td>
<td>1.2</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>1.18.9</td>
</tr>
<tr>
<td>NetApp Trident</td>
<td>21.01.2</td>
</tr>
<tr>
<td>NVIDIA DeepOps</td>
<td>Trident deployment functionality from master branch as of commit 61898cfdfe; All other functionality from version 21.03</td>
</tr>
</tbody>
</table>

### Support

NetApp does not offer enterprise support for Apache Airflow, Docker, Kubeflow, Kubernetes, or NVIDIA DeepOps. If you are interested in a fully supported solution with capabilities similar to the NetApp AI Control Plane solution, contact NetApp about fully supported AI/ML solutions that NetApp offers jointly with partners.

### Kubernetes Deployment

This section describes the tasks that you must complete to deploy a Kubernetes cluster in which to implement the NetApp AI Control Plane solution. If you already have a Kubernetes cluster, then you can skip this section as long as you are running a version of Kubernetes that is supported by Kubeflow and NetApp Trident. For a list of Kubernetes versions that are supported by Kubeflow, see the official Kubeflow documentation. For a list of Kubernetes versions that are supported by Trident, see the Trident documentation.

For on-premises Kubernetes deployments that incorporate bare-metal nodes featuring NVIDIA GPU(s), NetApp recommends using NVIDIA’s DeepOps Kubernetes deployment tool. This section outlines the deployment of a Kubernetes cluster using DeepOps.

### Prerequisites

Before you perform the deployment exercise that is outlined in this section, we assume that you have already performed the following tasks:

1. You have already configured any bare-metal Kubernetes nodes (for example, an NVIDIA DGX system that is part of an ONTAP AI pod) according to standard configuration instructions.
2. You have installed a supported operating system on all Kubernetes master and worker nodes and on a deployment jump host. For a list of operating systems that are supported by DeepOps, see the DeepOps GitHub site.

**Use NVIDIA DeepOps to Install and Configure Kubernetes**

To deploy and configure your Kubernetes cluster with NVIDIA DeepOps, perform the following tasks from a deployment jump host:

1. Download NVIDIA DeepOps by following the instructions on the Getting Started page on the NVIDIA DeepOps GitHub site.
2. Deploy Kubernetes in your cluster by following the instructions on the Kubernetes Deployment Guide page on the NVIDIA DeepOps GitHub site.

**NetApp Trident Deployment and Configuration**

**Prerequisites**

Before you perform the deployment exercise that is outlined in this section, we assume that you have already performed the following tasks:

1. You already have a working Kubernetes cluster, and you are running a version of Kubernetes that is supported by Trident. For a list of supported versions, see the Trident documentation.
2. You already have a working NetApp storage appliance, software-defined instance, or cloud storage service, that is supported by Trident.

**Install Trident**

To install and configure NetApp Trident in your Kubernetes cluster, perform the following tasks from the deployment jump host:

1. Deploy Trident using one of the following methods:
   - If you used NVIDIA DeepOps to deploy your Kubernetes cluster, you can also use NVIDIA DeepOps to deploy Trident in your Kubernetes cluster. To deploy Trident with DeepOps, follow the Trident deployment instructions on the NVIDIA DeepOps GitHub site.
   - If you did not use NVIDIA DeepOps to deploy your Kubernetes cluster or if you simply prefer to deploy Trident manually, you can deploy Trident by following the deployment instructions in the Trident documentation. Be sure to create at least one Trident Backend and at least one Kubernetes StorageClass, for more information about how to configure Backends and StorageClasses see the linked subsections at NetApp Docs.

   If you are deploying the NetApp AI Control Plane solution on an ONTAP AI pod, see Example Trident Backends for ONTAP AI Deployments for some examples of different Trident Backends that you might want to create and Example Kubernetes StorageClasses for ONTAP AI Deployments for some examples of different Kubernetes StorageClasses that you might want to create.
Before you can use Trident to dynamically provision storage resources within your Kubernetes cluster, you must create one or more Trident Backends. The examples that follow represent different types of Backends that you might want to create if you are deploying the NetApp AI Control Plane solution on an ONTAP AI pod. For more information about Backends, see the Trident documentation.

1. NetApp recommends creating a FlexGroup-enabled Trident Backend for each data LIF (logical network interface that provides data access) that you want to use on your NetApp AFF system. This will allow you to balance volume mounts across LIFs.

The example commands that follow show the creation of two FlexGroup-enabled Trident Backends for two different data LIFs that are associated with the same ONTAP storage virtual machine (SVM). These Backends use the `ontap-nas-flexgroup` storage driver. ONTAP supports two main data volume types: FlexVol and FlexGroup. FlexVol volumes are size-limited (as of this writing, the maximum size depends on the specific deployment). FlexGroup volumes, on the other hand, can scale linearly to up to 20PB and 400 billion files, providing a single namespace that greatly simplifies data management. Therefore, FlexGroup volumes are optimal for AI and ML workloads that rely on large amounts of data.

If you are working with a small amount of data and want to use FlexVol volumes instead of FlexGroup volumes, you can create Trident Backends that use the `ontap-nas` storage driver instead of the `ontap-nas-flexgroup` storage driver.

```bash
$ cat << EOF > ./trident-backend-ontap-ai-flexgroups-iface1.json
{
   "version": 1,
   "storageDriverName": "ontap-nas-flexgroup",
   "backendName": "ontap-ai-flexgroups-iface1",
   "managementLIF": "10.61.218.100",
   "dataLIF": "192.168.11.11",
   "svm": "ontapai_nfs",
   "username": "admin",
   "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-ontap-ai-flexgroups-iface1.json -n trident
+----------------------------+---------------------+---------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    | STATE               | VOLUMES |
+----------------------------+---------------------+---------------------+--------+---------+
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbdddb-e0b8-40b7-b263-b6da6dec0bdd | online |       0 |
+----------------------------+---------------------+---------------------+--------+---------+
```
2. NetApp also recommends creating one or more FlexVol-enabled Trident Backends. If you use FlexGroup volumes for training dataset storage, you might want to use FlexVol volumes for storing results, output, debug information, and so on. If you want to use FlexVol volumes, you must create one or more FlexVol-enabled Trident Backends. The example commands that follow show the creation of a single FlexVol-enabled Trident Backend that uses a single data LIF.
Example Kubernetes StorageClasses for ONTAP AI Deployments

Before you can use Trident to dynamically provision storage resources within your Kubernetes cluster, you must create one or more Kubernetes StorageClasses. The examples that follow represent different types of StorageClasses that you might want to

```bash
$ cat << EOF > ./trident-backend-ontap-ai-flexvols.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-ai-flexvols",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.11.11",
    "svm": "ontapai_nfs",
    "username": "admin",
    "password": "ontapai"
}
EOF

$ tridentctl create backend -f ./trident-backend-ontap-ai-flexvols.json -n trident

+----------------------------+---------------------+---------+
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |                 UUID    |
| STATE  | VOLUMES |
+----------------------------+---------------------+---------+
| ontap-ai-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-a9c1-52a69657fabe | online |       0 |
+----------------------------+---------------------+---------+

$ tridentctl get backend -n trident

+----------------------------+---------------------+---------+
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |                 UUID    |
| STATE  | VOLUMES |
+----------------------------+---------------------+---------+
| ontap-ai-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-a9c1-52a69657fabe | online |       0 |
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-b263-b6da6dec0bdd | online |       0 |
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------+---------+
```
create if you are deploying the NetApp AI Control Plane solution on an ONTAP AI pod. For more information about StorageClasses, see the Trident documentation.

1. NetApp recommends creating a separate StorageClass for each FlexGroup-enabled Trident Backend that you created in the section Example Trident Backends for ONTAP AI Deployments, step 1. These granular StorageClasses enable you to add NFS mounts that correspond to specific LIFs (the LIFs that you specified when you created the Trident Backends) as a particular Backend that is specified in the StorageClass spec file. The example commands that follow show the creation of two StorageClasses that correspond to the two example Backends that were created in the section Example Trident Backends for ONTAP AI Deployments, step 1. For more information about StorageClasses, see the Trident documentation.

So that a persistent volume isn’t deleted when the corresponding PersistentVolumeClaim (PVC) is deleted, the following example uses a reclaimPolicy value of Retain. For more information about the reclaimPolicy field, see the official Kubernetes documentation.
2. NetApp also recommends creating a StorageClass that corresponds to the FlexVol-enabled Trident Backend that you created in the section Example Trident Backends for ONTAP AI Deployments, step 2. The example commands that follow show the creation of a single StorageClass for FlexVol volumes.

In the following example, a particular Backend is not specified in the StorageClass definition file because only one FlexVol-enabled Trident backend was created. When you use Kubernetes to administer volumes that use this StorageClass, Trident attempts to use any available backend that uses the \texttt{ontap-nas} driver.
3. NetApp also recommends creating a generic StorageClass for FlexGroup volumes. The following example commands show the creation of a single generic StorageClass for FlexGroup volumes.

Note that a particular backend is not specified in the StorageClass definition file. Therefore, when you use Kubernetes to administer volumes that use this StorageClass, Trident attempts to use any available backend that uses the `ontap-nas-flexgroup` driver.

```yaml
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain created
$ kubectl get storageclass
NAME                              PROVISIONER         AGE
ontap-ai-flexgroups-retain-iface1  netapp.io/trident   1m
ontap-ai-flexgroups-retain-iface2  netapp.io/trident   1m
ontap-ai-flexgroups-retain         netapp.io/trident   0m
```
Kubeflow Deployment

This section describes the tasks that you must complete to deploy Kubeflow in your Kubernetes cluster.

Prerequisites

Before you perform the deployment exercise that is outlined in this section, we assume that you have already performed the following tasks:

1. You already have a working Kubernetes cluster, and you are running a version of Kubernetes that is supported by Kubeflow. For a list of supported versions, see the official Kubeflow documentation.
2. You have already installed and configured NetApp Trident in your Kubernetes cluster as outlined in Trident Deployment and Configuration.

Set Default Kubernetes StorageClass

Before you deploy Kubeflow, you must designate a default StorageClass within your Kubernetes cluster. The Kubeflow deployment process attempts to provision new persistent volumes using the default StorageClass. If no StorageClass is designated as the default StorageClass, then the deployment fails. To designate a default StorageClass within your cluster, perform the following task from the deployment jump host. If you have already designated a default StorageClass within your cluster, then you can skip this step.

1. Designate one of your existing StorageClasses as the default StorageClass. The example commands that follow show the designation of a StorageClass named `ontap-ai-flexvols-retain` as the default StorageClass.

   ```
   $ kubectl get sc
   NAME                                PROVISIONER             AGE
   ontpem-ai-flexgroups-retain          csi.trident.netapp.io   25h
   ontpem-ai-flexgroups-retain-iface1   csi.trident.netapp.io   25h
   ontpem-ai-flexgroups-retain-iface2   csi.trident.netapp.io   25h
   ontpem-ai-flexvols-retain           csi.trident.netapp.io   3s
   $ kubectl patch storageclass ontpem-ai-flexvols-retain -p '{"metadata":
   "annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}'
   storageclass.storage.k8s.io/ontap-ai-flexvols-retain patched
   $ kubectl get sc
   NAME                                PROVISIONER             AGE
   ontpem-ai-flexgroups-retain          csi.trident.netapp.io   25h
   ontpem-ai-flexgroups-retain-iface1   csi.trident.netapp.io   25h
   ontpem-ai-flexgroups-retain-iface2   csi.trident.netapp.io   25h
   ontpem-ai-flexvols-retain (default)  csi.trident.netapp.io   54s
   ```

   The `ontap-nas-flexgroup` Trident Backend type has a minimum PVC size that is fairly large. By default, Kubeflow attempts to provision PVCs that are only a few GBs in size. Therefore, you should not designate a StorageClass that utilizes the `ontap-nas-flexgroup` Backend type as the default StorageClass for the purposes of Kubeflow deployment.
Use NVIDIA DeepOps to Deploy Kubflow

NetApp recommends using the Kubflow deployment tool that is provided by NVIDIA DeepOps. To deploy Kubflow in your Kubernetes cluster using the DeepOps deployment tool, perform the following tasks from the deployment jump host.

Alternatively, you can deploy Kubflow manually by following the installation instructions in the official Kubflow documentation

1. Deploy Kubflow in your cluster by following the Kubflow deployment instructions on the NVIDIA DeepOps GitHub site.

2. Note down the Kubflow Dashboard URL that the DeepOps Kubflow deployment tool outputs.

   $ ./scripts/k8s/deploy_kubeflow.sh -x
   ...
   INFO[0007] Applied the configuration Successfully!
   filename="cmd/apply.go:72"
   Kubeflow app installed to: /home/ai/kubeflow
   It may take several minutes for all services to start. Run 'kubectl get pods -n kubeflow' to verify
   To remove (excluding CRDs, istio, auth, and cert-manager), run:
   $./scripts/k8s_deploy_kubeflow.sh -d
   To perform a full uninstall : .scripts/k8s_deploy_kubeflow.sh -D
   Kubeflow Dashboard (HTTP NodePort): http://10.61.188.111:31380

3. Confirm that all pods deployed within the Kubflow namespace show a STATUS of Running and confirm that no components deployed within the namespace are in an error state. It may take several minutes for all pods to start.

   $ kubectl get all -n kubeflow
   NAME                                                                 READY
   STATUS  RESTARTS  AGE
   pod/admission-webhook-bootstrap-stateful-set-0  1/1
   Running  0  95s
   pod/admission-webhook-deployment-6b89c84c98-vrtbh  1/1
   Running  0  91s
   pod/application-controller-stateful-set-0  1/1
   Running  0  98s
   pod/argo-ui-5dce5d8b4f-m2wn4  1/1
   Running  0  97s
   pod/centraldashboard-cf4874ddc-7hcr8  1/1
   Running  0  97s
   pod/jupyter-web-app-deployment-685b455447-gjhh7  1/1
   Running  0  96s
   pod/katib-controller-88c97d85c-kgq66  1/1
   Running  1  95s
<table>
<thead>
<tr>
<th>Pod Name</th>
<th>Replicas</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>pod/katib-db-8598468fd8-5jw2c</td>
<td>1/1</td>
<td>95s</td>
</tr>
<tr>
<td>pod/katib-manager-574c8c67f9-wtrf5</td>
<td>1/1</td>
<td>95s</td>
</tr>
<tr>
<td>pod/katib-manager-rest-778857c989-fjbzn</td>
<td>1/1</td>
<td>95s</td>
</tr>
<tr>
<td>pod/katib-suggestion-bayesianoptimization-65df4d7455-qthmw</td>
<td>1/1</td>
<td>94s</td>
</tr>
<tr>
<td>pod/katib-suggestion-grid-56bf69f597-98vwn</td>
<td>1/1</td>
<td>94s</td>
</tr>
<tr>
<td>pod/katib-suggestion-hyperband-7777b76cb9-9v6dq</td>
<td>1/1</td>
<td>93s</td>
</tr>
<tr>
<td>pod/katib-suggestion-nasrl-77f6f9458c-2qzxq</td>
<td>1/1</td>
<td>93s</td>
</tr>
<tr>
<td>pod/katib-suggestion-random-77b88b5c79-164j9</td>
<td>1/1</td>
<td>93s</td>
</tr>
<tr>
<td>pod/katib-ui-7587c5b967-nd629</td>
<td>1/1</td>
<td>95s</td>
</tr>
<tr>
<td>pod/metacontroller-0</td>
<td>1/1</td>
<td>96s</td>
</tr>
<tr>
<td>pod/metadata-db-5dd459cc-swzkm</td>
<td>1/1</td>
<td>94s</td>
</tr>
<tr>
<td>pod/metadata-deployment-6cf77db994-69fk7</td>
<td>1/1</td>
<td>93s</td>
</tr>
<tr>
<td>pod/metadata-deployment-6cf77db994-mpbjt</td>
<td>1/1</td>
<td>93s</td>
</tr>
<tr>
<td>pod/metadata-deployment-6cf77db994-xg7tz</td>
<td>1/1</td>
<td>94s</td>
</tr>
<tr>
<td>pod/metadata-ui-78f5b59b56-qb6kr</td>
<td>1/1</td>
<td>94s</td>
</tr>
<tr>
<td>pod/minio-758b769d67-1lvdr</td>
<td>1/1</td>
<td>91s</td>
</tr>
<tr>
<td>pod/ml-pipeline-5875b9db95-g8t2k</td>
<td>1/1</td>
<td>91s</td>
</tr>
<tr>
<td>pod/ml-pipeline-persistenceagent-9b69ddd46-bt9r9</td>
<td>1/1</td>
<td>90s</td>
</tr>
<tr>
<td>pod/ml-pipeline-scheduledworkflow-7b8d756c76-7x56s</td>
<td>1/1</td>
<td>90s</td>
</tr>
<tr>
<td>pod/ml-pipeline-ui-79ff9c76-fcpwd</td>
<td>1/1</td>
<td>90s</td>
</tr>
<tr>
<td>pod/ml-pipeline-viewer-controller-deployment-5fdc87f58-b2t9r</td>
<td>1/1</td>
<td>90s</td>
</tr>
<tr>
<td>pod/mysql-657f87857d-15k9z</td>
<td>1/1</td>
<td>91s</td>
</tr>
<tr>
<td>pod/notebook-controller-deployment-56b4f59bbf-8bvnr</td>
<td>1/1</td>
<td>92s</td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td>pod/profiles-deployment-6bc745947-mrdkh</td>
<td>Running 0 90s</td>
<td></td>
</tr>
<tr>
<td>pod/pytorch-operator-77c97f4879-hmlrv</td>
<td>Running 0 92s</td>
<td></td>
</tr>
<tr>
<td>pod/seldon-operator-controller-manager-0</td>
<td>Running 1 91s</td>
<td></td>
</tr>
<tr>
<td>pod/spartakus-volunteer-5fdfd2bb779-17qkm</td>
<td>Running 0 92s</td>
<td></td>
</tr>
<tr>
<td>pod/tensorboard-6544748d94-nh8b2</td>
<td>Running 0 92s</td>
<td></td>
</tr>
<tr>
<td>pod/tf-job-dashboard-56f79c59dd-6w59t</td>
<td>Running 0 92s</td>
<td></td>
</tr>
<tr>
<td>pod/tf-job-operator-79cbfd6dbc-rb58c</td>
<td>Running 0 91s</td>
<td></td>
</tr>
<tr>
<td>pod/workflow-controller-db644d554-cwrb</td>
<td>Running 0 97s</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>CLUSTER-IP</td>
<td>EXTERNAL-IP</td>
</tr>
<tr>
<td>service/admission-webhook-service</td>
<td>10.233.51.169</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/application-controller-service</td>
<td>10.233.4.54</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/argo-ui</td>
<td>10.233.47.191</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/centraldashboard</td>
<td>10.233.8.36</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/jupyter-web-app-service</td>
<td>10.233.1.42</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-controller</td>
<td>10.233.25.226</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-db</td>
<td>10.233.33.151</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-manager</td>
<td>10.233.46.239</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-manager-rest</td>
<td>10.233.55.32</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-suggestion-bayesianoptimization</td>
<td>10.233.49.191</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-suggestion-grid</td>
<td>10.233.9.105</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-suggestion-hyperband</td>
<td>10.233.22.2</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-suggestion-nasrl</td>
<td>10.233.63.73</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>service/katib-suggestion-random</td>
<td>10.233.57.210</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>NAME</td>
<td>READY</td>
<td>UP-TO-DATE</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>deployment.apps/admission-webhook-deployment</td>
<td>1/1</td>
<td>1</td>
</tr>
<tr>
<td>deployment.apps/argo-ui</td>
<td>1/1</td>
<td>1</td>
</tr>
<tr>
<td>deployment.apps/centraldashboard</td>
<td>1/1</td>
<td>1</td>
</tr>
<tr>
<td>deployment.apps/jupyter-web-app-deployment</td>
<td>1/1</td>
<td>1</td>
</tr>
<tr>
<td>deployment.apps/katib-controller</td>
<td>1/1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service/katib-ui</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.6.116 &lt;none&gt; 80/TCP 96s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/metadata-db</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.31.2 &lt;none&gt; 3306/TCP 96s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/metadata-service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.27.104 &lt;none&gt; 8080/TCP 96s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/metadata-ui</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.57.177 &lt;none&gt; 80/TCP 96s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/minio-service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.44.90 &lt;none&gt; 9000/TCP 94s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/ml-pipeline</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.41.201 &lt;none&gt; 8888/TCP,8887/TCP 94s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/ml-pipeline-tensorboard-ui</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.36.207 &lt;none&gt; 80/TCP 93s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/ml-pipeline-ui</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.61.150 &lt;none&gt; 80/TCP 93s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/mysql</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.55.117 &lt;none&gt; 3306/TCP 94s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/notebook-controller-service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.10.166 &lt;none&gt; 443/TCP 95s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/profiles-kfam</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.33.79 &lt;none&gt; 8081/TCP 92s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/pytorch-operator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.37.112 &lt;none&gt; 8443/TCP 95s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/seldon-operator-controller-manager-service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.30.178 &lt;none&gt; 443/TCP 92s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/tensorboard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.58.151 &lt;none&gt; 9000/TCP 94s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/tf-job-dashboard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.4.17 &lt;none&gt; 80/TCP 94s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/tf-job-operator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.60.32 &lt;none&gt; 8443/TCP 94s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service/webhook-server-service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.233.32.167 &lt;none&gt; 443/TCP 87s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td>Replicas</td>
<td>Age</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-db</td>
<td>1/1</td>
<td>97s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-manager</td>
<td>1/1</td>
<td>96s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-manager-rest</td>
<td>1/1</td>
<td>96s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-suggestion-bayesianoptimization</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-suggestion-grid</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-suggestion-hyperband</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-suggestion-nasrl</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-suggestion-random</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/katib-ui</td>
<td>1/1</td>
<td>96s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/metadata-db</td>
<td>1/1</td>
<td>96s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/metadata-deployment</td>
<td>3/3</td>
<td>96s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/metadata-ui</td>
<td>1/1</td>
<td>96s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/minio</td>
<td>1/1</td>
<td>94s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/ml-pipeline</td>
<td>1/1</td>
<td>94s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/ml-pipeline-persistenceagent</td>
<td>1/1</td>
<td>93s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/ml-pipeline-scheduledworkflow</td>
<td>1/1</td>
<td>93s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/ml-pipeline-ui</td>
<td>1/1</td>
<td>93s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/ml-pipeline-viewer-controller-deployment</td>
<td>1/1</td>
<td>93s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/mysql</td>
<td>1/1</td>
<td>94s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/notebook-controller-deployment</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/profiles-deployment</td>
<td>1/1</td>
<td>92s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/pytorch-operator</td>
<td>1/1</td>
<td>95s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deployment.apps/spartakus-volunteer</td>
<td>1/1</td>
<td>94s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>DESIRED</td>
<td>CURRENT</td>
<td>READY</td>
<td>AGE</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>deployment.apps/tensorboard</td>
<td>1/1</td>
<td>1</td>
<td></td>
<td>94s</td>
</tr>
<tr>
<td>deployment.apps/tf-job-dashboard</td>
<td>1/1</td>
<td>1</td>
<td></td>
<td>94s</td>
</tr>
<tr>
<td>deployment.apps/tf-job-operator</td>
<td>1/1</td>
<td>1</td>
<td></td>
<td>94s</td>
</tr>
<tr>
<td>deployment.apps/workflow-controller</td>
<td>1/1</td>
<td>1</td>
<td></td>
<td>97s</td>
</tr>
<tr>
<td>NAME</td>
<td>DESIRED</td>
<td>CURRENT</td>
<td>READY</td>
<td>AGE</td>
</tr>
<tr>
<td>replicaset.apps/admission-webhook-deployment-6b89c84c98</td>
<td>1</td>
<td>1</td>
<td>97s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/argo-ui-5dcf5d8b4f</td>
<td>1</td>
<td>1</td>
<td>97s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/centraldashboard-cf4874ddc</td>
<td>1</td>
<td>1</td>
<td>97s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/jupyter-web-app-deployment-685b455447</td>
<td>1</td>
<td>1</td>
<td>97s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-controller-88c97d85c</td>
<td>1</td>
<td>1</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-db-8598468fd8</td>
<td>1</td>
<td>1</td>
<td>97s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-manager-574c8c67f9</td>
<td>1</td>
<td>1</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-manager-rest-778857c989</td>
<td>1</td>
<td>1</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-suggestion-bayesianoptimization-65df4d7455</td>
<td>1</td>
<td>1</td>
<td>95s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-suggestion-grid-56bf69f597</td>
<td>1</td>
<td>1</td>
<td>95s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-suggestion-hyperband-7777b76cb9</td>
<td>1</td>
<td>1</td>
<td>95s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-suggestion-nasrl-77f6f9458c</td>
<td>1</td>
<td>1</td>
<td>95s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-suggestion-random-77b88b5c79</td>
<td>1</td>
<td>1</td>
<td>95s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/katib-ui-7587c5b967</td>
<td>1</td>
<td>1</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/metadata-db-5dd459cc</td>
<td>1</td>
<td>1</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/metadata-deployment-6cf77db994</td>
<td>3</td>
<td>3</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/metadata-ui-78f5b59b56</td>
<td>1</td>
<td>1</td>
<td>96s</td>
<td></td>
</tr>
<tr>
<td>replicaset.apps/minio-758b769d67</td>
<td>1</td>
<td>1</td>
<td>93s</td>
<td></td>
</tr>
</tbody>
</table>
replicaset.apps/ml-pipeline-5875b9db95 1
1 1 93s
replicaset.apps/ml-pipeline-persistenceagent-9b69ddd46 1
1 1 92s
replicaset.apps/ml-pipeline-scheduledworkflow-7b8d756c76 1
1 1 91s
replicaset.apps/ml-pipeline-ui-79ffd9c76 1
1 1 91s
replicaset.apps/ml-pipeline-viewer-controller-deployment-5fdc87f58 1
1 1 91s
replicaset.apps/mysql-657f87857d 1
1 1 92s
replicaset.apps/notebook-controller-deployment-56b4f59bbf 1
1 1 94s
replicaset.apps/profiles-deployment-6bc745947 1
1 1 91s
replicaset.apps/pytorch-operator-77c97f4879 1
1 1 94s
replicaset.apps/spartakus-volunteer-5fdffdb779 1
1 1 94s
replicaset.apps/tensorboard-6544748d94 1
1 1 93s
replicaset.apps/tf-job-dashboard-56f79c59dd 1
1 1 93s
replicaset.apps/tf-job-operator-79cbfd6dbc 1
1 1 93s
replicaset.apps/workflow-controller-db644d554 1
1 1 97s

NAME                                                        READY   AGE
statefulset.apps/admission-webhook-bootstrap-stateful-set   1/1     97s
statefulset.apps/application-controller-stateful-set        1/1     98s
statefulset.apps/metacontroller                             1/1     98s
statefulset.apps/seldon-operator-controller-manager         1/1     98s

$ kubectl get pvc -n kubeflow

NAME             STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS               AGE
katib-mysql      Bound    pvc-b07f293e-d028-11e9-9b9d-00505681a82d
10Gi       RWO            ontap-ai-flexvols-retain   27m
metadata-mysql   Bound    pvc-b0f3f032-d028-11e9-9b9d-00505681a82d
10Gi       RWO            ontap-ai-flexvols-retain   27m
minio-pv-claim   Bound    pvc-b22727ee-d028-11e9-9b9d-00505681a82d
20Gi       RWO            ontap-ai-flexvols-retain   27m
mysql-pv-claim   Bound    pvc-b2429afd-d028-11e9-9b9d-00505681a82d
20Gi       RWO            ontap-ai-flexvols-retain   27m

4. In your web browser, access the Kubeflow central dashboard by navigating to the URL that you noted
down in step 2.

The default username is admin@kubeflow.org, and the default password is 12341234. To create additional users, follow the instructions in the official Kubeflow documentation.

Example Kubeflow Operations and Tasks

This section includes examples of various operations and tasks that you may want to perform using Kubeflow.

Provision a Jupyter Notebook Workspace for Data Scientist or Developer Use

Kubeflow is capable of rapidly provisioning new Jupyter Notebook servers to act as data scientist workspaces. To provision a new Jupyter Notebook server with Kubeflow, perform the following tasks. For more information about Jupyter Notebooks within the Kubeflow
context, see the official Kubeflow documentation.

1. From the Kubeflow central dashboard, click Notebook Servers in the main menu to navigate to the Jupyter Notebook server administration page.

2. Click New Server to provision a new Jupyter Notebook server.
3. Give your new server a name, choose the Docker image that you want your server to be based on, and specify the amount of CPU and RAM to be reserved by your server. If the Namespace field is blank, use the Select Namespace menu in the page header to choose a namespace. The Namespace field is then auto-populated with the chosen namespace.

In the following example, the `kubeflow-anonymous` namespace is chosen. In addition, the default values for Docker image, CPU, and RAM are accepted.
4. Specify the workspace volume details. If you choose to create a new volume, then that volume or PVC is provisioned using the default StorageClass. Because a StorageClass utilizing Trident was designated as the default StorageClass in the section Kubeflow Deployment, the volume or PVC is provisioned with Trident. This volume is automatically mounted as the default workspace within the Jupyter Notebook Server container. Any notebooks that a user creates on the server that are not saved to a separate data volume are automatically saved to this workspace volume. Therefore, the notebooks are persistent across reboots.

5. Add data volumes. The following example specifies an existing PVC named 'pb-fg-all' and accepts the default mount point.
6. **Optional**: Request that the desired number of GPUs be allocated to your notebook server. In the following example, one GPU is requested.

7. Click Launch to provision your new notebook server.

8. Wait for your notebook server to be fully provisioned. This can take several minutes if you have never provisioned a server using the Docker image that you specified because the image needs to be downloaded. When your server has been fully provisioned, you see a green check mark in the Status column on the Jupyter Notebook server administration page.
9. Click Connect to connect to your new server web interface.

10. Confirm that the dataset volume that was specified in step 6 is mounted on the server. Note that this volume is mounted within the default workspace by default. From the perspective of the user, this is just another folder within the workspace. The user, who is likely a data scientist and not an infrastructure expert, does not need to possess any storage expertise in order to use this volume.
11. Open a Terminal and, assuming that a new volume was requested in step 5, execute `df -h` to confirm that a new Trident-provisioned persistent volume is mounted as the default workspace.

The default workspace directory is the base directory that you are presented with when you first access the server’s web interface. Therefore, any artifacts that you create by using the web interface are stored on this Trident-provisioned persistent volume.
12. Using the terminal, run `nvidia-smi` to confirm that the correct number of GPUs were allocated to the notebook server. In the following example, one GPU has been allocated to the notebook server as requested in step 7.
Example Notebooks and Pipelines

The **NetApp Data Science Toolkit for Kubernetes** can be used in conjunction with Kubeflow. Using the NetApp Data Science Toolkit with Kubeflow provides the following benefits:

- Data scientists can perform advanced NetApp data management operations directly from within a Jupyter Notebook.
- Advanced NetApp data management operations can be incorporated into automated workflows using the Kubeflow Pipelines framework.

Refer to the **Kubeflow Examples** section within the NetApp Data Science Toolkit GitHub repository for details on using the toolkit with Kubeflow.

### Apache Airflow Deployment

NetApp recommends running Apache Airflow on top of Kubernetes. This section describes the tasks that you must complete to deploy Airflow in your Kubernetes cluster.

| ![i] It is possible to deploy Airflow on platforms other than Kubernetes. Deploying Airflow on platforms other than Kubernetes is outside of the scope of this solution. |

**Prerequisites**

Before you perform the deployment exercise that is outlined in this section, we assume that you have already performed the following tasks:

1. You already have a working Kubernetes cluster.
2. You have already installed and configured NetApp Trident in your Kubernetes cluster as outlined in the section “NetApp Trident Deployment and Configuration.”

**Install Helm**

Airflow is deployed using Helm, a popular package manager for Kubernetes. Before you deploy Airflow, you must install Helm on the deployment jump host. To install Helm on the deployment jump host, follow the [installation instructions](#) in the official Helm documentation.

**Set Default Kubernetes StorageClass**

Before you deploy Airflow, you must designate a default StorageClass within your Kubernetes cluster. The Airflow deployment process attempts to provision new persistent volumes using the default StorageClass. If no StorageClass is designated as the default StorageClass, then the deployment fails. To designate a default StorageClass within your cluster, follow the instructions outlined in the section **Kubeflow Deployment**. If you have already designated a default StorageClass within your cluster, then you can skip this step.

**Use Helm to Deploy Airflow**

To deploy Airflow in your Kubernetes cluster using Helm, perform the following tasks from the deployment jump host:

1. Deploy Airflow using Helm by following the [deployment instructions](#) for the official Airflow chart on the Artifact Hub. The example commands that follow show the deployment of Airflow using Helm. Modify, add, and/or remove values in the `custom-values.yaml` file as needed depending on your environment and
desired configuration.

```bash
$ cat << EOF > custom-values.yaml
###################################
# Airflow - Common Configs
###################################
airflow:
  ## the airflow executor type to use
  executor: "CeleryExecutor"
  ## environment variables for the web/scheduler/worker Pods (for airflow configs)

###################################
# Airflow - WebUI Configs
###################################
web:
  ## configs for the Service of the web Pods
  service:
    type: NodePort

###################################
# Airflow - Logs Configs
###################################
logs:
  persistence:
    enabled: true

###################################
# Airflow - DAGs Configs
###################################
dags:
  ## configs for the DAG git repository & sync container
  gitSync:
    enabled: true
    ## url of the git repository
    repo: "git@github.com:mboglesby/airflow-dev.git"
    ## the branch/tag/sha1 which we clone
    branch: master
    revision: HEAD
    ## the name of a pre-created secret containing files for ~/.ssh/
    ## NOTE:
```
## - this is ONLY RELEVANT for SSH git repos
## - the secret commonly includes files: id_rsa, id_rsa.pub, known_hosts
## - known_hosts is NOT NEEDED if `git.sshKeyscan` is true
## sshSecret: "airflow-ssh-git-secret"
## the name of the private key file in your `git.secret`
##
## NOTE:
## - this is ONLY RELEVANT for PRIVATE SSH git repos
## sshSecretKey: id_rsa
## the git sync interval in seconds
##
## syncWait: 60
EOF

$ helm install airflow airflow-stable/airflow -n airflow --version 8.0.8 --values ./custom-values.yaml
...

Congratulations. You have just deployed Apache Airflow!

1. Get the Airflow Service URL by running these commands:
   ```
   export NODE_PORT=$(kubectl get --namespace airflow -o jsonpath='{.spec.ports[0].nodePort}' services airflow-web)
   export NODE_IP=$(kubectl get nodes --namespace airflow -o jsonpath='{.items[0].status.addresses[0].address}')
   echo http://$NODE_IP:$NODE_PORT/
   ```
   2. Open Airflow in your web browser

2. Confirm that all Airflow pods are up and running. It may take a few minutes for all pods to start.

   ```
   $ kubectl -n airflow get pod
   NAME                                READY   STATUS    RESTARTS   AGE
   airflow-flower-b5656d44f-h8qjk      1/1     Running   0          2h
   airflow-postgresql-0                1/1     Running   0          2h
   airflow-redis-master-0              1/1     Running   0          2h
   airflow-scheduler-9d95fcd9-clf4b     2/2     Running   2          2h
   airflow-web-59c94db9c5-z7rg4         1/1     Running   0          2h
   airflow-worker-0                    2/2     Running   2          2h
   ```

3. Obtain the Airflow web service URL by following the instructions that were printed to the console when you deployed Airflow using Helm in step 1.

   ```
   3. Obtain the Airflow web service URL by following the instructions that were printed to the console when you deployed Airflow using Helm in step 1.
   ```
4. Confirm that you can access the Airflow web service.

$ export NODE_PORT=$(kubectl get --namespace airflow -o jsonpath='{.spec.ports[0].nodePort}' services airflow-web)
$ export NODE_IP=$(kubectl get nodes --namespace airflow -o jsonpath='{.items[0].status.addresses[0].address}')
$ echo http://$NODE_IP:$NODE_PORT/

Example Apache Airflow Workflows

The NetApp Data Science Toolkit for Kubernetes can be used in conjunction with Airflow. Using the NetApp Data Science Toolkit with Airflow enables you to incorporate NetApp data management operations into automated workflows that are orchestrated by Airflow.

Refer to the Airflow Examples section within the NetApp Data Science Toolkit GitHub repository for details on using the toolkit with Airflow.
Example Trident Operations

This section includes examples of various operations that you may want to perform with Trident.

Import an Existing Volume

If there are existing volumes on your NetApp storage system/platform that you want to mount on containers within your Kubernetes cluster, but that are not tied to PVCs in the cluster, then you must import these volumes. You can use the Trident volume import functionality to import these volumes.

The example commands that follow show the importing of the same volume, named `pb_fg_all`, twice, once for each Trident Backend that was created in the example in the section Example Trident Backends for ONTAP AI Deployments, step 1. Importing the same volume twice in this manner enables you to mount the volume (an existing FlexGroup volume) multiple times across different LIFs, as described in the section Example Trident Backends for ONTAP AI Deployments, step 1. For more information about PVCs, see the official Kubernetes documentation. For more information about the volume import functionality, see the Trident documentation.

An `accessModes` value of `ReadOnlyMany` is specified in the example PVC spec files. For more information about the `accessMode` field, see the official Kubernetes documentation.

```
$ cat << EOF > ./pvc-import-pb_fg_all-iface1.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pb-fg-all-iface1
  namespace: default
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ontap-ai-flexgroups-retain-iface1
EOF

$ tridentctl import volume ontap-ai-flexgroups-iface1 pb_fg_all -f ./pvc-import-pb_fg_all-iface1.yaml -n trident
```

The Backend names that are specified in the following example import commands correspond to the Backends that were created in the example in the section Example Trident Backends for ONTAP AI Deployments, step 1. The StorageClass names that are specified in the following example PVC definition files correspond to the StorageClasses that were created in the example in the section Example Kubernetes StorageClasses for ONTAP AI Deployments, step 1.
```
|          NAME                  |  SIZE  |       STORAGE CLASS |
| PROTOCOL |             BACKEND UUID                         | STATE  |
| MANAGED  |                |                  |
| default-pb-fg-all-iface2-85aee | 10 TiB | ontap-ai-flexgroups-retain-iface2 | online | true |
```

```
$ tridentctl get volume -n trident
```

```
|          NAME                  |  SIZE  |       STORAGE CLASS |
| PROTOCOL |             BACKEND UUID                         | STATE  |
| MANAGED  |                |                  |
| default-pb-fg-all-iface1-7d9f1 | 10 TiB | ontap-ai-flexgroups-retain-iface1 | online | true |
```

```
Provision a New Volume

You can use Trident to provision a new volume on your NetApp storage system or platform. The following example commands show the provisioning of a new FlexVol volume. In this example, the volume is provisioned using the StorageClass that was created in the example in the section Example Kubernetes StorageClasses for ONTAP AI Deployments, step 2.

An `accessModes` value of `ReadWriteMany` is specified in the following example PVC definition file. For more information about the `accessMode` field, see the official Kubernetes documentation.
$ cat << EOF > ./pvc-tensorflow-results.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: tensorflow-results
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-ai-flexvols-retain
EOF
$ kubectl create -f ./pvc-tensorflow-results.yaml
persistentvolumeclaim/tensorflow-results created
$ kubectl get pvc

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>VOLUME</th>
</tr>
</thead>
<tbody>
<tr>
<td>pb-fg-all-iface1</td>
<td>Bound</td>
<td>default-pb-fg-all-iface1-7d9f1</td>
</tr>
<tr>
<td>pb-fg-all-iface2</td>
<td>Bound</td>
<td>default-pb-fg-all-iface2-85aee</td>
</tr>
<tr>
<td>tensorflow-results</td>
<td>Bound</td>
<td>default-tensorflow-results-2fd60</td>
</tr>
<tr>
<td>2fd60 1073741824</td>
<td>RWX</td>
<td>ontap-ai-flexvols-retain</td>
</tr>
<tr>
<td>25h</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example High-performance Jobs for ONTAP AI Deployments

This section includes examples of various high-performance jobs that can be executed when Kubernetes is deployed on an ONTAP AI pod.

Example High-performance Jobs for ONTAP AI Deployments

This section includes examples of various high-performance jobs that can be executed when Kubernetes is deployed on an ONTAP AI pod.

Execute a Single-Node AI Workload

To execute a single-node AI and ML job in your Kubernetes cluster, perform the following tasks from the deployment jump host. With Trident, you can quickly and easily make a data volume, potentially containing petabytes of data, accessible to a Kubernetes workload. To make such a data volume accessible from within a Kubernetes pod, simply specify a PVC in the pod definition. This step is a Kubernetes-native operation; no NetApp expertise is required.
This section assumes that you have already containerized (in the Docker container format) the specific AI and ML workload that you are attempting to execute in your Kubernetes cluster.

1. The following example commands show the creation of a Kubernetes job for a TensorFlow benchmark workload that uses the ImageNet dataset. For more information about the ImageNet dataset, see the ImageNet website.

This example job requests eight GPUs and therefore can run on a single GPU worker node that features eight or more GPUs. This example job could be submitted in a cluster for which a worker node featuring eight or more GPUs is not present or is currently occupied with another workload. If so, then the job remains in a pending state until such a worker node becomes available.

Additionally, in order to maximize storage bandwidth, the volume that contains the needed training data is mounted twice within the pod that this job creates. Another volume is also mounted in the pod. This second volume will be used to store results and metrics. These volumes are referenced in the job definition by using the names of the PVCs. For more information about Kubernetes jobs, see the official Kubernetes documentation.

An emptyDir volume with a medium value of Memory is mounted to /dev/shm in the pod that this example job creates. The default size of the /dev/shm virtual volume that is automatically created by the Docker container runtime can sometimes be insufficient for TensorFlow’s needs. Mounting an emptyDir volume as in the following example provides a sufficiently large /dev/shm virtual volume. For more information about emptyDir volumes, see the official Kubernetes documentation.

The single container that is specified in this example job definition is given a securityContext > privileged value of true. This value means that the container effectively has root access on the host. This annotation is used in this case because the specific workload that is being executed requires root access. Specifically, a clear cache operation that the workload performs requires root access. Whether or not this privileged: true annotation is necessary depends on the requirements of the specific workload that you are executing.

```
$ cat << EOF > ./netapp-tensorflow-single-imagenet.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-single-imagenet
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
EOF
```
- name: results
  persistentVolumeClaim:
    claimName: tensorflow-results
  containers:
    - name: netapp-tensorflow-py2
      image: netapp/tensorflow-py2:19.03.0
      command: 
        ["python", "/netapp/scripts/run.py", 
        "--dataset_dir=/mnt/mount_0/dataset/imagenet", 
        "--dgx_version=dgx1", 
        "--num_devices=8"]
      resources:
        limits:
          nvidia.com/gpu: 8
      volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
      securityContext:
        privileged: true
      restartPolicy: Never

EOF

$ kubectl create -f ./netapp-tensorflow-single-imagenet.yaml
job.batch/netapp-tensorflow-single-imagenet created

$ kubectl get jobs

NAME                                       COMPLETIONS   DURATION   AGE
netapp-tensorflow-single-imagenet          0/1           24s        24s

2. Confirm that the job that you created in step 1 is running correctly. The following example command confirms that a single pod was created for the job, as specified in the job definition, and that this pod is currently running on one of the GPU worker nodes.

$ kubectl get pods -o wide

NAME                      READY   STATUS    RESTARTS   AGE
RESTARTS   AGE
IP          NODE           NOMINATED NODE
netapp-tensorflow-single-imagenet-m7x92 1/1     Running   0 3m
10.233.68.61    10.61.218.154   <none>

3. Confirm that the job that you created in step 1 completes successfully. The following example commands confirm that the job completed successfully.
<table>
<thead>
<tr>
<th>NAME</th>
<th>COMPLETIONS</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>netapp-tensorflow-single-imagenet</td>
<td>1/1</td>
<td>5m42s</td>
</tr>
<tr>
<td>10m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$ kubectl get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>netapp-tensorflow-single-imagenet-m7x92</td>
<td>0/1</td>
<td>Completed</td>
</tr>
<tr>
<td>0</td>
<td>11m</td>
<td></td>
</tr>
</tbody>
</table>

$ kubectl logs netapp-tensorflow-single-imagenet-m7x92

```
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at line 702
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at line 711
Total images/sec = 6530.59125

================ Clean Cache !!! ==================
```

```bash
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 8 -H localhost:8 -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH python /netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=resnet50 --batch_size=256 --device=gpu --force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48 --variable_update=horovod --batch_group_size=20 --num_batches=500 --nondistortions --num_gpus=1 --data_format=NCHW --use_fp16=True --use_tf_layers=False --data_name=imagenet --use_datasets=True --data_dir=/mnt/mount_0/dataset/imagenet --datasets_parallel_interleave_cycle_length=10 --datasets_sloppy_parallel_interleave=False --num_mounts=2 --mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000 --datasets_use_prefetch=True --datasets_num_private_threads=4 --horovod_device=gpu > /tmp/20190814_105450_tensorflow_horovod_rdma_resnet50_gpu_8_256_b500_imagenet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

4. **Optional**: Clean up job artifacts. The following example commands show the deletion of the job object that was created in step 1.

When you delete the job object, Kubernetes automatically deletes any associated pods.
Execute a Synchronous Distributed AI Workload

To execute a synchronous multinode AI and ML job in your Kubernetes cluster, perform the following tasks on the deployment jump host. This process enables you to take advantage of data that is stored on a NetApp volume and to use more GPUs than a single worker node can provide. See the following figure for a depiction of a synchronous distributed AI job.

Synchronous distributed jobs can help increase performance and training accuracy compared with asynchronous distributed jobs. A discussion of the pros and cons of synchronous jobs versus asynchronous jobs is outside the scope of this document.

1. The following example commands show the creation of one worker that participates in the synchronous distributed execution of the same TensorFlow benchmark job that was executed on a single node in the example in the section Execute a Single-Node AI Workload. In this specific example, only a single worker is deployed because the job is executed across two worker nodes.
This example worker deployment requests eight GPUs and thus can run on a single GPU worker node that features eight or more GPUs. If your GPU worker nodes feature more than eight GPUs, to maximize performance, you might want to increase this number to be equal to the number of GPUs that your worker nodes feature. For more information about Kubernetes deployments, see the official Kubernetes documentation.

A Kubernetes deployment is created in this example because this specific containerized worker would never complete on its own. Therefore, it doesn’t make sense to deploy it by using the Kubernetes job construct. If your worker is designed or written to complete on its own, then it might make sense to use the job construct to deploy your worker.

The pod that is specified in this example deployment specification is given a hostNetwork value of true. This value means that the pod uses the host worker node’s networking stack instead of the virtual networking stack that Kubernetes usually creates for each pod. This annotation is used in this case because the specific workload relies on Open MPI, NCCL, and Horovod to execute the workload in a synchronous distributed manner. Therefore, it requires access to the host networking stack. A discussion about Open MPI, NCCL, and Horovod is outside the scope of this document. Whether or not this hostNetwork: true annotation is necessary depends on the requirements of the specific workload that you are executing. For more information about the hostNetwork field, see the official Kubernetes documentation.

```yaml
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-worker.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: netapp-tensorflow-multi-imagenet-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: netapp-tensorflow-multi-imagenet-worker
  template:
    metadata:
      labels:
        app: netapp-tensorflow-multi-imagenet-worker
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
EOF
```
claimName: tensorflow-results
containers:
  - name: netapp-tensorflow-py2
    image: netapp/tensorflow-py2:19.03.0
    command: ["bash", "/netapp/scripts/start-slave-multi.sh", "22122"]
resources:
  limits:
    nvidia.com/gpu: 8
volumeMounts:
  - mountPath: /dev/shm
    name: dshm
  - mountPath: /mnt/mount_0
    name: testdata-iface1
  - mountPath: /mnt/mount_1
    name: testdata-iface2
  - mountPath: /tmp
    name: results
securityContext:
  privileged: true
EOF

$ kubectl create -f ./netapp-tensorflow-multi-imagenet-worker.yaml
deployment.apps/netapp-tensorflow-multi-imagenet-worker created

$ kubectl get deployments
NAME                                      DESIRED   CURRENT   UP-TO-DATE
AVAILAble   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           4s

2. Confirm that the worker deployment that you created in step 1 launched successfully. The following example commands confirm that a single worker pod was created for the deployment, as indicated in the deployment definition, and that this pod is currently running on one of the GPU worker nodes.

$ kubectl get pods -o wide
NAME                                                       READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          60s   10.61.218.154   10.61.218.154   <none>

$ kubectl logs netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725
22122

3. Create a Kubernetes job for a master that kicks off, participates in, and tracks the execution of the synchronous multinode job. The following example commands create one master that kicks off, participates in, and tracks the synchronous distributed execution of the same TensorFlow benchmark job that was executed on a single node in the example in the section Execute a Single-Node AI Workload.
This example master job requests eight GPUs and thus can run on a single GPU worker node that features eight or more GPUs. If your GPU worker nodes feature more than eight GPUs, to maximize performance, you might want to increase this number to be equal to the number of GPUs that your worker nodes feature.

The master pod that is specified in this example job definition is given a hostNetwork value of true, just as the worker pod was given a hostNetwork value of true in step 1. See step 1 for details about why this value is necessary.

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-master.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-multi-imagenet-master
spec:
  backoffLimit: 5
  template:
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
        - mountPath: /mnt/mount_1
EOF
```
4. Confirm that the master job that you created in step 3 is running correctly. The following example command confirms that a single master pod was created for the job, as indicated in the job definition, and that this pod is currently running on one of the GPU worker nodes. You should also see that the worker pod that you originally saw in step 1 is still running and that the master and worker pods are running on different nodes.

$ kubectl get pods -o wide
NAME                                                       READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-master-ppwwj              1/1
Running 0 45s 10.61.218.152 10.61.218.152 <none>
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running 0 26m 10.61.218.154 10.61.218.154 <none>

5. Confirm that the master job that you created in step 3 completes successfully. The following example commands confirm that the job completed successfully.

$ kubectl get jobs
NAME                                      COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      9m18s
$ kubectl get pods
NAME                                                       READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj              0/1
Completed 0 9m38s
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running 0 35m
$ kubectl logs netapp-tensorflow-multi-imagenet-master-ppwwj
[10.61.218.152:00008] WARNING: local probe returned unhandled
shell:unknown assuming bash
rm: cannot remove '/lib': Is a directory
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
6. Delete the worker deployment when you no longer need it. The following example commands show the deletion of the worker deployment object that was created in step 1.

When you delete the worker deployment object, Kubernetes automatically deletes any associated worker pods.
$ kubectl get deployments
NAME                                      DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           43m
$ kubectl get pods
NAME                                      READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj 0/1
Completed   0          17m
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725 1/1
Running     0          43m
$ kubectl delete deployment netapp-tensorflow-multi-imagenet-worker
deployment.extensions "netapp-tensorflow-multi-imagenet-worker" deleted
$ kubectl get deployments
No resources found.
$ kubectl get pods
NAME                                      READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj 0/1     Completed   0
18m

7. **Optional**: Clean up the master job artifacts. The following example commands show the deletion of the master job object that was created in step 3.

When you delete the master job object, Kubernetes automatically deletes any associated master pods.

$ kubectl get jobs
NAME                                      COMPLETIONS   DURATION  AGE
netapp-tensorflow-multi-imagenet-master     1/1       5m50s      19m
$ kubectl get pods
NAME                                      READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj 0/1     Completed   0
19m
$ kubectl delete job netapp-tensorflow-multi-imagenet-master
job.batch "netapp-tensorflow-multi-imagenet-master" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
Performance Testing

We performed a simple performance comparison as part of the creation of this solution. We executed several standard NetApp AI benchmarking jobs by using Kubernetes, and we compared the benchmark results with executions that were performed by using a simple Docker run command. We did not see any noticeable differences in performance. Therefore, we concluded that the use of Kubernetes to orchestrate containerized AI training jobs does not adversely affect performance. See the following table for the results of our performance comparison.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Dataset</th>
<th>Docker Run (images/sec)</th>
<th>Kubernetes (images/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-node TensorFlow</td>
<td>Synthetic data</td>
<td>6,667.2475</td>
<td>6,661.93125</td>
</tr>
<tr>
<td>Single-node TensorFlow</td>
<td>ImageNet</td>
<td>6,570.2025</td>
<td>6,530.59125</td>
</tr>
<tr>
<td>Synchronous distributed two-node TensorFlow</td>
<td>Synthetic data</td>
<td>13,213.70625</td>
<td>13,218.288125</td>
</tr>
<tr>
<td>Synchronous distributed two-node TensorFlow</td>
<td>ImageNet</td>
<td>12,941.69125</td>
<td>12,881.33875</td>
</tr>
</tbody>
</table>

Conclusion

Companies and organizations of all sizes and across all industries are turning to artificial intelligence (AI), machine learning (ML), and deep learning (DL) to solve real-world problems, deliver innovative products and services, and to get an edge in an increasingly competitive marketplace. As organizations increase their use of AI, ML, and DL, they face many challenges, including workload scalability and data availability. These challenges can be addressed through the use of the NetApp AI Control Plane solution.

This solution enables you to rapidly clone a data namespace. Additionally, it allows you to define and implement AI, ML, and DL training workflows that incorporate the near-instant creation of data and model baselines for traceability and versioning. With this solution, you can trace every single model training run back to the exact dataset(s) that the model was trained and/or validated with. Lastly, this solution enables you to swiftly provision Jupyter Notebook workspaces with access to massive datasets.

Because this solution is targeted towards data scientists and data engineers, minimal NetApp or NetApp ONTAP expertise is required. With this solution, data management functions can be executed using simple and familiar tools and interfaces. Furthermore, this solution utilizes fully open-source and free components. Therefore, if you already have NetApp storage in your environment, you can implement this solution today. If you want to test drive this solution but you do not have already have NetApp storage, visit cloud.netapp.com, and you can be up and running with a cloud-based NetApp storage solution in no time.

MLRun Pipeline with Iguazio

TR-4834: NetApp and Iguazio for MLRun Pipeline

Rick Huang, David Arnette, NetApp
Marcelo Litovsky, Iguazio
This document covers the details of the MLRun pipeline using NetApp ONTAP AI, NetApp AI Control Plane, NetApp Cloud Volumes software, and the Iguazio Data Science Platform. We used Nuclio serverless function, Kubernetes Persistent Volumes, NetApp Cloud Volumes, NetApp Snapshot copies, Grafana dashboard, and other services on the Iguazio platform to build an end-to-end data pipeline for the simulation of network failure detection. We integrated Iguazio and NetApp technologies to enable fast model deployment, data replication, and production monitoring capabilities on premises as well as in the cloud.

The work of a data scientist should be focused on the training and tuning of machine learning (ML) and artificial intelligence (AI) models. However, according to research by Google, data scientists spend ~80% of their time figuring out how to make their models work with enterprise applications and run at scale, as shown in the following image depicting model development in the AI/ML workflow.

To manage end-to-end AI/ML projects, a wider understanding of enterprise components is needed. Although DevOps have taken over the definition, integration, and deployment these types of components, machine learning operations target a similar flow that includes AI/ML projects. To get an idea of what an end-to-end AI/ML pipeline touches in the enterprise, see the following list of required components:

- Storage
- Networking
- Databases
- File systems
- Containers
- Continuous integration and continuous deployment (CI/CD) pipeline
- Development integrated development environment (IDE)
In this paper, we demonstrate how the partnership between NetApp and Iguazio drastically simplifies the development of an end-to-end AI/ML pipeline. This simplification accelerates the time to market for all of your AI/ML applications.

Target Audience

The world of data science touches multiple disciplines in information technology and business.

- The data scientist needs the flexibility to use their tools and libraries of choice.
- The data engineer needs to know how the data flows and where it resides.
- A DevOps engineer needs the tools to integrate new AI/ML applications into their CI/CD pipelines.
- Business users want to have access to AI/ML applications. We describe how NetApp and Iguazio help each of these roles bring value to business with our platforms.

Solution Overview

This solution follows the lifecycle of an AI/ML application. We start with the work of data scientists to define the different steps needed to prep data and train and deploy models. We follow with the work needed to create a full pipeline with the ability to track artifacts, experiment with execution, and deploy to Kubeflow. To complete the full cycle, we integrate the pipeline with NetApp Cloud Volumes to enable data versioning, as seen in the following image.
Technology Overview

NetApp Overview

NetApp is the data authority for the hybrid cloud. NetApp provides a full range of hybrid cloud data services that simplify management of applications and data across cloud and on-premises environments to accelerate digital transformation. Together with our partners, NetApp empowers global organizations to unleash the full potential of their data to expand customer touch points, foster greater innovation, and optimize their operations.

NetApp ONTAP AI

NetApp ONTAP AI, powered by NVIDIA DGX systems and NetApp cloud-connected all-flash storage, streamlines the flow of data reliably and speeds up analytics, training, and inference with your data fabric that spans from edge to core to cloud. It gives IT organizations an architecture that provides the following benefits:

• Eliminates design complexities
• Allows independent scaling of compute and storage
• Enables customers to start small and scale seamlessly
• Offers a range of storage options for various performance and cost points

NetApp ONTAP AI offers converged infrastructure stacks incorporating NVIDIA DGX-1, a petaflop-scale AI system, and NVIDIA Mellanox high-performance Ethernet switches to unify AI workloads, simplify deployment, and accelerate ROI. We leveraged ONTAP AI with one DGX-1 and NetApp AFF A800 storage system for this technical report. The following image shows the topology of ONTAP AI with the DGX-1 system used in this validation.
NetApp AI Control Plane

The NetApp AI Control Plane enables you to unleash AI and ML with a solution that offers extreme scalability, streamlined deployment, and nonstop data availability. The AI Control Plane solution integrates Kubernetes and Kubeflow with a data fabric enabled by NetApp. Kubernetes, the industry-standard container orchestration platform for cloud-native deployments, enables workload scalability and portability. Kubeflow is an open-source machine-learning platform that simplifies management and deployment, enabling developers to do more data science in less time. A data fabric enabled by NetApp offers uncompromising data availability and portability to make sure that your data is accessible across the pipeline, from edge to core to cloud. This technical report uses the NetApp AI Control Plane in an MLRun pipeline. The following image shows Kubernetes cluster management page where you can have different endpoints for each cluster. We connected NFS Persistent Volumes to the Kubernetes cluster, and the following images show an Persistent Volume connected to the cluster, where NetApp Trident offers persistent storage support and data management capabilities.
Persistent Volumes for Kubernetes

Connected with Kubernetes Cluster

Cloud Volumes ONTAP is connected to 1 Kubernetes cluster. View Cluster  

You can connect another Kubernetes cluster to this Cloud Volumes ONTAP system. If the Kubernetes cluster is in a different network than Cloud Volumes ONTAP, specify a custom export policy to provide access to clients.

Kubernetes Cluster

Select Kubernetes Cluster

Custom Export Policy (Optional)

Custom Export Policy

172.31.0.0/16

Set as default storage class

- NFS
- iSCSI

Connect  Cancel
Iguazio Overview

The Iguazio Data Science Platform is a fully integrated and secure data-science platform as a service (PaaS) that simplifies development, accelerates performance, facilitates collaboration, and addresses operational challenges. This platform incorporates the following components, and the Iguazio Data Science Platform is presented in the following image:

- A data-science workbench that includes Jupyter Notebooks, integrated analytics engines, and Python packages
- Model management with experiments tracking and automated pipeline capabilities
- Managed data and ML services over a scalable Kubernetes cluster
- Nuclio, a real-time serverless functions framework
- An extremely fast and secure data layer that supports SQL, NoSQL, time-series databases, files (simple objects), and streaming
- Integration with third-party data sources such as NetApp, Amazon S3, HDFS, SQL databases, and streaming or messaging protocols
- Real-time dashboards based on Grafana
Software and Hardware Requirements

Network Configuration

The following is the network configuration requirement for setting up in the cloud:

- The Iguazio cluster and NetApp Cloud Volumes must be in the same virtual private cloud.
- The cloud manager must have access to port 6443 on the Iguazio app nodes.
- We used Amazon Web Services in this technical report. However, users have the option of deploying the solution in any Cloud provider. For on-premises testing in ONTAP AI with NVIDIA DGX-1, we used the Iguazio hosted DNS service for convenience.

Clients must be able to access dynamically created DNS domains. Customers can use their own DNS if desired.

Hardware Requirements

You can install Iguazio on-premises in your own cluster. We have verified the solution in NetApp ONTAP AI with an NVIDIA DGX-1 system. The following table lists the hardware used to test this solution.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGX-1 systems</td>
<td>1</td>
</tr>
<tr>
<td>NetApp AFF A800 system</td>
<td>1 high-availability (HA) pair, includes 2 controllers and 48 NVMe SSDs (3.8TB or above)</td>
</tr>
<tr>
<td>Cisco Nexus 3232C network switches</td>
<td>2</td>
</tr>
</tbody>
</table>

The following table lists the software components required for on-premise testing:
This solution was fully tested with Iguazio version 2.5 and NetApp Cloud Volumes ONTAP for AWS. The Iguazio cluster and NetApp software are both running on AWS.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version or Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iguazio</td>
<td>Version 2.8+</td>
</tr>
<tr>
<td>App node</td>
<td>M5.4xlarge</td>
</tr>
<tr>
<td>Data node</td>
<td>I3.4xlarge</td>
</tr>
</tbody>
</table>

**Network Device Failure Prediction Use Case Summary**

This use case is based on an Iguazio customer in the telecommunications space in Asia. With 100K enterprise customers and 125k network outage events per year, there was a critical need to predict and take proactive action to prevent network failures from affecting customers. This solution provided them with the following benefits:

- Predictive analytics for network failures
- Integration with a ticketing system
- Taking proactive action to prevent network failures

As a result of this implementation of Iguazio, 60% of failures were proactively prevented.

**Setup Overview**

Iguazio can be installed on-premises or on a cloud provider.

**Iguazio Installation**

 Provisioning can be done as a service and managed by Iguazio or by the customer. In both cases, Iguazio provides a deployment application (Provazio) to deploy and manage clusters.

For on-premises installation, please refer to NVA-1121 for compute, network, and storage setup. On-premises deployment of Iguazio is provided by Iguazio without additional cost to the customer. See this page for DNS and SMTP server configurations. The Provazio installation page is shown as follows.
This section is divided into two parts for cloud and on-premises deployment respectively.

### Cloud Deployment Kubernetes Configuration

Through NetApp Cloud Manager, you can define the connection to the Iguazio Kubernetes cluster. Trident requires access to multiple resources in the cluster to make the volume available.

1. To enable access, obtain the Kubernetes config file from one the Iguazio nodes. The file is located under `/home/Iguazio/.kube/config`. Download this file to your desktop.

2. Go to Discover Cluster to configure.
3. Upload the Kubernetes config file. See the following image.

Upload Kubernetes Configuration File

Upload the Kubernetes configuration file (kubeconfig) so Cloud Manager can install Trident on the Kubernetes cluster.

Connecting Cloud Volumes ONTAP with a Kubernetes cluster enables users to request and manage persistent volumes using native Kubernetes interfaces and constructs. Users can take advantage of ONTAP’s advanced data management features without having to know anything about it. Storage provisioning is enabled by using NetApp Trident. Learn more about Trident for Kubernetes.

Upload File

4. Deploy Trident and associate a volume with the cluster. See the following image on defining and assigning a Persistent Volume to the Iguazio cluster. This process creates a Persistent Volume (PV) in Iguazio’s Kubernetes cluster. Before you can use it, you must define a Persistent Volume Claim (PVC).
On-Premises Deployment Kubernetes Configuration

For on-premises installation of NetApp Trident, see TR-4798 for details. After configuring your Kubernetes cluster and installing NetApp Trident, you can connect Trident to the Iguazio cluster to enable NetApp data management capabilities, such as taking Snapshot copies of your data and model.

Define Persistent Volume Claim

1. Save the following YAML to a file to create a PVC of type Basic.

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
storageClassName: netapp-file
```

2. Apply the YAML file to your Iguazio Kubernetes cluster.
Kubectl -n default-tenant apply -f <your yaml file>

**Attach NetApp Volume to the Jupyter Notebook**

Iguazio offers several managed services to provide data scientists with a full end-to-end stack for development and deployment of AI/ML applications. You can read more about these components at the Iguazio Overview of Application Services and Tools.

One of the managed services is Jupyter Notebook. Each developer gets its own deployment of a notebook container with the resources they need for development. To give them access to the NetApp Cloud Volume, you can assign the volume to their container and resource allocation, running user, and environment variable settings for Persistent Volume Claims is presented in the following image.

For an on-premises configuration, you can refer to TR-4798 on the Trident setup to enable NetApp ONTAP data management capabilities, such as taking Snapshot copies of your data or model for versioning control. Add the following line in your Trident back-end config file to make Snapshot directories visible:

```json
{
    "defaults": {
        "snapshotDir": "true"
    }
}
```

You must create a Trident back-end config file in JSON format, and then run the following Trident command to reference it:

```
tridentctl create backend -f <backend-file>
```

**Deploying the Application**

The following sections describe how to install and deploy the application.
Get Code from GitHub

Now that the NetApp Cloud Volume or NetApp Trident volume is available to the Iguazio cluster and the developer environment, you can start reviewing the application.

Users have their own workspace (directory). On every notebook, the path to the user directory is /User. The Iguazio platform manages the directory. If you follow the instructions above, the NetApp Cloud volume is available in the /netapp directory.

Get the code from GitHub using a Jupyter terminal.

At the Jupyter terminal prompt, clone the project.

```
cd /User
git clone .
```

You should now see the netops-netapp folder on the file tree in Jupyter workspace.

Configure Working Environment

Copy the Notebook set_env-Example.ipynb as set_env.ipynb. Open and edit set_env.ipynb. This notebook sets variables for credentials, file locations, and execution drivers.

If you follow the instructions above, the following steps are the only changes to make:

1. Obtain this value from the Iguazio services dashboard: `docker_registry`

   Example: `docker-registry.default-tenant.app.clusterq.iguazidev.com:80`
2. Change `admin` to your Iguazio username:

```
IGZ_CONTAINER_PATH = '/users/admin'
```

The following are the ONTAP system connection details. Include the volume name that was generated when Trident was installed. The following setting is for an on-premises ONTAP cluster:

```
ontapClusterMgmtHostname = '0.0.0.0'
ontapClusterAdminUsername = 'USER'
ontapClusterAdminPassword = 'PASSWORD'
sourceVolumeName = 'SOURCE VOLUME'
```

The following setting is for Cloud Volumes ONTAP:

```
MANAGER=ontapClusterMgmtHostname
svm='svm'
email='email'
password=ontapClusterAdminPassword
weid="weid"
volume=sourceVolumeName
```

Create Base Docker Images

Everything you need to build an ML pipeline is included in the Iguazio platform. The developer can define the specifications of the Docker images required to run the pipeline and execute the image creation from Jupyter Notebook. Open the notebook `create_images.ipynb` and Run All Cells.

This notebook creates two images that we use in the pipeline.

- `iguazio/netapp`. Used to handle ML tasks.

Create image for training pipeline

```
[4]: fn.build_config(image=docker_registry + '/iguazio/netapp', commands=['pip install \n v3io_frames fsspec==0.3.3 PyYAML==5.1.2 pyarrow==0.15.1 pandas==0.25.3 matplotlib seaborn yellowbr
 fn.deploy()]
```

- `netapp/pipeline`. Contains utilities to handle NetApp Snapshot copies.

Create image for Ontap utilities

```
[9]: fn.build_config(image=dockerregistry + '/netapp/pipeline:latest',commands=['apt -y update', 'pip install v3io_frames netapp_ontapp
 fn.deploy()]
```

Review Individual Jupyter Notebooks

The following table lists the libraries and frameworks we used to build this task. All these components have been fully integrated with Iguazio’s role-based access and security controls.
<table>
<thead>
<tr>
<th>Libraries/Framework</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLRun</td>
<td>An managed by Iguazio to enable the assembly, execution, and monitoring of an ML/AI pipeline.</td>
</tr>
<tr>
<td>Nuclio</td>
<td>A serverless functions framework integrated with Iguazio. Also available as an open-source project managed by Iguazio.</td>
</tr>
<tr>
<td>Kubeflow</td>
<td>A Kubernetes-based framework to deploy the pipeline. This is also an open-source project to which Iguazio contributes. It is integrated with Iguazio for added security and integration with the rest of the infrastructure.</td>
</tr>
<tr>
<td>Docker</td>
<td>A Docker registry run as a service in the Iguazio platform. You can also change this to connect to your registry.</td>
</tr>
<tr>
<td>NetApp Cloud Volumes</td>
<td>Cloud Volumes running on AWS give us access to large amounts of data and the ability to take Snapshot copies to version the datasets used for training.</td>
</tr>
<tr>
<td>Trident</td>
<td>Trident is an open-source project managed by NetApp. It facilitates the integration with storage and compute resources in Kubernetes.</td>
</tr>
</tbody>
</table>

We used several notebooks to construct the ML pipeline. Each notebook can be tested individually before being brought together in the pipeline. We cover each notebook individually following the deployment flow of this demonstration application.

The desired result is a pipeline that trains a model based on a Snapshot copy of the data and deploys the model for inference. A block diagram of a completed MLRun pipeline is shown in the following image.
Deploy Data Generation Function

This section describes how we used Nuclio serverless functions to generate network device data. The use case is adapted from an Iguazio client that deployed the pipeline and used Iguazio services to monitor and predict network device failures.

We simulated data coming from network devices. Executing the Jupyter notebook `data-generator.ipynb` creates a serverless function that runs every 10 minutes and generates a Parquet file with new data. To deploy the function, run all the cells in this notebook. See the Nuclio website to review any unfamiliar components in this notebook.

A cell with the following comment is ignored when generating the function. Every cell in the notebook is assumed to be part of the function. Import the Nuclio module to enable `%nuclio magic`.

```
# nuclio: ignore
import nuclio
```

In the spec for the function, we defined the environment in which the function executes, how it is triggered, and the resources it consumes.
spec = nuclio.ConfigSpec(config={"spec.triggers.inference.kind":"cron",
"spec.triggers.inference.attributes.interval" :"10m",
    "spec.readinessTimeoutSeconds" : 60,
    "spec.minReplicas" : 1},......

The `init_context` function is invoked by the Nuclio framework upon initialization of the function.

def init_context(context):
    ...

Any code not in a function is invoked when the function initializes. When you invoke it, a handler function is executed. You can change the name of the handler and specify it in the function spec.

def handler(context, event):
    ...

You can test the function from the notebook prior to deployment.

```python
%%time
# nuclio: ignore
init_context(context)
event = nuclio.Event(body='')
output = handler(context, event)
output
```

The function can be deployed from the notebook or it can be deployed from a CI/CD pipeline (adapting this code).

```python
addr = nuclio.deploy_file(name='generator',project='netops',spec=spec,
tag='v1.1')
```

**Pipeline Notebooks**

These notebooks are not meant to be executed individually for this setup. This is just a review of each notebook. We invoked them as part of the pipeline. To execute them individually, review the MLRun documentation to execute them as Kubernetes jobs.

**snap_cv.ipynb**

This notebook handles the Cloud Volume Snapshot copies at the beginning of the pipeline. It passes the name of the volume to the pipeline context. This notebook invokes a shell script to handle the Snapshot copy. While running in the pipeline, the execution context contains variables to help locate all files needed for execution.
While writing this code, the developer does not have to worry about the file location in the container that executes it. As described later, this application is deployed with all its dependencies, and it is the definition of the pipeline parameters that provides the execution context.

```
command = os.path.join(context.get_param('APP_DIR'),"snap_cv.sh")
```

The created Snapshot copy location is placed in the MLRun context to be consumed by steps in the pipeline.

```
context.log_result('snapVolumeDetails',snap_path)
```

The next three notebooks are run in parallel.

**data-prep.ipynb**

Raw metrics must be turned into features to enable model training. This notebook reads the raw metrics from the Snapshot directory and writes the features for model training to the NetApp volume.

When running in the context of the pipeline, the input `DATA_DIR` contains the Snapshot copy location.

```
metrics_table = os.path.join(str(mlruncontext.get_input('DATA_DIR',
  os.getenv('DATA_DIR','/netpp'))),
  mlruncontext.get_param('metrics_table',
  os.getenv('metrics_table','netops_metrics_parquet')))
```

**describe.ipynb**

To visualize the incoming metrics, we deploy a pipeline step that provides plots and graphs that are available through the Kubeflow and MLRun UIs. Each execution has its own version of this visualization tool.

```
ax.set_title("features correlation")
plt.savefig(os.path.join(base_path, "plots/corr.png"))
context.log_artifact(PlotArtifact("correlation", body=plt.gcf()),
local_path="plots/corr.html")
```

**deploy-feature-function.ipynb**

We continuously monitor the metrics looking for anomalies. This notebook creates a serverless function that generates the features need to run prediction on incoming metrics. This notebook invokes the creation of the function. The function code is in the notebook `data-prep.ipynb`. Notice that we use the same notebook as a step in the pipeline for this purpose.

**training.ipynb**

After we create the features, we trigger the model training. The output of this step is the model to be used for inferencing. We also collect statistics to keep track of each execution (experiment).
For example, the following command enters the accuracy score into the context for that experiment. This value is visible in Kubeflow and MLRun.

```python
context.log_result('accuracy', score)
```

**deploy-inference-function.ipynb**

The last step in the pipeline is to deploy the model as a serverless function for continuous inferencing. This notebook invokes the creation of the serverless function defined in `nuclio-inference-function.ipynb`.

**Review and Build Pipeline**

The combination of running all the notebooks in a pipeline enables the continuous run of experiments to reassess the accuracy of the model against new metrics. First, open the `pipeline.ipynb` notebook. We take you through details that show how NetApp and Iguazio simplify the deployment of this ML pipeline.

We use MLRun to provide context and handle resource allocation to each step of the pipeline. The MLRun API service runs in the Iguazio platform and is the point of interaction with Kubernetes resources. Each developer cannot directly request resources; the API handles the requests and enables access controls.

```python
# MLRun API connection definition
mlconf.dbpath = 'http://mlrun-api:8080'
```

The pipeline can work with NetApp Cloud Volumes and on-premises volumes. We built this demonstration to use Cloud Volumes, but you can see in the code the option to run on-premises.
# Initialize the NetApp snap function once for all functions in a notebook
if [ NETAPP_CLOUD_VOLUME ]:
    snapfn =
code_to_function('snap',project='NetApp',kind='job',filename="snap_cv.ipynb").apply(mount_v3io())
    snap_params = {
        "metrics_table" : metrics_table,
        "NETAPP_MOUNT_PATH" : NETAPP_MOUNT_PATH,
        'MANAGER' : MANAGER,
        'svm' : svm,
        'email': email,
        'password': password ,
        'weid': weid,
        'volume': volume,
        "APP_DIR" : APP_DIR
    }
else:
    snapfn =
code_to_function('snap',project='NetApp',kind='job',filename="snapshot.ipynb").apply(mount_v3io())
    ...
    snapfn.spec.image = docker Registry + '/netapp/pipeline:latest'
    snapfn.spec.volume_mounts =
    [snapfn.spec.volume_mounts[0],netapp_volume_mounts]
    snapfn.spec.volumes = [ snapfn.spec.volumes[0],netapp_volumes]

The first action needed to turn a Jupyter notebook into a Kubeflow step is to turn the code into a function. A function has all the specifications required to run that notebook. As you scroll down the notebook, you can see that we define a function for every step in the pipeline.

<table>
<thead>
<tr>
<th>Part of the Notebook</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;code_to_function&gt;</td>
<td>Name of the function: Project name. used to organize all project artifacts. This is visible in the MLRun UI. Kind. In this case, a Kubernetes job. This could be Dask, mpi, sparkk8s, and more. See the MLRun documentation for more details. File. The name of the notebook. This can also be a location in Git (HTTP).</td>
</tr>
<tr>
<td>(part of the MLRun module)</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>The name of the Docker image we are using for this step. We created this earlier with the create-image.ipynb notebook.</td>
</tr>
<tr>
<td>volume_mounts &amp; volumes</td>
<td>Details to mount the NetApp Cloud Volume at run time.</td>
</tr>
</tbody>
</table>

We also define parameters for the steps.
After you have the function definition for all steps, you can construct the pipeline. We use the `kfp` module to make this definition. The difference between using MLRun and building on your own is the simplification and shortening of the coding.

The functions we defined are turned into step components using the `as_step` function of MLRun.

**Snapshot Step Definition**

Initiate a Snapshot function, output, and mount v3io as source:

```python
snap = snapfn.as_step(NewTask(handler='handler',params=snap_params),
    name='NetApp_Cloud_Volume_Snapshot',outputs=['snapVolumeDetails','training_parquet_file']).apply(mount_v3io())
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewTask</td>
<td>NewTask is the definition of the function run.</td>
</tr>
<tr>
<td>(MLRun module)</td>
<td>Handler. Name of the Python function to invoke. We used the name handler in the notebook, but it is not required. params. The parameters we passed to the execution. Inside our code, we use context.get_param('PARAMETER') to get the values.</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>as_step</td>
<td>Name. Name of the Kubeflow pipeline step.</td>
</tr>
<tr>
<td>inputs</td>
<td>You can pass to a step the outputs of a previous step. In this case, snap.outputs['snapVolumeDetails'] is the name of the Snapshot copy we created on the snap step.</td>
</tr>
<tr>
<td>out_path</td>
<td>A location to place artifacts generating using the MLRun module log_artifacts.</td>
</tr>
</tbody>
</table>

```
prep = data_prep.as_step(name='data-prep',
handler='handler',params=params,
    inputs = {'DATA_DIR':
    snap.outputs['snapVolumeDetails']},

out_path=artifacts_path).apply(mount_v3io()).after(snap)
```

You can run pipeline.ipynb from top to bottom. You can then go to the Pipelines tab from the Iguazio dashboard to monitor progress as seen in the Iguazio dashboard Pipelines tab.
Because we logged the accuracy of training step in every run, we have a record of accuracy for each experiment, as seen in the record of training accuracy.

<table>
<thead>
<tr>
<th>Run name</th>
<th>Status</th>
<th>Duration</th>
<th>Pipeline Version</th>
<th>Recurring</th>
<th>Start time</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>xgb_pipeline 2020-03-24 18:51-...</td>
<td>✔️</td>
<td>0:08:43</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/24/2020, 2:51:09 PM</td>
<td>0.985</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-19 13:31-...</td>
<td>✔️</td>
<td>0:06:14</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/19/2020, 9:31:19 AM</td>
<td>0.980</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-18 12:56-...</td>
<td>✔️</td>
<td>0:08:11</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/18/2020, 8:56:08 AM</td>
<td>0.990</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-17 19:49-...</td>
<td>✔️</td>
<td>0:08:03</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/17/2020, 3:49:31 PM</td>
<td>0.985</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-17 18:34-...</td>
<td>✔️</td>
<td>0:05:54</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/17/2020, 2:34:56 PM</td>
<td>0.980</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-17 17:34-...</td>
<td>✔️</td>
<td>0:04:48</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/17/2020, 1:34:16 PM</td>
<td>0.982</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-17 17:01-...</td>
<td>✔️</td>
<td>0:05:25</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/17/2020, 1:01:58 PM</td>
<td>0.987</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-16 16:47-...</td>
<td>✔️</td>
<td>0:06:08</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/16/2020, 12:47:19 ...</td>
<td>0.983</td>
</tr>
<tr>
<td>xgb_pipeline 2020-03-16 13:57-...</td>
<td>✔️</td>
<td>0:05:18</td>
<td>[View pipeline]</td>
<td>-</td>
<td>3/15/2020, 9:57:03 AM</td>
<td>0.980</td>
</tr>
</tbody>
</table>

If you select the Snapshot step, you can see the name of the Snapshot copy that was used to run this experiment.
The described step has visual artifacts to explore the metrics we used. You can expand to view the full plot as seen in the following image.

The MLRun API database also tracks inputs, outputs, and artifacts for each run organized by project. An example of inputs, outputs, and artifacts for each run can be seen in the following image.
For each job, we store additional details.

There is more information about MLRun than we can cover in this document. All artifacts, including the definition of the steps and functions, can be saved to the API database, versioned, and invoked individually or as a full project. Projects can also be saved and pushed to Git for later use. We encourage you to learn more at the MLRun GitHub site.

Deploy Grafana Dashboard

After everything is deployed, we run inferences on new data. The models predict failure on network device equipment. The results of the prediction are stored in an Iguazio TimeSeries table. You can visualize the results with Grafana in the platform integrated with Iguazio’s security and data access policy.

You can deploy the dashboard by importing the provided JSON file into the Grafana interfaces in the cluster.
1. To verify that the Grafana service is running, look under Services.

![Services Table]

<table>
<thead>
<tr>
<th>Name</th>
<th>Running User</th>
<th>Version</th>
<th>CPU (cores)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker-registry</td>
<td></td>
<td>2.7.1</td>
<td>96μ</td>
<td>1.67 GiB</td>
</tr>
<tr>
<td>framesd</td>
<td></td>
<td>0.5.10</td>
<td>369μ</td>
<td>795.19 MB</td>
</tr>
<tr>
<td>grafana</td>
<td></td>
<td>6.6.0</td>
<td>1m</td>
<td>38.39 MB</td>
</tr>
<tr>
<td>Jupyter</td>
<td>admin</td>
<td>1.0.2</td>
<td>81μ</td>
<td>3.27 GB</td>
</tr>
<tr>
<td>Log Forwarder</td>
<td></td>
<td>6.7.2</td>
<td>0</td>
<td>0 bytes</td>
</tr>
</tbody>
</table>

2. If it is not present, deploy an instance from the Services section:
   a. Click New Service.
   b. Select Grafana from the list.
   c. Accept the defaults.
   d. Click Next Step.
   e. Enter your user ID.
   f. Click Save Service.
   g. Click Apply Changes at the top.

3. To deploy the dashboard, download the file `NetopsPredictions-Dashboard.json` through the Jupyter interface.
4. Open Grafana from the Services section and import the dashboard.

5. Click Upload *.json File and select the file that you downloaded earlier (NetopsPredictions-Dashboard.json). The dashboard displays after the upload is completed.
Deploy Cleanup Function

When you generate a lot of data, it is important to keep things clean and organized. To do so, deploy the cleanup function with the cleanup.ipynb notebook.

Benefits

NetApp and Iguazio speed up and simplify the deployment of AI and ML applications by building in essential frameworks, such as Kubeflow, Apache Spark, and TensorFlow, along with orchestration tools like Docker and Kubernetes. By unifying the end-to-end data pipeline, NetApp and Iguazio reduce the latency and complexity inherent in many advanced computing workloads, effectively bridging the gap between development and operations. Data scientists can run queries on large datasets and securely share data and algorithmic models with authorized users during the training phase. After the containerized models are ready for production, you can easily move them from development environments to operational environments.

Conclusion

When building your own AI/ML pipelines, configuring the integration, management, security, and accessibility of the components in an architecture is a challenging task. Giving developers access and control of their environment presents another set of challenges.

The combination of NetApp and Iguazio brings these technologies together as managed services to accelerate technology adoption and improve the time to market for new AI/ML applications.

**TR-4915: Data movement with E-Series and BeeGFS for AI and analytics workflows**

Cody Harryman and Ryan Rodine, NetApp

TR-4915 describes how to move data from any data repository into a BeeGFS file system.
backed by NetApp E-Series SAN storage. For artificial intelligence (AI) and machine learning (ML) applications, customers might routinely need to move large data sets exceeding many petabytes of data into their BeeGFS clusters for model development. This document explores how to accomplish this by using NetApp XCP and NetApp BlueXP Copy and Sync tools.

TR-4915: Data movement with E-Series and BeeGFS for AI and analytics workflows

Use Cases

Responsible AI and confidential inferencing - NetApp AI with Protopia Image Transformation

TR-4928: Responsible AI and confidential inferencing - NetApp AI with Protopia Image and Data Transformation

Sathish Thyagarajan, Michael Oglesby, NetApp
Byung Hoon Ahn, Jennifer Cwagenberg, Protopia

Visual interpretations have become an integral part of communication with the emergence of image capturing and image processing. Artificial intelligence (AI) in digital image processing brings novel business opportunities, such as in the medical field for cancer and other disease identification, in geospatial visual analytics for studying environmental hazards, in pattern recognition, in video processing for fighting crime, and so on. However, this opportunity also comes with extraordinary responsibilities.

The more decisions organizations put into the hands of AI, the more they accept risks related to data privacy and security and legal, ethical, and regulatory issues. Responsible AI enables a practice that allows companies and government organizations to build trust and governance that is crucial for AI at scale in large enterprises. This document describes an AI inferencing solution validated by NetApp under three different scenarios by using NetApp data management technologies with Protopia data obfuscation software to privatize sensitive data and reduce risks and ethical concerns.

Millions of images are generated every day with various digital devices by both consumers and business entities. The consequent massive explosion of data and computational workload makes businesses turn to cloud computing platforms for scale and efficiency. Meanwhile, privacy concerns over the sensitive information contained in image data arise with transfer to a public cloud. The lack of security and privacy assurances become the main barrier to deployment of image-processing AI systems.

Additionally, there is the right to erasure by the GDPR, the right of an individual to request that an organization erase all their personal data. There is also the Privacy Act, which establishes a code of fair information practices. Digital images such as photographs can constitute personal data under the GDPR, which governs how data must be collected, processed, and erased. Failure to do so is a failure to comply with GDPR, which might lead to hefty fines for breaching compliances that can be seriously damaging to organizations. Privacy principles are among the backbone of implementing responsible AI that ensure fairness in the machine learning (ML) and deep learning (DL) model predictions and lowers risks associated with violating privacy or regulatory compliance.

This document describes a validated design solution under three different scenarios with and without image obfuscation relevant to preserving privacy and deploying a responsible AI solution:
• **Scenario 1.** On-demand inferencing within Jupyter notebook.
• **Scenario 2.** Batch inferencing on Kubernetes.
• **Scenario 3.** NVIDIA Triton inference server.

For this solution, we use the Face Detection Data Set and Benchmark (FDDB), a dataset of face regions designed for studying the problem of unconstrained face detection, combined with the PyTorch machine learning framework for implementation of FaceBoxes. This dataset contains the annotations for 5171 faces in a set of 2845 images of various resolutions. Furthermore, this technical report presents some of the solution areas and relevant use cases gathered from NetApp customers and field engineers in situations where this solution is applicable.

**Target audience**

This technical report is intended for the following audiences:

- Business leaders and enterprise architects who want to design and deploy responsible AI and address data protection and privacy issues concerning facial image processing in public spaces.
- Data scientists, data engineers, AI/machine learning (ML) researchers, and developers of AI/ML systems who aim to protect and preserve privacy.
- Enterprise architects who design data obfuscation solutions for AI/ML models and applications that comply with regulatory standards such as GDPR, CCPA, or the Privacy Act of the Department of Defense (DoD) and government organizations.
- Data scientists and AI engineers looking for efficient ways to deploy deep learning (DL) and AI/ML/DL inferencing models that protect sensitive information.
- Edge device managers and edge server administrators responsible for deployment and management of edge inferencing models.

**Solution architecture**

This solution is designed to handle real-time and batch inferencing AI workloads on large datasets by using the processing power of GPUs alongside traditional CPUs. This validation demonstrates the privacy-preserving inference for ML and optimal data management required for organizations seeking responsible AI deployments. This solution provides an architecture suited for a single or multi-node Kubernetes platform for edge and cloud computing interconnected with NetApp ONTAP AI at the core on-premises, NetApp DataOps Toolkit, and Protopia obfuscation software using Jupyter Lab and CLI interfaces. The following figure shows the logical architecture overview of data fabric powered by NetApp with DataOps Toolkit and Protopia.
Protopia obfuscation software runs seamlessly on top of the NetApp DataOps Toolkit and transforms the data before leaving the storage server.

Solution areas

Digital image processing comes with a lot of advantages, allowing many organizations to make the most of data associated with visual representations. This NetApp and Protopia solution provides a unique AI inferencing design to protect and privatize AI/ML data across the ML/DL life cycle. It enables customers to retain ownership of sensitive data, use public- or hybrid-cloud deployment models for scale and efficiency by alleviating concerns related to privacy, and deploy AI inferencing at the edge.
Environmental intelligence

There are many ways industries can take advantage of geospatial analytics in the areas of environmental hazards. Governments and the department of public works can derive actionable insights on public health and weather conditions to better advise the public during a pandemic or a natural disaster such as wildfires. For example, you can identify a COVID-positive patient in public spaces, such as airports or hospitals, without compromising the privacy of the affected individual and alert the respective authorities and the public in the vicinity for necessary safety measures.

Edge device wearables

In the military and on battlefields, you can use AI inferencing on the edge as wearable devices to track soldier health, monitor driver behavior, and alert authorities on the safety and associated risks of approaching military vehicles while preserving and protecting the privacy of soldiers. The future of the military is going high-tech with the Internet of Battlefield Things (IoBT) and the Internet of Military Things (IoMT) for wearable combat gear that help soldiers identify enemies and perform better in battle by using rapid edge computing. Protecting and preserving visual data collected from edge devices such as drones and wearable gears is crucial to keep hackers and the enemy at bay.

Noncombatant evacuation operations

Noncombatant evacuation operations (NEOs) are conducted by the DoD to assist in evacuating US citizens and nationals, DoD civilian personnel, and designated persons (host nation (HN) and third-country nationals (TCNs)) whose lives are in danger to an appropriate safe haven. The administrative controls in place use largely manual evacuee screening processes. However, the accuracy, security, and speed of evacuee identification, evacuee tracking, and threat screening could potentially be improved by using highly automated AI/ML tools combined with AI/ML video obfuscation technologies.

Healthcare and biomedical research

Image processing is used to diagnose pathologies for surgical planning from 3D images obtained from computed tomography (CT) or magnetic resonance imaging (MRI). HIPAA privacy rules govern how data must be collected, processed, and erased by organizations for all personal information and digital images like photographs. For data to qualify as sharable under the HIPAA Safe Harbor regulations, full-face photographic images and any comparable images must be removed. Automated techniques like de-identification or skull-stripping algorithms used to obscure an individual's facial features from structural CT/MR images have become an essential part of the data sharing process for biomedical research institutions.

Cloud migration of AI/ML analytics

Enterprise customers have traditionally trained and deployed AI/ML models on-premises. For economies of scale and efficiency reasons, these customers are expanding to move AI/ML functions into public, hybrid, or multi-cloud cloud deployments. However, they are bound by what data can be exposed to other infrastructures. NetApp solutions address a full range of cybersecurity threats required for data protection and security assessment and, when combined with Protopia data transformation, minimize the risks associated with migrating image processing AI/ML workloads to the cloud.

For additional use cases for edge computing and AI inferencing across other industries, see TR-4886 AI Inferencing at the Edge and the NetApp AI blog, Intelligence versus privacy.

Technology overview

This section provides an overview of the various technical components required to complete this solution.
Protopia

Protopia AI offers a unobtrusive, software-only solution for confidential inference in the market today. The Protopia solution delivers unparalleled protection for inference services by minimizing exposure of sensitive information. AI is only fed the information in the data record that is truly essential to perform the task at hand and nothing more. Most inference tasks do not use all the information that exists in every data record. Regardless of whether your AI is consuming images, voice, video, or even structured tabular data, Protopia delivers only what the inference service needs. The patented core technology uses mathematically curated noise to stochastically transform the data and garble the information that is not needed by a given ML service. This solution does not mask the data; rather, it changes the data representation by using curated random noise.

The Protopia solution formulates the problem of changing the representation as a gradient-based perturbation maximization method that still retains the pertinent information in the input feature space with respect to the functionality of the model. This discovery process is run as a fine-tuning pass at the end of training the ML model. After the pass automatically generates a set of probability distributions, a low-overhead data transformation applies noise samples from these distributions to the data, obfuscating it before passing it to the model for inferencing.

NetApp ONTAP AI

The NetApp ONTAP AI reference architecture, powered by DGX A100 systems and NetApp cloud connected storage systems, was developed and verified by NetApp and NVIDIA. It gives IT organizations an architecture that provides the following benefits:

- Eliminates design complexities
- Allows independent scaling of compute and storage
- Enables customers to start small and scale seamlessly
- Offers a range of storage options for various performance and cost points

ONTAP AI tightly integrates DGX A100 systems and NetApp AFF A800 storage systems with state-of-the-art networking. ONTAP AI simplifies AI deployments by eliminating design complexity and guesswork. Customers can start small and grow nondisruptively while intelligently managing data from the edge to the core to the cloud and back.

The following figure shows several variations in the ONTAP AI family of solutions with DGX A100 systems. AFF A800 system performance is verified with up to eight DGX A100 systems. By adding storage controller pairs to the ONTAP cluster, the architecture can scale to multiple racks to support many DGX A100 systems and petabytes of storage capacity with linear performance. This approach offers the flexibility to alter compute-to-storage ratios independently based on the size of the DL models that are used and the required performance metrics.
For additional information about ONTAP AI, see NVA-1153: NetApp ONTAP AI with NVIDIA DGX A100 Systems and Mellanox Spectrum Ethernet Switches.

**NetApp ONTAP**

ONTAP 9.11, the latest generation of storage management software from NetApp, enables businesses to modernize infrastructure and transition to a cloud-ready data center. Leveraging industry-leading data management capabilities, ONTAP enables the management and protection of data with a single set of tools, regardless of where that data resides. You can also move data freely to wherever it is needed: the edge, the core, or the cloud. ONTAP 9.11 includes numerous features that simplify data management, accelerate, and protect critical data, and enable next generation infrastructure capabilities across hybrid cloud architectures.

**NetApp DataOps Toolkit**

NetApp DataOps Toolkit is a Python library that makes it simple for developers, data scientists, DevOps engineers, and data engineers to perform various data management tasks, such as near-instantaneous provisioning of a new data volume or JupyterLab workspace, near-instantaneous cloning of a data volume or JupyterLab workspace, and near-instantaneous taking snapshots of a data volume or JupyterLab workspace for traceability or baselining. This Python library can function as either a command-line utility or a library of functions that you can import into any Python program or Jupyter notebook.
NVIDIA Triton Inference Server

NVIDIA Triton Inference Server is an open-source inference serving software that helps standardize model deployment and execution to deliver fast and scalable AI in production. Triton Inference Server streamlines AI inferencing by enabling teams to deploy, run, and scale trained AI models from any framework on any GPU- or CPU-based infrastructure. Triton Inference Server supports all major frameworks, such as TensorFlow, NVIDIA TensorRT, PyTorch, MXNet, OpenVINO, and so on. Triton integrates with Kubernetes for orchestration and scaling that you can use in all major public cloud AI and Kubernetes platforms. It’s also integrated with many MLOps software solutions.

PyTorch

PyTorch is an open-source ML framework. It is an optimized tensor library for deep learning that uses GPUs and CPUs. The PyTorch package contains data structures for multidimensional tensors that provide many utilities for efficient serializing of tensors among other useful utilities. It also has a CUDA counterpart that enables you to run your tensor computations on an NVIDIA GPU with compute capability. In this validation, we use the OpenCV-Python (cv2) library to validate our model while taking advantage of Python’s most intuitive computer vision concepts.

Simplify data management

Data management is crucial to enterprise IT operations and data scientists so that appropriate resources are used for AI applications and training AI/ML datasets. The following additional information about NetApp technologies is out of scope for this validation but might be relevant depending on your deployment.

ONTAP data management software includes the following features to streamline and simplify operations and reduce your total cost of operation:

- Inline data compaction and expanded deduplication. Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity. This applies to data stored locally and data tiered to the cloud.
- Minimum, maximum, and adaptive quality of service (AQoS). Granular quality of service (QoS) controls help maintain performance levels for critical applications in highly shared environments.
- NetApp FabricPool. Provides automatic tiering of cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID storage solution. For more information about FabricPool, see TR-4598: FabricPool best practices.

Accelerate and protect data

ONTAP delivers superior levels of performance and data protection and extends these capabilities in the following ways:

- Performance and lower latency. ONTAP offers the highest possible throughput at the lowest possible latency.
- Data protection. ONTAP provides built-in data protection capabilities with common management across all platforms.
- NetApp Volume Encryption (NVE). ONTAP offers native volume-level encryption with both onboard and External Key Management support.
- Multitenancy and multifactor authentication. ONTAP enables sharing of infrastructure resources with the highest levels of security.
**Future-proof infrastructure**

ONTAP helps meet demanding and constantly changing business needs with the following features:

- Seamless scaling and nondisruptive operations. ONTAP supports the nondisruptive addition of capacity to existing controllers and to scale-out clusters. Customers can upgrade to the latest technologies, such as NVMe and 32Gb FC, without costly data migrations or outages.

- Cloud connection. ONTAP is the most cloud-connected storage management software, with options for software-defined storage (ONTAP Select) and cloud-native instances (NetApp Cloud Volumes Service) in all public clouds.

- Integration with emerging applications. ONTAP offers enterprise-grade data services for next generation platforms and applications, such as autonomous vehicles, smart cities, and Industry 4.0, by using the same infrastructure that supports existing enterprise apps.

**NetApp Astra Control**

The NetApp Astra product family offers storage and application-aware data management services for Kubernetes applications on-premises and in the public cloud, powered by NetApp storage and data management technologies. It enables you to easily back up Kubernetes applications, migrate data to a different cluster, and instantly create working application clones. If you need to manage Kubernetes applications running in a public cloud, see the documentation for Astra Control Service. Astra Control Service is a NetApp-managed service that provides application-aware data management of Kubernetes clusters in Google Kubernetes Engine (GKE) and Azure Kubernetes Service (AKS).

**NetApp Astra Trident**

Astra Trident from NetApp is an open-source dynamic storage orchestrator for Docker and Kubernetes that simplifies the creation, management, and consumption of persistent storage. Trident, a Kubernetes-native application, runs directly within a Kubernetes cluster. Trident enables customers to seamlessly deploy DL container images onto NetApp storage and provides an enterprise-grade experience for AI container deployments. Kubernetes users (ML developers, data scientists, and so on) can create, manage, and automate orchestration and cloning to take advantage of advanced data management capabilities powered by NetApp technology.

**NetApp BlueXP Copy and Sync**

BlueXP Copy and Sync is a NetApp service for rapid and secure data synchronization. Whether you need to transfer files between on-premises NFS or SMB file shares, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, Amazon Simple Storage Service (Amazon S3), Amazon Elastic File System (Amazon EFS), Azure Blob, Google Cloud Storage, or IBM Cloud Object Storage, BlueXP Copy and Sync moves the files where you need them quickly and securely. After your data is transferred, it is fully available for use on both source and target. BlueXP Copy and Sync continuously synchronizes the data based on your predefined schedule, moving only the deltas, so that time and money spent on data replication is minimized. BlueXP Copy and Sync is a software-as-a-service (SaaS) tool that is extremely simple to set up and use. Data transfers that are triggered by BlueXP Copy and Sync are carried out by data brokers. You can deploy BlueXP Copy and Sync data brokers in AWS, Azure, Google Cloud Platform, or on-premises.

**NetApp BlueXP Classification**

Driven by powerful AI algorithms, NetApp BlueXP Classification provides automated controls and data governance across your entire data estate. You can easily pinpoint cost-savings, identify compliance and privacy concerns, and find optimization opportunities. The BlueXP Classification dashboard gives you the insight to identify duplicate data to eliminate redundancy, map personal, nonpersonal, and sensitive data and turn on alerts for sensitive data and anomalies.
Test and validation plan

For this solution design, the following three scenarios were validated:

• An inferencing task, with and without Protopia obfuscation, within a JupyterLab workspace that was orchestrated by using the NetApp DataOps Toolkit for Kubernetes.

• A batch inferencing job, with and without Protopia obfuscation, on Kubernetes with a data volume that was orchestrated by using NetApp DataOps Toolkit for Kubernetes.

• An inferencing task using an NVIDIA Triton Inference Server instance that was orchestrated by using the NetApp DataOps Toolkit for Kubernetes. We applied Protopia obfuscation to the image before invoking the Triton inference API to simulate the common requirement that any data that is transmitted over the network must be obfuscated. This workflow is applicable to use cases where data is collected within a trusted zone but must be passed outside of that trusted zone for inferencing. Without Protopia obfuscation, it is not possible to implement this type of workflow without sensitive data leaving the trusted zone.

Test configuration

The following table outlines the solution design validation environment.

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes</td>
<td>1.21.6</td>
</tr>
<tr>
<td>NetApp Astra Trident CSI Driver</td>
<td>22.01.0</td>
</tr>
<tr>
<td>NetApp DataOps Toolkit for Kubernetes</td>
<td>2.3.0</td>
</tr>
<tr>
<td>NVIDIA Triton Inference Server</td>
<td>21.11-py3</td>
</tr>
</tbody>
</table>

Test procedure

This section describes the tasks needed to complete the validation.

Prerequisites

To execute the tasks outlined in this section, you must have access to a Linux or macOS host with the following tools installed and configured:

• Kubectl (configured for access to an existing Kubernetes cluster)
  ◦ Installation and configuration instructions can be found here.

• NetApp DataOps Toolkit for Kubernetes
  ◦ Installation instructions can be found here.

Scenario 1 – On-demand inferencing in JupyterLab

1. Create a Kubernetes namespace for AI/ML inferencing workloads.

   $ kubectl create namespace inference
   namespace/inference created

2. Use the NetApp DataOps Toolkit to provision a persistent volume for storing the data on which you will
perform the inferencing.

```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
-name=inference-data --size=50Gi
Creating PersistentVolumeClaim (PVC) 'inference-data' in namespace
'inference'.
PersistentVolumeClaim (PVC) 'inference-data' created. Waiting for
Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC)
'inference-data' in namespace 'inference'.
```

3. Use the NetApp DataOps Toolkit to create a new JupyterLab workspace. Mount the persistent volume that
was created in the previous step by using the --mount-pvc option. Allocate NVIDIA GPUs to the
workspace as necessary by using the --nvidia-gpu option.

In the following example, the persistent volume inference-data is mounted to the JupyterLab
workspace container at /home/jovyan/data. When using official Project Jupyter container images,
/home/jovyan is presented as the top-level directory within the JupyterLab web interface.

```
$ netapp_dataops_k8s_cli.py create jupyterlab --namespace=inference
--workspace-name=live-inference --size=50Gi --nvidia-gpu=2 --mount
-pvc=inference-data:/home/jovyan/data
Set workspace password (this password will be required in order to
access the workspace):
Re-enter password:
Creating persistent volume for workspace...
Creating PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-live-inference' in namespace 'inference'.
PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-live-inference' created. Waiting for Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC)
'ntap-dsutil-jupyterlab-live-inference' in namespace 'inference'.
Creating Service 'ntap-dsutil-jupyterlab-live-inference' in namespace
'inference'.
Service successfully created.
Attaching Additional PVC: 'inference-data' at mount_path:
'/home/jovyan/data'.
Creating Deployment 'ntap-dsutil-jupyterlab-live-inference' in namespace
'inference'.
Deployment 'ntap-dsutil-jupyterlab-live-inference' created.
Waiting for Deployment 'ntap-dsutil-jupyterlab-live-inference' to reach
Ready state.
Deployment successfully created.
Workspace successfully created.
To access workspace, navigate to http://192.168.0.152:32721
```
4. Access the JupyterLab workspace by using the URL specified in the output of the `create jupyterlab` command. The data directory represents the persistent volume that was mounted to the workspace.

5. Open the data directory and upload the files on which the inferencing is to be performed. When files are uploaded to the data directory, they are automatically stored on the persistent volume that was mounted to the workspace. To upload files, click the Upload Files icon, as shown in the following image.
6. Return to the top-level directory and create a new notebook.

7. Add inferencing code to the notebook. The following example shows inferencing code for an image detection use case.
8. Add Protopia obfuscation to your inferencing code. Protopia works directly with customers to provide use-case specific documentation and is outside of the scope of this technical report. The following example shows inferencing code for an image detection use case with Protopia obfuscation added.
STEP 3-2: Protopia AI (With obfuscation) detection

```python
# get current frame
frame = input_image

# preprocess input
preprocessed_input = preprocess_input(frame)
preprocessed_input = torch.tensor(preprocessed_input).to(device)

# run forward pass
net_noisy_activation = noisy_model.forward_head(preprocessed_input)  # run the first few layers
net_noisy_activation, noisy_activations = noisy_model.forward_tail(noisy_activation)  # run rest of the layers

# postprocess output
noisy_pred = torch.argmax(noisy_activations, dim=1)
noisy_outputs = postprocess_outputs(noisy_pred, [[input_image_width, input_image_height]], priors, THRESHOLD = 0.5)

# get reconstruction of the noisy activation
noisy_reconstruction = decoder_function(noisy_activations)
noisy_reconstruction = noisy_reconstruction.detach().cpu().numpy()
noisy_reconstruction = unpreprocess_output(noisy_reconstruction, (input_image_width, input_image_height), True)

# draw rectangles
for (x1, y1, x2, y2, s) in noisy_outputs:
    cx, cy = int((x1 + x2) / 2), int((y1 + y2) / 2)
    cv2.rectangle(noisy_reconstruction, (x1, y1), (x2, y2), (0, 0, 255), 4)
```

Visualize Protopia AI (With obfuscation) detection

```python
show_cv2_image(noisy_reconstruction, scale=2)
```
Scenario 2 – Batch inferencing on Kubernetes

1. Create a Kubernetes namespace for AI/ML inferencing workloads.

```
$ kubectl create namespace inference
namespace/inference created
```

2. Use the NetApp DataOps Toolkit to provision a persistent volume for storing the data on which you will perform the inferencing.

```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc -name=inference-data --size=50Gi
Creating PersistentVolumeClaim (PVC) 'inference-data' in namespace 'inference'.
PersistentVolumeClaim (PVC) 'inference-data' created. Waiting for Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC) 'inference-data' in namespace 'inference'.
```

3. Populate the new persistent volume with the data on which you will perform the inferencing.

There are several methods for loading data onto a PVC. If your data is currently stored in an S3-compatible object storage platform, such as NetApp StorageGRID or Amazon S3, then you can use NetApp DataOps Toolkit S3 Data Mover capabilities. Another simple method is to create a JupyterLab workspace and then upload files through the JupyterLab web interface, as outlined in Steps 3 to 5 in the section "Scenario 1 – On-demand inferencing in JupyterLab."

4. Create a Kubernetes job for your batch inferencing task. The following example shows a batch inferencing job for an image detection use case. This job performs inferencing on each image in a set of images and writes inferencing accuracy metrics to stdout.

```bash
$ netapp_dataops_k8s.cli.py create job --namespace=inference --pvc
Creating Job 'image-detection' in namespace 'inference'.
Job 'image-detection' created. Waiting for Kubernetes to submit job to its queue.
Job submitted to Kubernetes queue. Job 'image-detection' in namespace 'inference'.
```
5. Confirm that the inferencing job completed successfully.
6. Add Protopia obfuscation to your inferencing job. You can find use case-specific instructions for adding Protopia obfuscation directly from Protopia, which is outside of the scope of this technical report. The following example shows a batch inferencing job for a face detection use case with Protopia obfuscation added by using an ALPHA value of 0.8. This job applies Protopia obfuscation before performing inferencing for each image in a set of images and then writes inferencing accuracy metrics to stdout.

We repeated this step for ALPHA values 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 0.9, and 0.95. You can see the results in "Inferencing accuracy comparison."
$ vi inference-job-protopia-0.8.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-inference-protopia-0.8
  namespace: inference
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: inference-data
      - name: dshm
        emptyDir:
          medium: Memory
      containers:
      - name: inference
        image: netapp-protopia-inference:latest
        imagePullPolicy: IfNotPresent
        env:
        - name: ALPHA
          value: "0.8"
        command: ["python3", "run-accuracy-measurement.py", "--dataset", "/data/netapp-face-detection/FDDB", "--alpha", "$\{\text{ALPHA}\}$", "--noisy"]
        resources:
        limits:
          nvidia.com/gpu: 2
      volumeMounts:
      - mountPath: /data
        name: data
      - mountPath: /dev/shm
        name: dshm
      restartPolicy: Never
$ kubectl create -f inference-job-protopia-0.8.yaml
job.batch/netapp-inference-protopia-0.8 created

7. Confirm that the inferencing job completed successfully.
$ kubectl -n inference logs netapp-inference-protopia-0.8-b4dkz
100% |██████████| 89/89 [01:05<00:00, 1.37it/s]
Reading Predictions: 100% |██████████| 10/10 [00:02<00:00, 3.67it/s]
Predicting ...: 100% |██████████| 10/10 [00:22<00:00, 2.24s/it]

Results

FDDB-fold-1 Val AP: 0.8953066115834589
FDDB-fold-2 Val AP: 0.8819580264029936
FDDB-fold-3 Val AP: 0.8781107458462862
FDDB-fold-4 Val AP: 0.9085731346308461
FDDB-fold-5 Val AP: 0.9166445508275378
FDDB-fold-6 Val AP: 0.910178994188819
FDDB-fold-7 Val AP: 0.8383443678423771
FDDB-fold-8 Val AP: 0.8476311547659464
FDDB-fold-9 Val AP: 0.8739624502111121
FDDB-fold-10 Val AP: 0.8905468076424851
FDDB Dataset Average AP: 0.8841195749171925

mAP: 0.8841195749171925

Scenario 3 – NVIDIA Triton Inference Server

1. Create a Kubernetes namespace for AI/ML inferencing workloads.

   $ kubectl create namespace inference
   namespace/inference created

2. Use the NetApp DataOps Toolkit to provision a persistent volume to use as a model repository for the NVIDIA Triton Inference Server.

   $ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
   -name=triton-model-repo --size=100Gi
   Creating PersistentVolumeClaim (PVC) 'triton-model-repo' in namespace 'inference'.
   PersistentVolumeClaim (PVC) 'triton-model-repo' created. Waiting for Kubernetes to bind volume to PVC.
   Volume successfully created and bound to PersistentVolumeClaim (PVC) 'triton-model-repo' in namespace 'inference'.

3. Store your model on the new persistent volume in a format that is recognized by the NVIDIA Triton Inference Server.

   There are several methods for loading data onto a PVC. A simple method is to create a JupyterLab workspace and then upload files through the JupyterLab web interface, as outlined in steps 3 to 5 in “Scenario 1 – On-demand inferencing in JupyterLab.”
4. Use NetApp DataOps Toolkit to deploy a new NVIDIA Triton Inference Server instance.

```
$ netapp_dataops_k8s_cli.py create triton-server --namespace=inference
--server-name=netapp-inference --model-repo-pvc-name=triton-model-repo
Creating Service 'ntap-dsutil-triton-netapp-inference' in namespace 'inference'.
Service successfully created.
Creating Deployment 'ntap-dsutil-triton-netapp-inference' in namespace 'inference'.
Deployment 'ntap-dsutil-triton-netapp-inference' created.
Waiting for Deployment 'ntap-dsutil-triton-netapp-inference' to reach Ready state.
Deployment successfully created.
Server successfully created.
Server endpoints:
http: 192.168.0.152: 31208
grpc: 192.168.0.152: 32736
metrics: 192.168.0.152: 30009/metrics
```

5. Use a Triton client SDK to perform an inferencing task. The following Python code excerpt uses the Triton Python client SDK to perform an inferencing task for a face detection use case. This example calls the Triton API and passes in an image for inferencing. The Triton Inference Server then receives the request, invokes the model, and returns the inferencing output as part of the API results.

```
# get current frame
frame = input_image
# preprocess input
preprocessed_input = preprocess_input(frame)
preprocessed_input = torch.Tensor(preprocessed_input).to(device)
# run forward pass
clean_activation = clean_model_head(preprocessed_input)  # runs the first few layers
#########################################################################
#          pass clean image to Triton Inference Server API for inferencing  #
#########################################################################
triton_client = httpclient.InferenceServerClient(url="192.168.0.152:31208",
verbose=False)
model_name = "face_detection_base"
inputs = []
outputs = []
inputs.append(httpclient.InferInput("INPUT__0", [1, 128, 32, 32],
```
inputs[0].set_data_from_numpy(clean_activation.detach().cpu().numpy(), binary_data=False)
outputs.append(httpclient.InferRequestedOutput("OUTPUT__0", binary_data=False))
outputs.append(httpclient.InferRequestedOutput("OUTPUT__1", binary_data=False))
results = triton_client.infer(model_name, inputs, outputs=outputs, #query_params=query_params, headers=None, request_compression_algorithm=None, response_compression_algorithm=None)
# print(results.get_response())
statistics = triton_client.get_inference_statistics(model_name=model_name, headers=None)
print(statistics)
if len(statistics["model_stats"]) != 1:
    print("FAILED: Inference Statistics")
sys.exit(1)
loc_numpy = results.as_numpy("OUTPUT__0")
pred_numpy = results.as_numpy("OUTPUT__1")
# postprocess output
clean_pred = (loc_numpy, pred_numpy)
clean_outputs = postprocess_outputs(clean_pred, [[input_image_width, input_image_height]], priors, THRESHOLD)
# draw rectangles
clean_frame = copy.deepcopy(frame)  # needs to be deep copy
for (x1, y1, x2, y2, s) in clean_outputs[0]:
    x1, y1 = int(x1), int(y1)
    x2, y2 = int(x2), int(y2)
    cv2.rectangle(clean_frame, (x1, y1), (x2, y2), (0, 0, 255), 4)

6. Add Protopia obfuscation to your inferencing code. You can find use case-specific instructions for adding Protopia obfuscation directly from Protopia; however, this process is outside the scope of this technical report. The following example shows the same Python code that is shown in the preceding step 5, but with Protopia obfuscation added.

Note that the Protopia obfuscation is applied to the image before it is passed to the Triton API. Thus, the
non-obfuscated image never leaves the local machine. Only the obfuscated image is passed across the network. This workflow is applicable to use cases in which data is collected within a trusted zone but then needs to be passed outside of that trusted zone for inferencing. Without Protopia obfuscation, it is not possible to implement this type of workflow without sensitive data ever leaving the trusted zone.

```python
# get current frame
frame = input_image

# preprocess input
preprocessed_input = preprocess_input(frame)
preprocessed_input = torch.Tensor(preprocessed_input).to(device)

# run forward pass
not_noisy_activation = noisy_model_head(preprocessed_input)  # runs the first few layers

# obfuscate image locally prior to inferencing
# SINGLE ADITIONAL LINE FOR PRIVATE INFERENCE
noisy_activation = noisy_model_noise(not_noisy_activation)

# pass obfuscated image to Triton Inference Server API for inferencing
triton_client = httpclient.InferenceServerClient(url="192.168.0.152:31208",
verbose=False)
model_name = "face_detection_noisy"
inputs = []
outputs = []
inputs.append(httpclient.InferInput("INPUT__0", [1, 128, 32, 32], "FP32"))
inputs[0].set_data_from_numpy(noisy_activation.detach().cpu().numpy(),
binary_data=False)
outputs.append(httpclient.InferRequestedOutput("OUTPUT__0",
binary_data=False))
outputs.append(httpclient.InferRequestedOutput("OUTPUT__1",
binary_data=False))
results = triton_client.infer(
    model_name,
    inputs,
    outputs=outputs,
    #query_params=query_params,
    headers=None,
    request_compression_algorithm=None,
)```
Inferencing accuracy comparison

For this validation, we performed inferencing for an image detection use case by using a set of raw images. We then performed the same inferencing task on the same set of images with Protopia obfuscation added before inferencing. We repeated the task using different values of ALPHA for the Protopia obfuscation component. In the context of Protopia obfuscation, the ALPHA value represents the amount of obfuscation that is applied, with a higher ALPHA value representing a higher level of obfuscation. We then compared inferencing accuracy across these different runs.

The following two tables provide details about our use case and outline the results.
Protopia works directly with customers to determine the appropriate ALPHA value for a specific use case.

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>FaceBoxes (PyTorch) -</td>
</tr>
<tr>
<td>Dataset</td>
<td>FDDB dataset</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protopia obfuscation</th>
<th>ALPHA</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>N/A</td>
<td>0.9337148153739079</td>
</tr>
<tr>
<td>Yes</td>
<td>0.05</td>
<td>0.9028766627325002</td>
</tr>
<tr>
<td>Yes</td>
<td>0.1</td>
<td>0.9024301009661478</td>
</tr>
<tr>
<td>Yes</td>
<td>0.2</td>
<td>0.9081836283186224</td>
</tr>
<tr>
<td>Yes</td>
<td>0.4</td>
<td>0.9073066107482036</td>
</tr>
<tr>
<td>Yes</td>
<td>0.6</td>
<td>0.8847816568680239</td>
</tr>
<tr>
<td>Yes</td>
<td>0.8</td>
<td>0.8841195749171925</td>
</tr>
<tr>
<td>Yes</td>
<td>0.9</td>
<td>0.8455427675252052</td>
</tr>
<tr>
<td>Yes</td>
<td>0.95</td>
<td>0.8455427675252052</td>
</tr>
</tbody>
</table>

**Obfuscation speed**

For this validation, we applied Protopia obfuscation to a 1920 x 1080 pixel image five times and measured the amount of time that it took for the obfuscation step to complete each time.

We used PyTorch running on a single NVIDIA V100 GPU to apply the obfuscation, and we cleared the GPU cache between runs. The obfuscation step took 5.47ms, 5.27ms, 4.54ms, 5.24ms, and 4.84ms respectively to complete across the five runs. The average speed was 5.072ms.

**Conclusion**

Data exists in three states: at rest, in transit, and in compute. An important part of any AI inferencing service should be the protection of data from threats during the entire process. Protecting data during inferencing is critical because the process can expose private information about both external customers and the business providing the inferencing service. Protopia AI is a nonobtrusive software-only solution for confidential AI inferencing in today’s market. With Protopia, AI is fed only the transformed information in the data records that is essential to carrying out the AI/ML task at hand and nothing more. This stochastic transformation is not a form of masking and is based on mathematically changing the representation of the data by using curated noise.

NetApp storage systems with ONTAP capabilities deliver the same or better performance as local SSD storage and, combined with the NetApp DataOps Toolkit, offer the following benefits to data scientists, data engineers, AI/ML developers, and business or enterprise IT decision makers:

- Effortless sharing of data between AI systems, analytics, and other critical business systems. This data
sharing reduces infrastructure overhead, improves performance, and streamlines data management across the enterprise.

- Independently scalable compute and storage to minimize costs and improve resource usage.
- Streamlined development and deployment workflows using integrated Snapshot copies and clones for instantaneous and space-efficient user workspaces, integrated version control, and automated deployment.
- Enterprise-grade data protection and data governance for disaster recovery, business continuity, and regulatory requirements.
- Simplified invocation of data management operations; rapidly take Snapshot copies of data scientist workspaces for backup and traceability from the NetApp DataOps Toolkit in Jupyter notebooks.

The NetApp and Protopia solution provides a flexible, scale-out architecture that is ideal for enterprise-grade AI inference deployments. It enables data protection and provides privacy for sensitive information where confidential AI inferencing requirements can be met with responsible AI practices in both on-premises and hybrid cloud deployments.

**Where to find additional information and acknowledgements**

To learn more about the information described in this document, refer to the following documents and/or websites:

- NetApp ONTAP data management software — ONTAP information library
  
- NetApp Persistent Storage for Containers—NetApp Trident
  
  [https://netapp.io/persistent-storage-provisioner-for-kubernetes/](https://netapp.io/persistent-storage-provisioner-for-kubernetes/)
- NetApp DataOps Toolkit
  
- NetApp Persistent Storage for Containers—NetApp Astra Trident
  
  [https://netapp.io/persistent-storage-provisioner-for-kubernetes/](https://netapp.io/persistent-storage-provisioner-for-kubernetes/)
- Protopia AI—Confidential Inference
  
- NetApp BlueXP Copy and Sync
  
  [https://docs.netapp.com/us-en/occm/concept_cloud_sync.html#how-cloud-sync-works](https://docs.netapp.com/us-en/occm/concept_cloud_sync.html#how-cloud-sync-works)
- NVIDIA Triton Inference Server
  
- NVIDIA Triton Inference Server Documentation
  
- FaceBoxes in PyTorch
Acknowledgments

- Mark Cates, Principal Product Manager, NetApp
- Sufian Ahmad, Technical Marketing Engineer, NetApp
- Hadi Esmaeilzadeh, Chief Technology Officer and Professor, Protopia AI

Sentiment analysis with NetApp AI

TR-4910: Sentiment Analysis from Customer Communications with NetApp AI

Rick Huang, Sathish Thyagarajan, and David Arnette, NetApp
Diego Sosa-Coba, SFL Scientific

This technical report provides design guidance for customers to perform sentiment analysis in an enterprise-level global support center by using NetApp data management technologies with an NVIDIA software framework using transfer learning and conversational AI. This solution is applicable to any industry wanting to gain customer insights from recorded speech or text files representing chat logs, emails, and other text or audio communications. We implemented an end-to-end pipeline to demonstrate automatic speech recognition, real-time sentiment analysis, and deep-learning natural-language processing model-retraining capabilities on a GPU-accelerated compute cluster with NetApp cloud-connected all flash storage. Massive, state-of-the-art language models can be trained and optimized to perform inference rapidly with the global support center to create an exceptional customer experience and objective, long-term employee performance evaluations.

Sentiment analysis is a field of study within Natural Language Processing (NLP) by which positive, negative, or neutral sentiments are extracted from text. Conversational AI systems have risen to a near global level of integration as more and more people come to interact with them. Sentiment analysis has a variety of use cases, from determining support center employee performance in conversations with callers and providing appropriate automated chatbot responses to predicting a firm’s stock price based on the interactions between firm representatives and the audience at quarterly earnings calls. Furthermore, sentiment analysis can be used to determine the customer’s view on the products, services, or support provided by the brand.

This end-to-end solution uses NLP models to perform high level sentiment analysis that enables support-center analytical frameworks. Audio recordings are processed into written text, and sentiment is extracted from each sentence in the conversation. Results, aggregated into a dashboard, can be crafted to analyze conversation sentiments, both historically and in real-time. This solution can be generalized to other solutions with similar data modalities and output needs. With the appropriate data, other use cases can be accomplished. For example, company earnings calls can be analyzed for sentiment using the same end-to-end pipeline. Other forms of NLP analyses, such as topic modeling and named entity recognition (NER), are also possible due to the flexible nature of the pipeline.

These AI implementations were made possible by NVIDIA RIVA, the NVIDIA TAO Toolkit, and the NetApp DataOps Toolkit working together. NVIDIA’s tools are used to rapidly deploy highly performant AI solutions using prebuilt models and pipelines. The NetApp DataOps Toolkit simplifies various data management tasks to speed up development.
Customer value

Businesses see value from an employee-assessment and customer-reaction tool for text, audio, and video conversation for sentiment analysis. Managers benefit from the information presented in the dashboard, allowing for an assessment of the employees and customer satisfaction based on both sides of the conversation.

Additionally, the NetApp DataOps Toolkit manages the versioning and allocation of data within the customer’s infrastructure. This leads to frequent updates of the analytics presented within the dashboard without creating unwieldy data storage costs.

Use cases

Due to the number of calls that these support centers process, assessment of call performance could take significant time if performed manually. Traditional methods, like bag-of-words counting and other methods, can achieve some automation, but these methods do not capture more nuanced aspects and semantic context of dynamic language. AI modeling techniques can be used to perform some of these more nuanced analyses in an automated manner. Furthermore, with the current state of the art, pretrained modeling tools published by NVIDIA, AWS, Google, and others, an end-to-end pipeline with complex models can be now stood up and customized with relative ease.

An end-to-end pipeline for support center sentiment analysis ingests audio files in real time as employees converse with callers. Then, these audio files are processed for use in the speech-to-text component which converts them into a text format. Each sentence in the conversation receives a label indicating the sentiment (positive, negative, or neutral).

Sentiment analysis can provide an essential aspect of the conversations for assessment of call performance. These sentiments add an additional level of depth to the interactions between employees and callers. The AI-assisted sentiment dashboard provides managers with a real-time tracking of sentiment within a conversation, along with a retrospective analysis of the employee’s past calls.

There are prebuilt tools that can be combined in powerful ways to quickly create an end-to-end AI pipeline to solve this problem. In this case, the NVIDIA RIVA library can be used to perform the two in-series tasks: audio transcription and sentiment analysis. The first is a supervised learning signal processing algorithm and the second is a supervised learning NLP classification algorithm. These out-of-the-box algorithms can be fine-tuned for any relevant use case with business-relevant data using the NVIDIA TAO Toolkit. This leads to more accurate and powerful solutions being built for only a fraction of the cost and resources. Customers can incorporate the NVIDIA Maxine framework for GPU-accelerated video conferencing applications in their support center design.

The following use cases are at the core of this solution. Both use cases use the TAO Toolkit for model fine-tuning and RIVA for model deployment.

• Speech-to-text
• Sentiment analysis

To analyze support center interactions between employees and customers, each customer conversation in the form of audio calls can be run through the pipeline to extract sentence-level sentiments. Those sentiments can then be verified by a human to justify the sentiments or adjust them as needed. The labeled data is then passed onto the fine-tuning step to improve sentiment predictions. If labeled sentiment data already exists, then model fine-tuning can be expedited. In either case, the pipeline is generalizable to other solutions that require the ingestion of audio and the classification of sentences.
AI sentiment outputs are either uploaded to an external cloud database or to a company-managed storage system. The sentiment outputs are transferred from this larger database into local storage for use within the dashboard that displays the sentiment analysis for managers. The dashboard’s primary functionality is to interface with the customer service employee in real time. Managers can assess and provide feedback on employees during their calls with live updates of the sentiment of each sentence, as well as an historic review of the employee’s past performance or customer reactions.

The NetApp DataOps Toolkit can continue to manage data storage systems even after the RIVA inference pipeline generates sentiment labels. Those AI results can be uploaded to a data storage system managed by the NetApp DataOps Toolkit. The data storage systems must be capable of managing hundreds of inserts and selects every minute. The local device storage system queries the larger data storage in real-time for extraction. The larger data storage instance can also be queried for historical data to further enhance the dashboard experience. The NetApp DataOps Toolkit facilitates both these uses by rapidly cloning data and distributing it across all the dashboards that use it.
Target Audience

The target audience for the solution includes the following groups:

- Employee managers
- Data engineers/data scientists
- IT administrators (on-premises, cloud, or hybrid)

Tracking sentiments throughout conversations is a valuable tool for assessing employee performance. Using the AI-dashboard, managers can see how employees and callers change their feelings in real time, allowing for live assessments and guidance sessions. Moreover, businesses can gain valuable customer insights from customers engaged in vocal conversations, text chatbots, and video conferencing. Such customer analytics uses the capabilities of multimodal processing at scale with modern, state-of-the-art AI models and workflows.

On the data side, a large number of audio files are processed daily by the support center. The NetApp DataOps Toolkit facilitates this data handling task for both the periodic fine-tuning of models and sentiment analysis dashboards.

IT administrators also benefit from the NetApp DataOps Toolkit as it allows them to move data quickly between deployment and production environments. The NVIDIA environments and servers must also be managed and distributed to allow for real time inference.

Architecture

The architecture of this support center solution revolves around NVIDIA’s prebuilt tools and the NetApp DataOps Toolkit. NVIDIA’s tools are used to rapidly deploy high-performance AI-solutions using prebuilt models and pipelines. The NetApp DataOps Toolkit simplifies various data management tasks to speed up development.

Solution technology

NVIDIA RIVA is a GPU-accelerated SDK for building multimodal conversational AI applications that deliver real-time performance on GPUs. The NVIDIA Train, Adapt, and Optimize (TAO) Toolkit provides a faster, easier way to accelerate training and quickly create highly accurate and performant, domain-specific AI models.

The NetApp DataOps Toolkit is a Python library that makes it simple for developers, data scientists, DevOps engineers, and data engineers to perform various data management tasks. This includes near-instantaneous provisioning of a new data volume or JupyterLab workspace, near-instantaneous cloning of a data volume or JupyterLab workspace, and near-instantaneous snapshotting of a data volume or JupyterLab workspace for traceability and baselining.

Architectural Diagram

The following diagram shows the solution architecture. There are three main environment categories: the cloud, the core, and the edge. Each of the categories can be geographically dispersed. For example, the cloud contains object stores with audio files in buckets in different regions, whereas the core might contain datacenters linked via a high-speed network or NetApp BlueXP Copy and Sync. The edge nodes denote the individual human agent’s daily working platforms, where interactive dashboard tools and microphones are available to visualize sentiment and collect audio data from conversations with customers.

In GPU-accelerated datacenters, businesses can use the NVIDIA RIVA framework to build conversational AI applications, to which the Tao Toolkit connects for model finetuning and retraining using transfer L-learning techniques. These compute applications and workflows are powered by the NetApp DataOps Toolkit, enabling the best data management capabilities ONTAP has to offer. The toolkit allows corporate data teams to rapidly
prototype their models with associated structured and unstructured data via snapshots and clones for traceability, versioning, A/B testing, thus providing security, governance, and regulatory compliance. See the section "Storage Design" for more details.

This solution demonstrates the audio file processing, NLP model training, transfer learning, and data management detail steps. The resulting end-to-end pipeline generates a sentiment summary that displays in real-time on human support agents’ dashboards.

### Hardware requirements

The following table lists the hardware components that are required to implement the solution. The hardware components that are used in any particular implementation of the solution might vary based on customer requirements.

<table>
<thead>
<tr>
<th>Response latency tests</th>
<th>Time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data processing</td>
<td>10</td>
</tr>
<tr>
<td>Inferencing</td>
<td>10</td>
</tr>
</tbody>
</table>

These response-time tests were run on 50,000+ audio files across 560 conversations. Each audio file was ~100KB in size as an MP3 and ~1 MB when converted to WAV. The data processing step converts MP3s into WAV files. The inference steps convert the audio files into text and extract a sentiment from the text. These steps are all independent of one another and can be parallelized to speed up the process.

Taking into account the latency of transferring data between stores, managers should be able to see updates to the real time sentiment analysis within a second of the end of the sentence.
### NVIDIA RIVA hardware

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux x86_64</td>
</tr>
<tr>
<td>GPU memory (ASR)</td>
<td>Streaming models: ~5600 MB</td>
</tr>
<tr>
<td></td>
<td>Non-streaming models: ~3100 MB</td>
</tr>
<tr>
<td>GPU memory (NLP)</td>
<td>~500MB per BERT model</td>
</tr>
</tbody>
</table>

### NVIDIA TAO Toolkit hardware

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>System RAM</td>
<td>32GB</td>
</tr>
<tr>
<td>GPU RAM</td>
<td>32GB</td>
</tr>
<tr>
<td>CPU</td>
<td>8 core</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA (A100, V100 and RTX 30x0)</td>
</tr>
<tr>
<td>SSD</td>
<td>100GB</td>
</tr>
</tbody>
</table>

### Flash storage system

**NetApp ONTAP 9**

ONTAP 9.9, the latest generation of storage management software from NetApp, enables businesses to modernize infrastructure and transition to a cloud-ready data center. Leveraging industry-leading data management capabilities, ONTAP enables the management and protection of data with a single set of tools, regardless of where that data resides. You can also move data freely to wherever it is needed: the edge, the core, or the cloud. ONTAP 9.9 includes numerous features that simplify data management, accelerate, and protect critical data, and enable next generation infrastructure capabilities across hybrid cloud architectures.

**NetApp BlueXP Copy and Sync**

BlueXP Copy and Sync is a NetApp service for rapid and secure data synchronization that allows you to transfer files between on-premises NFS or SMB file shares to any of the following targets:

- NetApp StorageGRID
- NetApp ONTAP S3
- NetApp Cloud Volumes Service
- Azure NetApp Files
- Amazon Simple Storage Service (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Azure Blob
- Google Cloud Storage
- IBM Cloud Object Storage

BlueXP Copy and Sync moves the files where you need them quickly and securely. After your data is transferred, it is fully available for use on both the source and the target. BlueXP Copy and Sync continuously
synchronizes the data, based on your predefined schedule, moving only the deltas, so that time and money spent on data replication is minimized. BlueXP Copy and Sync is a software as a service (SaaS) tool that is simple to set up and use. Data transfers that are triggered by BlueXP Copy and Sync are carried out by data brokers. You can deploy BlueXP Copy and Sync data brokers in AWS, Azure, Google Cloud Platform, or on-premises.

NetApp StorageGRID

The StorageGRID software-defined object storage suite supports a wide range of use cases across public, private, and hybrid multi-cloud environments seamlessly. With industry leading innovations, NetApp StorageGRID stores, secures, protect, and preserves unstructured data for multi-purpose use including automated lifecycle management for long periods of time. For more information, see the NetApp StorageGRID site.

Software requirements

The following table lists the software components that are required to implement this solution. The software components that are used in any particular implementation of the solution might vary based on customer requirements.

<table>
<thead>
<tr>
<th>Host machine</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIVA (formerly JARVIS)</td>
<td>1.4.0</td>
</tr>
<tr>
<td>TAO Toolkit (formerly Transfer Learning Toolkit)</td>
<td>3.0</td>
</tr>
<tr>
<td>ONTAP</td>
<td>9.9.1</td>
</tr>
<tr>
<td>DGX OS</td>
<td>5.1</td>
</tr>
<tr>
<td>DOTK</td>
<td>2.0.0</td>
</tr>
</tbody>
</table>

**NVIDIA RIVA Software**

<table>
<thead>
<tr>
<th>Software</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker</td>
<td>&gt;19.02 (with nvidia-docker installed)&gt;19.03 if not using DGX</td>
</tr>
<tr>
<td>NVIDIA Driver</td>
<td>465.19.01+, 418.40+, 440.33+, 450.51+, 460.27+ for Data Center GPUs</td>
</tr>
<tr>
<td>Container OS</td>
<td>Ubuntu 20.04</td>
</tr>
<tr>
<td>CUDA</td>
<td>11.3.0</td>
</tr>
<tr>
<td>cuBLAS</td>
<td>11.5.1.101</td>
</tr>
<tr>
<td>cuDNN</td>
<td>8.2.0.41</td>
</tr>
<tr>
<td>NCCL</td>
<td>2.9.6</td>
</tr>
<tr>
<td>TensorRT</td>
<td>7.2.3.4</td>
</tr>
<tr>
<td>Triton Inference Server</td>
<td>2.9.0</td>
</tr>
</tbody>
</table>
NVIDIA TAO Toolkit software

<table>
<thead>
<tr>
<th>Software</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu 18.04 LTS</td>
<td>18.04</td>
</tr>
<tr>
<td>python</td>
<td>&gt;=3.6.9</td>
</tr>
<tr>
<td>docker-ce</td>
<td>&gt;19.03.5</td>
</tr>
<tr>
<td>docker-API</td>
<td>1.40</td>
</tr>
<tr>
<td>nvidia-container-toolkit</td>
<td>&gt;1.3.0-1</td>
</tr>
<tr>
<td>nvidia-container-runtime</td>
<td>3.4.0-1</td>
</tr>
<tr>
<td>nvidia-docker2</td>
<td>2.5.0-1</td>
</tr>
<tr>
<td>nvidia-driver</td>
<td>&gt;455</td>
</tr>
<tr>
<td>python-pip</td>
<td>&gt;21.06</td>
</tr>
<tr>
<td>nvidia-pyindex</td>
<td>Latest version</td>
</tr>
</tbody>
</table>

Use case details

This solution applies to the following use cases:

- Speech-to-text
- Sentiment analysis
The speech-to-text use case begins by ingesting audio files for the support centers. This audio is then processed to fit the structure required by RIVA. If the audio files have not already been split into their units of analysis, then this must be done before passing the audio to RIVA. After the audio file is processed, it is passed to the RIVA server as an API call. The server employs one of the many models it is hosting and returns a response. This speech-to-text (part of Automatic Speech Recognition) returns a text representation of the audio. From there, the pipeline switches over to the sentiment analysis portion.

For sentiment analysis, the text output from the Automatic Speech Recognition serves as the input to the Text Classification. Text Classification is the NVIDIA component for classifying text to any number of categories. The sentiment categories range from positive to negative for the support center conversations. The performance of the models can be assessed using a holdout set to determine the success of the fine-tuning step.
A similar pipeline is used for both the speech-to-text and sentiment analysis within the TAO Toolkit. The major difference is the use of labels which are required for the fine-tuning of the models. The TAO Toolkit pipeline begins with the processing of the data files. Then the pretrained models (coming from the NVIDIA NGC Catalog) are fine-tuned using the support center data. The fine-tuned models are evaluated based on their corresponding performance metrics and, if they are more performant than the pretrained models, are deployed to the RIVA server.

Design considerations

This section describes the design considerations for the different components of this solution.

Network and compute design

Depending on the restrictions on data security, all data must remain within the customer’s infrastructure or a secure environment.
Storage design

The NetApp DataOps Toolkit serves as the primary service for managing storage systems. The DataOps Toolkit is a Python library that makes it simple for developers, data scientists, DevOps engineers, and data engineers to perform various data management tasks, such as near-instantaneous provisioning of a new data volume or JupyterLab workspace, near-instantaneous cloning of a data volume or JupyterLab workspace, and near-instantaneous snapshotting of a data volume or JupyterLab workspace for traceability or baselining. This Python library can function as either a command line utility or a library of functions that can be imported into any Python program or Jupyter Notebook.

RIVA best practices

NVIDIA provides several general best data practices for using RIVA:

- **Use lossless audio formats if possible.** The use of lossy codecs such as MP3 can reduce quality.
- **Augment training data.** Adding background noise to audio training data can initially decrease accuracy and yet increase robustness.
- **Limit vocabulary size if using scraped text.** Many online sources contain typos or ancillary pronouns and uncommon words. Removing these can improve the language model.
- **Use a minimum sampling rate of 16kHz if possible.** However, try not to resample, because doing so decreases audio quality.

In addition to these best practices, customers must prioritize gathering a representative sample dataset with accurate labels for each step of the pipeline. In other words, the sample dataset should proportionally reflect specified characteristics exemplified in a target dataset. Similarly, the dataset annotators have a responsibility to balance accuracy and the speed of labeling so that the quality and quantity of the data are both maximized. For example, this support center solution requires audio files, labeled text, and sentiment labels. The sequential nature of this solution means that errors from the beginning of the pipeline are propagated all the way through to the end. If the audio files are of poor quality, the text transcriptions and sentiment labels will be as well.

This error propagation similarly applies to the models trained on this data. If the sentiment predictions are 100% accurate but the speech-to-text model performs poorly, then the final pipeline is limited by the initial audio- to- text transcriptions. It is essential that developers consider each model’s performance individually and as a component of a larger pipeline. In this particular case, the end goal is to develop a pipeline that can accurately predict the sentiment. Therefore, the overall metric on which to assess the pipeline is the accuracy of the sentiments, which the speech-to-text transcription directly affects.
The NetApp DataOps Toolkit complements the data quality-checking pipeline through the use of its near-instantaneous data cloning technology. Each labeled file must be assessed and compared to the existing labeled files. Distributing these quality checks across various data storage systems ensures that these checks are executed quickly and efficiently.

**Deploying support center sentiment analysis**

Deploying the solution involves the following components:

1. NetApp DataOps Toolkit
2. NGC Configuration
3. NVIDIA RIVA Server
4. NVIDIA TAO Toolkit
5. Export TAO models to RIVA

To perform deployment, complete the following steps:

**NetApp DataOps Toolkit: Support center sentiment analysis**

To use the NetApp DataOps Toolkit, complete the following steps:

1. Pip install the toolkit.

   ```bash
   python3 -m pip install netapp-dataops-traditional
   ```

2. Configure the data management

   ```bash
   netapp_dataops_cli.py config
   ```
NGC configuration: Support center sentiment analysis

To set up NVIDIA NGC, complete the following steps:

1. Download the NGC.

```
```

2. Add your current directory to path.

```
echo "export PATH="\$PATH:$(pwd)"" >> ~/.bash_profile && source ~/.bash_profile
```

3. You must configure NGC CLI for your use so that you can run the commands. Enter the following command, including your API key when prompted.

```
ngc config set
```

For operating systems that are not Linux-based, visit here.

NVIDIA RIVA server: Support center sentiment analysis

To set up NVIDIA RIVA, complete the following steps:

1. Download the RIVA files from NGC.

```
ngc registry resource download-version nvidia/riva/riva_quickstart:1.4.0-beta
```

2. Initialize the RIVA setup (riva_init.sh).

3. Start the RIVA server (riva_start.sh).

4. Start the RIVA client (riva_start_client.sh).

5. Within the RIVA client, install the audio processing library (FFMPEG)

```
apt-get install ffmpeg
```

6. Start the Jupyter server.

7. Run the RIVA Inference Pipeline Notebook.
NVIDIA TAO Toolkit: Support center sentiment analysis

To set up NVIDIA TAO Toolkit, complete the following steps:

1. Prepare and activate a virtual environment for TAO Toolkit.
2. Install the required packages.
3. Manually pull the image used during training and fine-tuning.

```docker pull nvcr.io/nvidia/tao/tao-toolkit-pyt:v3.21.08-py3```

4. Start the Jupyter server.
5. Run the TAO Fine-Tuning Notebook.

Export TAO models to RIVA: Support center sentiment analysis

To use TAO Toolkit models in RIVA, complete the following steps:

1. Save models within the TAO Fine-Tuning Notebook.
2. Copy TAO trained models to the RIVA model directory.
3. Start the RIVA server (riva_start.sh).

Deployment roadblocks

Here are a few things to keep in mind as you develop your own solution:

- The NetApp DataOps Toolkit is installed first to ensure that the data storage system runs optimally.
- NVIDIA NGC must be installed before anything else because it authenticates the downloading of images and models.
- RIVA must be installed before the TAO Toolkit. The RIVA installation configures the docker daemon to pull images as needed.
- DGX and docker must have internet access to download the models.

Validation results

As mentioned in the previous section, errors are propagated throughout the pipeline whenever there are two or more machine learning models running in sequence. For this solution, the sentiment of the sentence is the most important factor in measuring the firm’s stock risk level. The speech-to-text model, although essential to the pipeline, serves as the preprocessing unit before the sentiments can be predicted. What really matters is the difference in sentiment between the ground truth sentences and the predicted sentences. This serves as a proxy for the word error rate (WER). The speech-to-text accuracy is important, but the WER is not directly used in the final pipeline metric.

```
PIPELINE_SENTIMENT_METRIC = MEAN(DIFF(GT_sentiment, ASR_sentiment))
```

These sentiment metrics can be calculated for the F1 Score, Recall, and Precision of each sentence. The
results can be then aggregated and displayed within a confusion matrix, along with the confidence intervals for each metric.

The benefit of using transfer learning is an increase in model performance for a fraction of data requirements, training time, and cost. The fine-tuned models should also be compared to their baseline versions to ensure the transfer learning enhances the performance instead of impairing it. In other words, the fine-tuned model should perform better on the support center data than the pretrained model.

**Pipeline assessment**

<table>
<thead>
<tr>
<th>Test case</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test number</td>
<td>Pipeline sentiment metric</td>
</tr>
<tr>
<td>Test prerequisites</td>
<td>Fine-tuned models for speech-to-text and sentiment analysis models</td>
</tr>
<tr>
<td>Expected outcome</td>
<td>The sentiment metric of the fine-tuned model performs better than the original pretrained model.</td>
</tr>
</tbody>
</table>

**Pipeline sentiment metric**

1. Calculate the sentiment metric for the baseline model.
2. Calculate the sentiment metric for the fine-tuned model.
3. Calculate the difference between those metrics.
4. Average the differences across all sentences.

**Videos and demos**

There are two notebooks that contain the sentiment analysis pipeline: “Support-Center-Model-Transfer-Learning-and-Fine-Tuning.ipynb” and “Support-Center-Sentiment-Analysis-Pipeline.ipynb”. Together, these notebooks demonstrate how to develop a pipeline to ingest support center data and extract sentiments from each sentence using state-of-the-art deep learning models fine-tuned on the user's data.

**Support Center - Sentiment Analysis Pipeline.ipynb**

This notebook contains the inference RIVA pipeline for ingesting audio, converting it to text, and extracting sentiments for use in an external dashboard. Dataset are automatically downloaded and processed if this has not already been done. The first section in the notebook is the Speech-to-Text which handles the conversion of audio files to text. This is followed by the Sentiment Analysis section which extracts sentiments for each text sentence and displays those results in a format similar to the proposed dashboard.

- This notebook must be run before the model training and fine-tuning because the MP3 dataset must be downloaded and converted into the correct format.
Call Center - Sentiment Analysis Pipeline

This notebook demonstrates how to build a pipeline for sentiment analysis of call center conversations. The goal of this pipeline is to develop sentiment analysis for use within an external dashboard.

This tutorial will guide you through the use of NVIDIA’s RIVA for automatic speech recognition and text classification. This tutorial uses NetApp cloud storage for data storage and a pre-trained RIVA model.

Channels

These are the channels on which RIVA is hosting models.

- **speech**: 51051
- **voice**: 61051

These channels must be aligned with `riva_speech_api_port` and `riva_vision_api_port` within `config.sh`.

In [4]: speech_channel = "localhost:51051"
    voice_channel = "localhost:61051"

Speech-To-Text

Automatic Speech Recognition (ASR) takes as input an audio stream or audio buffer and returns one or more text transcripts, along with additional optional metadata. ASR represents a full speech recognition pipeline that is GPU accelerated with optimized performance and accuracy. ASR supports synchronous and streaming recognition modes.

For more information on NVIDIA RIVA’s Automatic Speech Recognition, visit [here](#).

Constants

Use these constants to affect different aspects of this pipeline:

- **DATA_DIR**: base folder where data is stored
- **DATASET_NAME**: name of the call center dataset
- **COMPANY_DATE**: folder name identifying the particular call center conversation

Support Center - Model Training and Fine-Tuning.ipynb

The TAO Toolkit virtual environment must be set up before executing the notebook (see the TAO Toolkit section in the Commands Overview for installation instructions).

This notebook relies on the TAO Toolkit to fine-tune deep learning models on the customers data. As with the previous notebook, this one is separated into two sections for the Speech-to-Text and Sentiment Analysis components. Each section goes through data processing, model training and fine-tuning, evaluation of results, and model export. Finally, there is an end section for deploying both your fine-tuned models for use in RIVA.
Conclusion

As customer experience has become increasingly regarded as a key competitive battleground, an AI-augmented global support center becomes a critical component that companies in almost every industry cannot afford to neglect. The solution proposed in this technical report has been demonstrated to support the delivery of such exceptional customer experiences, and the challenge now is to ensure businesses are taking actions to modernize their AI infrastructure and workflows.

The best implementations of AI in customer service are not to replace human agents. Rather, AI can empower them to create exceptional customer experiences via real-time sentiment analysis, dispute escalation, and multimodal affective computing to detect verbal, non-verbal, and facial cues with which comprehensive AI models can make recommendations at scale and supplement what an individual human agent might be.
lacking. AI can also provide a better match between a particular customer with currently available agents. Using AI, businesses can extract valuable customer sentiment regarding their thoughts and impressions of the provider's products, services, and brand image.

The solution can also be used to construct time-series data for support agents to serve as an objective performance evaluation metric. Conventional customer satisfaction surveys often lack sufficient responses. By collecting long-term employee and customer sentiment, employers can make informed decisions regarding support agents' performance.

The combination of NetApp, SFL Scientific, opens-source orchestration frameworks, and NVIDIA brings the latest technologies together as managed services with great flexibility to accelerate technology adoption and improve the time to market for new AI/ML applications. These advanced services are delivered on-premises that can be easily ported for cloud-native environment as well as hybrid deployment architectures.

Where to find additional information

To learn more about the information that is described in this document, review the following documents and/or websites:

• 3D interactive demos
  www.netapp.com/ai

• Connect directly with a NetApp AI specialist
  https://www.netapp.com/artificial-intelligence/

• NVIDIA Base Command Platform with NetApp solution brief

• NetApp for AI 10 Good Reasons infographic

• AI in Healthcare: Deep learning to identify COVID-19 lesions in lung CT scans white paper

• AI in Healthcare: Monitoring face mask usage in healthcare settings white paper

• AI in Healthcare: Diagnostic Imaging Technical Report

• AI for Retail: NetApp Conversational AI using NVIDIA RIVA

• NetApp ONTAP AI solution brief
Distributed training in Azure - Click-Through Rate Prediction

TR-4904: Distributed training in Azure - Click-Through Rate Prediction

Rick Huang, Verron Martina, Muneer Ahmad, NetApp

The work of a data scientist should be focused on the training and tuning of machine learning (ML) and artificial intelligence (AI) models. However, according to research by Google, data scientists spend approximately 80% of their time figuring out how to make their models work with enterprise applications and run at scale.

To manage end-to-end AI/ML projects, a wider understanding of enterprise components is needed. Although DevOps have taken over the definition, integration, and deployment, these types of components, ML operations target a similar flow that includes AI/ML projects. To get an idea of what an end-to-end AI/ML pipeline touches in the enterprise, see the following list of required components:

- Storage
- Networking
- Databases
- File systems
- Containers
• Continuous integration and continuous deployment (CI/CD) pipeline
• Integrated development environment (IDE)
• Security
• Data access policies
• Hardware
• Cloud
• Virtualization
• Data science toolsets and libraries

Target audience

The world of data science touches multiple disciplines in IT and business:

• The data scientist needs the flexibility to use their tools and libraries of choice.
• The data engineer needs to know how the data flows and where it resides.
• A DevOps engineer needs the tools to integrate new AI/ML applications into their CI/CD pipelines.
• Cloud administrators and architects need to be able to set up and manage Azure resources.
• Business users want to have access to AI/ML applications.

In this technical report, we describe how Azure NetApp Files, RAPIDS AI, Dask, and Azure help each of these roles bring value to business.

Solution overview

This solution follows the lifecycle of an AI/ML application. We start with the work of data scientists to define the different steps needed to prepare data and train models. By leveraging RAPIDS on Dask, we perform distributed training across the Azure Kubernetes Service (AKS) cluster to drastically reduce the training time when compared to the conventional Python scikit-learn approach. To complete the full cycle, we integrate the pipeline with Azure NetApp Files.

Azure NetApp Files provides various performance tiers. Customers can start with a Standard tier and scale out and scale up to a high-performance tier nondisruptively without moving any data. This capability enables data scientists to train models at scale without any performance issues, avoiding any data silos across the cluster, as shown in figure below.
Technology overview

This page provides an overview of the technology used in this solution.

Microsoft and NetApp

Since May 2019, Microsoft has delivered an Azure native, first-party portal service for enterprise NFS and SMB file services based on NetApp ONTAP technology. This development is driven by a strategic partnership between Microsoft and NetApp and further extends the reach of world-class ONTAP data services to Azure.

Azure NetApp Files

The Azure NetApp Files service is an enterprise-class, high-performance, metered file storage service. Azure NetApp Files supports any workload type and is highly available by default. You can select service and performance levels and set up Snapshot copies through the service. Azure NetApp Files is an Azure first-party service for migrating and running the most demanding enterprise-file workloads in the cloud, including databases, SAP, and high-performance computing applications with no code changes.

This reference architecture gives IT organizations the following advantages:

- Eliminates design complexities
- Enables independent scaling of compute and storage
- Enables customers to start small and scale seamlessly
- Offers a range of storage tiers for various performance and cost points

Dask and NVIDIA RAPIDS overview

Dask is an open-source, parallel computing tool that scales Python libraries on multiple machines and provides faster processing of large amounts of data. It provides an API similar to single-threaded conventional Python libraries, such as Pandas, Numpy, and scikit-learn. As a result, native Python users are not forced to change much in their existing code to use resources across the cluster.

NVIDIA RAPIDS is a suite of open-source libraries that makes it possible to run end-to-end ML and data analytics workflows entirely on GPUs. Together with Dask, it enables you to easily scale from GPU workstation (scale up) to multinode, multi-GPU clusters (scale out).
For deploying Dask on a cluster, you could use Kubernetes for resource orchestration. You could also scale up or scale down the worker nodes as per the process requirement, which in-turn can help to optimize the cluster resource consumption, as shown in the following figure.

**Software requirements**

The following table lists the software requirements needed for this solution.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Kubernetes Service</td>
<td>1.18.14</td>
</tr>
<tr>
<td>RAPIDS and Dask container image</td>
<td>Repository: &quot;rapidsai/rapidsai&quot;</td>
</tr>
<tr>
<td></td>
<td>Tag: 0.17-cuda11.0-runtime-ubuntu18.04</td>
</tr>
<tr>
<td>NetApp Trident</td>
<td>20.01.1</td>
</tr>
<tr>
<td>Helm</td>
<td>3.0.0</td>
</tr>
</tbody>
</table>

**Cloud resource requirements**

This page describes the configuration of cloud resources for Azure NetApp Files.

**Configure Azure NetApp Files**

Configure Azure NetApp Files as described in [QuickStart: Set up Azure NetApp Files and create an NFS volume](#).

You can proceed past the section “Create NFS volume for Azure NetApp Files” because you are going to create volumes through Trident. Before continuing, complete the following steps:

1. Register for Azure NetApp Files and NetApp Resource Provider (through the Azure Shell) ([link](#)).
2. Create an account in Azure NetApp Files (link).

3. Set up a capacity pool (a minimum 4TB Standard or Premium, depending on your need) (link). The following table lists the network configuration requirements for setting up in the cloud. The Dask cluster and Azure NetApp Files must be in the same Azure Virtual Network (VNet) or a peered VNet.

<table>
<thead>
<tr>
<th>Resources</th>
<th>Type/version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Kubernetes Service</td>
<td>1.18.14</td>
</tr>
<tr>
<td>Agent node</td>
<td>3x Standard_DS2_v2</td>
</tr>
<tr>
<td>GPU node</td>
<td>3x Standard_NC6s_v3</td>
</tr>
<tr>
<td>Azure NetApp Files</td>
<td>Standard capacity pool</td>
</tr>
<tr>
<td>Capacity in TB</td>
<td>4</td>
</tr>
</tbody>
</table>

Click-through rate prediction use case summary

This use case is based on the publicly available Terabyte Click Logs dataset from Criteo AI Lab. With the recent advances in ML platforms and applications, a lot of attention is now on learning at scale. The click-through rate (CTR) is defined as the average number of click-throughs per hundred online ad impressions (expressed as a percentage). It is widely adopted as a key metric in various industry verticals and use cases, including digital marketing, retail, e-commerce, and service providers. Examples of using CTR as an important metric for potential customer traffic include the following:

- **Digital marketing**: In Google Analytics, CTR can be used to gauge how well an advertiser or merchant's keywords, ads, and free listings are performing. A high CTR is a good indication that users find your ads and listings helpful and relevant. CTR also contributes to your keyword's expected CTR, which is a component of Ad Rank.

- **E-commerce**: In addition to leveraging Google Analytics, there are at least some visitor statistics in an e-commerce backend. Although these statistics might not seem useful at first glance, they are typically easy to read and might be more accurate than other information. First-party datasets composed of such statistics are proprietary and are therefore the most relevant to e-commerce sellers, buyers, and platforms. These datasets can be used for setting benchmarks, comparing results to last year and yesterday by constructing a time-series for further analysis.

- **Retail**: Brick-and-mortar retailers can correlate the number of visitors and the number of customers to the CTR. The number of customers can be seen from their point-of-sale history. The CTR from retailers' websites or ad traffic might result in the aforementioned sales. Loyalty programs are another use case, because customers redirected from online ads or other websites might join to earn rewards. Retailers can acquire customers via loyalty programs and record behaviors from sales histories to build a recommendation system that not only predicts consumer buying behaviors in different categories but also personalizes coupons and decreases churn.

- **Service providers**: Telecommunication companies and internet service providers have an abundance of first-party user telemetry data for insightful AI, ML, and analytics use cases. For example, a telecom can leverage its mobile subscribers' web browsing top level domain history logs daily to fine-tune existing models to produce up-to-date audience segmentation, predict customer behavior, and collaborate with advertisers to place real-time ads for better online experience. In such data-driven marketing workflow, CTR is an important metric to reflect conversions.

In the context of digital marketing, Criteo Terabyte Click Logs are now the dataset of reference in assessing the scalability of ML platforms and algorithms. By predicting the click-through rate, an advertiser can select the
visitors who are most likely to respond to the ads, analyze their browsing history, and show the most relevant ads based on the interests of the user.

The solution provided in this technical report highlights the following benefits:

- Azure NetApp Files advantages in distributed or large-scale training
- RAPIDS CUDA-enabled data processing (cuDF, cuPy, and so on) and ML algorithms (cuML)
- The Dask parallel computing framework for distributed training

An end-to-end workflow built on RAPIDS AI and Azure NetApp Files demonstrates the drastic improvement in random forest model training time by two orders of magnitude. This improvement is significant comparing to the conventional Pandas approach when dealing with real-world click logs with 45GB of structured tabular data (on average) each day. This is equivalent to a DataFrame containing roughly twenty billion rows. We will demonstrate cluster environment setup, framework and library installation, data loading and processing, conventional versus distributed training, visualization and monitoring, and compare critical end-to-end runtime results in this technical report.

**Setup**

**Install and set up the AKS cluster**

To install and set up the AKS cluster, see the webpage Create an AKS Cluster and then complete the following steps:

1. When selecting the type of node (system [CPU] or worker [GPU] nodes), select the following:
   a. Primary system nodes should be Standard DS2v2 (agentpool default three nodes).
   b. Then add the worker node Standard_NC6s_v3 pool (three nodes minimum) for the user group (for GPU nodes) named gpupool.

2. Deployment takes 5 to 10 minutes. After it is complete, click Connect to Cluster.

3. To connect to the newly created AKS cluster, install the following from your local environment (laptop/pc):
   a. The Kubernetes command-line tool using the instructions provided for your specific OS
   b. The Azure CLI as described in the document, Install the Azure CLI

4. To access the AKS cluster from the terminal, enter az login and enter the credentials.

5. Run the following two commands:

```
az account set --subscription xxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxx
aks get-credentials --resource-group resourcegroup --name aksclustername
```
6. Enter Azure CLI: `kubectl get nodes`.

7. If all six nodes are up and running, as shown in the following example, your AKS cluster is ready and connected to your local environment.

```
vernonmartina@vernon-mac-0 ~ % kubectl get nodes
NAME                        STATUS    ROLES     AGE       VERSION
aks-agentpool-34613062-vmss00000 Ready agent 22m v1.18.14
aks-agentpool-34613062-vmss00001 Ready agent 22m v1.18.14
aks-agentpool-34613062-vmss00002 Ready agent 22m v1.18.14
aks-gpupool-34613062-vmss00000 Ready agent 20m v1.18.14
aks-gpupool-34613062-vmss00001 Ready agent 20m v1.18.14
aks-gpupool-34613062-vmss00002 Ready agent 20m v1.18.14
vernonmartina@vernon-mac-0 ~ %
```

Create a delegated subnet for Azure NetApp Files

To create a delegated subnet for Azure NetApp Files, complete the following steps:

1. Navigate to Virtual Networks within the Azure portal. Find your newly created virtual network. It should have a prefix such as `aks-vnet`.

2. Click the name of the VNet.

```
Dashboard
Virtual networks

Add Manage view Refresh Export to CSV Open query Assign tags Feedback

Filter by name... Subscription == AzureSub01 Resource group == all Location == all Add filter

Showing 1 to 5 of 5 records.

<table>
<thead>
<tr>
<th>Name</th>
<th>Resource group</th>
<th>Location</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>aks-vnet-22885919</td>
<td>MC_sluce_rg_TridentDemo_eastus2</td>
<td>East US 2</td>
<td>AzureSub01</td>
</tr>
</tbody>
</table>
```

3. Click Subnets and click +Subnet from the top toolbar.
4. Provide the subnet with a name such as `ANF.sn` and, under the Subnet Delegation heading, select `Microsoft.Netapp/volumes`. Do not change anything else. Click OK.
Azure NetApp Files volumes are allocated to the application cluster and are consumed as persistent volume claims (PVCs) in Kubernetes. In turn, this process provides you the flexibility to map them to different services, such as Jupyter notebooks, serverless functions, and so on.

Users of services can consume storage from the platform in many ways. As this technical report discusses NFSs, the main benefits of Azure NetApp Files are:

- Providing users with the ability to use Snapshot copies.
- Enabling users to store large quantities of data on Azure NetApp Files volumes.
- Using the performance benefits of Azure NetApp Files volumes when running their models on large sets of files.
Peer AKS VNet and Azure NetApp Files VNet

To peer the AKS VNet to the Azure NetApp Files VNet, complete the following steps:

1. Enter Virtual Networks in the search field.
2. Select vnet aks-vnet-name. Click it and enter Peerings in the search field.
3. Click +Add.
4. Enter the following descriptors:
   a. The peering link name is aks-vnet-name_to_anf.
   b. subscriptionID and Azure NetApp Files VNet as the VNet peering partner.
   c. Leave all the nonasterisk sections with the default values.
5. Click Add.

For more information, see Create, change, or delete a virtual network peering.

Install Trident

To install Trident using Helm, complete the following steps:

1. Install Helm (for installation instructions, visit the source).
2. Download and extract the Trident 20.01.1 installer.

   $wget
   $tar -xf trident-installer-21.01.1.tar.gz

3. Change the directory to trident-installer.

   $cd trident-installer

4. Copy tridentctl to a directory in your system $PATH.

   $sudo cp ./tridentctl /usr/local/bin

5. Install Trident on the Kubernetes (K8s) cluster with Helm (source):
   a. Change the directory to the helm directory.

      $cd helm

   b. Install Trident.
c. Check the status of Trident pods.

```bash
kubectl -n trident get pods
```

If all the pods are up and running, then Trident is installed and you can move forward.

6. Set up the Azure NetApp Files backend and storage class for AKS.


      The service principal is how Trident communicates with Azure to manipulate your Azure NetApp Files resources.

      ```bash
      az ad sp create-for-rbac --name ""
      ```

      The output should look like the following example:

      ```json
      {
      "appId": "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx",
      "displayName": "netapptrident",
      "name": "",
      "password": "xxxxxxxxxxxxxxxx.xxxxxxxxxxxxxxx",
      "tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
      }
      ```

7. Create a Trident backend json file, example name `anf-backend.json`.

8. Using your preferred text editor, complete the following fields inside the `anf-backend.json` file:
9. Substitute the following fields:
   - subscriptionID. Your Azure subscription ID.
   - tenantID. Your Azure Tenant ID from the output of `az ad sp` in the previous step.
   - clientID. Your appID from the output of `az ad sp` in the previous step.
   - clientSecret. Your password from the output of `az ad sp` in the previous step.

10. Instruct Trident to create the Azure NetApp Files backend in the `trident` namespace using `anf-backend.json` as the configuration file:

    ```bash
    $tridentctl create backend -f anf-backend.json -n trident
    ```

11. Create a storage class. Kubernetes users provision volumes by using PVCs that specify a storage class by name. Instruct K8s to create a storage class `azurenetappfiles` that references the Trident backend created in the previous step.

12. Create a YAML (`anf-storage-class.yaml`) file for storage class and copy.
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: azurenetappfiles
  provisioner: netapp.io/trident
parameters:
  backendType: "azure-netapp-files"
kubectl create -f anf-storage-class.yaml

13. Verify that the storage class was created.
kubectl get sc azurenetappfiles

Set up Dask with RAPIDS deployment on AKS using Helm

To set up Dask with RAPIDS deployment on AKS using Helm, complete the following steps:

1. Create a namespace for installing Dask with RAPIDS.
kubectl create namespace rapids-dask

2. Create a PVC to store the click-through rate dataset:
   a. Save the following YAML content to a file to create a PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-criteo-data
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1000Gi
  storageClassName: azurenetappfiles
```

   b. Apply the YAML file to your Kubernetes cluster.
kubectl -n rapids-dask apply -f <your yaml file>


```bash
git clone https://github.com/rapidsai/helm-chart helm-chart
```

4. Modify values.yaml and include the PVC created earlier for workers and Jupyter workspace.
   a. Go to the rapidsai directory of the repository.

```bash
cd helm-chart/rapidsai
```
   b. Update the values.yaml file and mount the volume using PVC.

```
dask:
  ...
  worker:
    name: worker
  ...
  mounts:
    volumes:
      - name: data
        persistentVolumeClaim:
          claimName: pvc-criteo-data
        volumeMounts:
          - name: data
            mountPath: /data
  ...

jupyter:
  name: jupyter
  ...
  mounts:
    volumes:
      - name: data
        persistentVolumeClaim:
          claimName: pvc-criteo-data
        volumeMounts:
          - name: data
            mountPath: /data
  ...
```

5. Go to the repository’s home directory and deploy Dask with three worker nodes on AKS using Helm.
cd ..
helm dep update rapidsai
helm install rapids-dask --namespace rapids-dask rapidsai

Azure NetApp Files performance tiers

You can change the service level of an existing volume by moving the volume to another capacity pool that uses the service level you want for the volume. This solution enables customers to start with a small dataset and small number of GPUs in Standard Tier and scale out or scale up to Premium Tier as the amount of data and GPUs increase. The Premium Tier offers four times the throughput per terabyte as the Standard Tier, and scale up is performed without having to move any data to change the service level of a volume.

Dynamically change the service level of a volume

To dynamically change the service level of a volume, complete the following steps:

1. On the Volumes page, right-click the volume whose service level you want to change. Select Change Pool.

2. In the Change Pool window, select the capacity pool to which you want to move the volume.
3. Click OK.

**Automate performance tier change**

The following options are available to automate performance tier changes:

- Dynamic Service Level change is still in Public Preview at this time and not enabled by default. To enable this feature on the Azure Subscription, see this documentation about how to [Dynamically change the service level of a volume](#).

- Azure CLI volume pool change commands are provided in [volume pool change documentation](#) and in the following example:

  ```
  az netappfiles volume pool-change -g mygroup --account-name myaccname --pool-name mypoolname --name myvolname --new-pool-resource-id mynewresourceid
  ```

- PowerShell: The [Set-AzNetAppFilesVolumePool cmdlet](#) changes the pool of an Azure NetApp Files volume and is shown in the following example:
Click through rate prediction data processing and model training

Libraries for data processing and model training

The following table lists the libraries and frameworks that were used to build this task. All these components have been fully integrated with Azure’s role-based access and security controls.

<table>
<thead>
<tr>
<th>Libraries/framework</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dask cuML</td>
<td>For ML to work on GPU, the cuML library provides access to the RAPIDS cuML package with Dask. RAPIDS cuML implements popular ML algorithms, including clustering, dimensionality reduction, and regression approaches, with high-performance GPU-based implementations, offering speed-ups of up to 100x over CPU-based approaches.</td>
</tr>
<tr>
<td>Dask cuDF</td>
<td>cuDF includes various other functions supporting GPU-accelerated extract, transform, load (ETL), such as data subsetting, transformations, one-hot encoding, and more. The RAPIDS team maintains a dask-cudf library that includes helper methods to use Dask and cuDF.</td>
</tr>
<tr>
<td>Scikit Learn</td>
<td>Scikit-learn provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method.</td>
</tr>
</tbody>
</table>

We used two notebooks to construct the ML pipelines for comparison; one is the conventional Pandas scikit-learn approach, and the other is distributed training with RAPIDS and Dask. Each notebook can be tested individually to see the performance in terms of time and scale. We cover each notebook individually to demonstrate the benefits of distributed training using RAPIDS and Dask.

Load Criteo Click Logs day 15 in Pandas and train a scikit-learn random forest model

This section describes how we used Pandas and Dask DataFrames to load Click Logs data from the Criteo Terabyte dataset. The use case is relevant in digital advertising for ad exchanges to build users’ profiles by predicting whether ads will be clicked or if the exchange isn’t using an accurate model in an automated pipeline.

We loaded day 15 data from the Click Logs dataset, totaling 45GB. Running the following cell in Jupyter

```bash
Set-AzNetAppFilesVolumePool -ResourceGroupName "MyRG" -AccountName "MyAnfAccount" -PoolName "MyAnfPool" -Name "MyAnfVolume" -NewPoolResourceId 7d6e4069-6c78-6c61-7bf6-c60968e45fbf
```
notebook CTR-PandasRF-collated.ipynb creates a Pandas DataFrame that contains the first 50 million rows and generates a scikit-learn random forest model.

```python
import pandas as pd
import numpy as np
header = ['col'+str(i) for i in range(1,41)] #note that according to criteo, the first column in the dataset is Click Through (CT). Consist of 40 columns
first_row_taken = 50_000_000 # use this in pd.read_csv() if your compute resource is limited.
# total number of rows in day15 is 20B
# take 50M rows

""" Read data & display the following metrics:
1. Total number of rows per day
2. df loading time in the cluster
3. Train a random forest model
"""

df = pd.read_csv(file, nrows=first_row_taken, delimiter='\t', names=header)
# take numerical columns
df_sliced = df.iloc[:, 0:14]
# split data into training and Y
Y = df_sliced.pop('col1') # first column is binary (click or not)
# change df_sliced data types & fillna
df_sliced = df_sliced.astype(np.float32).fillna(0)
from sklearn.ensemble import RandomForestClassifier
# Random Forest building parameters
# n_streams = 8 # optimization
max_depth = 10
n_bins = 16
n_trees = 10
rf_model = RandomForestClassifier(max_depth=max_depth,
n_estimators=n_trees)
rf_model.fit(df_sliced, Y)
```

To perform prediction by using a trained random forest model, run the following paragraph in this notebook. We took the last one million rows from day 15 as the test set to avoid any duplication. The cell also calculates accuracy of prediction, defined as the percentage of occurrences the model accurately predicts whether a user clicks an ad or not. To review any unfamiliar components in this notebook, see the official scikit-learn documentation.
Load Day 15 in Dask and train a Dask cuML random forest model

In a manner similar to the previous section, load Criteo Click Logs day 15 in Pandas and train a scikit-learn random forest model. In this example, we performed DataFrame loading with Dask cuDF and trained a random forest model in Dask cuML. We compared the differences in training time and scale in the section “Training time comparison.”

criteo_dask_RF.ipynb

This notebook imports numpy, cuml, and the necessary dask libraries, as shown in the following example:

```python
import cuml
from dask.distributed import Client, progress, wait
import dask_cudf
import numpy as np
import cudf
from cuml.dask.ensemble import RandomForestClassifier as cumlDaskRF
from cuml.dask.common import utils as dask_utils
```

Initiate Dask Client().

```python
client = Client()
```

If your cluster is configured correctly, you can see the status of worker nodes.
client
workers = client.has_what().keys()
n_workers = len(workers)
n_streams = 8  # Performance optimization

In our AKS cluster, the following status is displayed:

![Cluster Status](image)

Note that Dask employs the lazy execution paradigm: rather than executing the processing code instantly, Dask builds a Directed Acyclic Graph (DAG) of execution instead. DAG contains a set of tasks and their interactions that each worker needs to run. This layout means the tasks do not run until the user tells Dask to execute them in one way or another. With Dask you have three main options:

- **Call compute() on a DataFrame.** This call processes all the partitions and then returns results to the scheduler for final aggregation and conversion to cuDF DataFrame. This option should be used sparingly and only on heavily reduced results unless your scheduler node runs out of memory.

- **Call persist() on a DataFrame.** This call executes the graph, but, instead of returning the results to the scheduler node, it maintains them across the cluster in memory so the user can reuse these intermediate results down the pipeline without the need for rerunning the same processing.

- **Call head() on a DataFrame.** Just like with cuDF, this call returns 10 records back to the scheduler node. This option can be used to quickly check if your DataFrame contains the desired output format, or if the records themselves make sense, depending on your processing and calculation.

Therefore, unless the user calls either of these actions, the workers sit idle waiting for the scheduler to initiate the processing. This lazy execution paradigm is common in modern parallel and distributed computing frameworks such as Apache Spark.

The following paragraph trains a random forest model by using Dask cuML for distributed GPU-accelerated computing and calculates model prediction accuracy.
Monitor Dask using native Task Streams dashboard

The Dask distributed scheduler provides live feedback in two forms:

- An interactive dashboard containing many plots and tables with live information
- A progress bar suitable for interactive use in consoles or notebooks

In our case, the following figure shows how you can monitor the task progress, including Bytes Stored, the Task Stream with a detailed breakdown of the number of streams, and Progress by task names with associated functions executed. In our case, because we have three worker nodes, there are three main chunks of stream and the color codes denote different tasks within each stream.

You have the option to analyze individual tasks and examine the execution time in milliseconds or identify any obstacles or hindrances. For example, the following figure shows the Task Streams for the random forest model fitting stage. There are considerably more functions being executed, including unique chunk for DataFrame processing, _construct_rf for fitting the random forest, and so on. Most of the time was spent on DataFrame operations due to the large size (45GB) of one day of data from the Criteo Click Logs.
Training time comparison

This section compares the model training time using conventional Pandas compared to Dask. For Pandas, we loaded a smaller amount of data due to the nature of slower processing time to avoid memory overflow. Therefore, we interpolated the results to offer a fair comparison.

The following table shows the raw training time comparison when there is significantly less data used for the Pandas random forest model (50 million rows out of 20 billion per day of the dataset). This sample is only using less than 0.25% of all available data. Whereas for Dask-cuML we trained the random forest model on all 20 billion available rows. The two approaches yielded comparable training time.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scikit-learn: Using only 50M rows in day15 as the training data</td>
<td>47 minutes and 21 seconds</td>
</tr>
<tr>
<td>RAPIDS-Dask: Using all 20B rows in day15 as the training data</td>
<td>1 hour, 12 minutes, and 11 seconds</td>
</tr>
</tbody>
</table>

If we interpolate the training time results linearly, as shown in the following table, there is a significant advantage to using distributed training with Dask. It would take the conventional Pandas scikit-learn approach 13 days to process and train 45GB of data for a single day of click logs, whereas the RAPIDS-Dask approach processes the same amount of data 262.39 times faster.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scikit-learn: Using all 20B rows in day15 as the training data</td>
<td>13 days, 3 hours, 40 minutes, and 11 seconds</td>
</tr>
<tr>
<td>RAPIDS-Dask: Using all 20B rows in day15 as the training data</td>
<td>1 hour, 12 minutes, and 11 seconds</td>
</tr>
</tbody>
</table>

In the previous table, you can see that by using RAPIDS with Dask to distribute the data processing and model training across multiple GPU instances, the run time is significantly shorter compared to conventional Pandas DataFrame processing with scikit-learn model training. This framework enables scaling up and out in the cloud.
as well as on-premises in a multinode, multi-GPU cluster.

**Monitor Dask and RAPIDS with Prometheus and Grafana**

After everything is deployed, run inferences on new data. The models predict whether a user clicks an ad based on browsing activities. The results of the prediction are stored in a Dask cuDF. You can monitor the results with Prometheus and visualize in Grafana dashboards.

For more information, see this [RAPIDS AI Medium post](#).

**Dataset and model versioning using NetApp DataOps Toolkit**

The NetApp DataOps Toolkit for Kubernetes abstracts storage resources and Kubernetes workloads up to the data-science workspace level. These capabilities are packaged in a simple, easy-to-use interface that is designed for data scientists and data engineers. Using the familiar form of a Python program, the Toolkit enables data scientists and engineers to provision and destroy JupyterLab workspaces in just seconds. These workspaces can contain terabytes, or even petabytes, of storage capacity, enabling data scientists to store all their training datasets directly in their project workspaces. Gone are the days of separately managing workspaces and data volumes.

For more information, visit the Toolkit’s [GitHub repository](#).

**Jupyter notebooks for reference**

There are two Jupyter notebooks associated with this technical report:

- **CTR-PandasRF-collated.ipynb.** This notebook loads Day 15 from the Criteo Terabyte Click Logs dataset, processes and formats data into a Pandas DataFrame, trains a Scikit-learn random forest model, performs prediction, and calculates accuracy.

- **criteo_dask_RF.ipynb.** This notebook loads Day 15 from the Criteo Terabyte Click Logs dataset, processes and formats data into a Dask cuDF, trains a Dask cuML random forest model, performs prediction, and calculates accuracy. By leveraging multiple worker nodes with GPUs, this distributed data and model processing and training approach is highly efficient. The more data you process, the greater the time savings versus a conventional ML approach. You can deploy this notebook in the cloud, on-premises, or in a hybrid environment where your Kubernetes cluster contains compute and storage in different locations, as long as your networking setup enables the free movement of data and model distribution.

**Conclusion**

Azure NetApp Files, RAPIDS, and Dask speed up and simplify the deployment of large-scale ML processing and training by integrating with orchestration tools such as Docker and Kubernetes. By unifying the end-to-end data pipeline, this solution reduces the latency and complexity inherent in many advanced computing workloads, effectively bridging the gap between development and operations. Data scientists can run queries on large datasets and securely share data and algorithmic models with other users during the training phase.

When building your own AI/ML pipelines, configuring the integration, management, security, and accessibility of
the components in an architecture is a challenging task. Giving developers access and control of their environment presents another set of challenges.

By building an end-to-end distributed training model and data pipeline in the cloud, we demonstrated two orders of magnitude improvement in total workflow completion time versus a conventional, open-source approach that did not leverage GPU-accelerated data processing and compute frameworks.

The combination of NetApp, Microsoft, opens-source orchestration frameworks, and NVIDIA brings the latest technologies together as managed services with great flexibility to accelerate technology adoption and improve the time to market for new AI/ML applications. These advanced services are delivered in a cloud-native environment that can be easily ported for on-premises as well as hybrid deployment architectures.

Where to find additional information

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

• Azure NetApp Files:
  ◦ Solutions architecture page for Azure NetApp Files
    https://docs.microsoft.com/azure/azure-netapp-files/azure-netapp-files-solution-architectures

• Trident persistent storage for containers:
  ◦ Azure NetApp Files and Trident

• Dask and RAPIDS:
  ◦ Dask
    https://docs.dask.org/en/latest/
  ◦ Install Dask
    https://docs.dask.org/en/latest/install.html
  ◦ Dask API
    https://docs.dask.org/en/latest/api.html
  ◦ Dask Machine Learning
    https://examples.dask.org/machine-learning.html
  ◦ Dask Distributed Diagnostics

• ML framework and tools:
    https://www.tensorflow.org/
TR-4896: Distributed training in Azure: Lane detection - Solution design

Muneer Ahmad and Verron Martina, NetApp
Ronen Dar, RUN:AI

Since May 2019, Microsoft delivers an Azure native, first-party portal service for enterprise NFS and SMB file services based on NetApp ONTAP technology. This development is driven by a strategic partnership between Microsoft and NetApp and further extends the reach of world-class ONTAP data services to Azure.

NetApp, a leading cloud data services provider, has teamed up with RUN: AI, a company virtualizing AI infrastructure, to allow faster AI experimentation with full GPU utilization. The partnership enables teams to speed up AI by running many experiments in parallel, with fast access to data, and leveraging limitless compute resources. RUN: AI enables full GPU utilization by automating resource allocation, and the proven architecture of Azure NetApp Files enables every experiment to run at maximum speed by eliminating data pipeline obstructions.

NetApp and RUN: AI have joined forces to offer customers a future-proof platform for their AI journey in Azure. From analytics and high-performance computing (HPC) to autonomous decisions (where customers can optimize their IT investments by only paying for what they need, when they need it), the alliance between NetApp and RUN: AI offers a single unified experience in the Azure Cloud.

Solution overview

In this architecture, the focus is on the most computationally intensive part of the AI or machine learning (ML) distributed training process of lane detection. Lane detection is one of the most important tasks in autonomous driving, which helps to guide vehicles by localization of the lane markings. Static components like lane markings guide the vehicle to drive on the highway interactively and safely.

Convolutional Neural Network (CNN)-based approaches have pushed scene understanding and segmentation to a new level. Although it doesn’t perform well for objects with long structures and regions that could be occluded (for example, poles, shade on the lane, and so on). Spatial Convolutional Neural Network (SCNN) generalizes the CNN to a rich spatial level. It allows information propagation between neurons in the same layer, which makes it best suited for structured objects such as lanes, poles, or truck with occlusions. This compatibility is because the spatial information can be reinforced, and it preserves smoothness and continuity.
Thousands of scene images need to be injected in the system to allow the model learn and distinguish the various components in the dataset. These images include weather, daytime or nighttime, multilane highway roads, and other traffic conditions.

For training, there is a need for good quality and quantity of data. Single GPU or multiple GPUs can take days to weeks to complete the training. Data-distributed training can speed up the process by using multiple and multinode GPUs. Horovod is one such framework that grants distributed training but reading data across clusters of GPUs could act as a hindrance. Azure NetApp Files provides ultrafast, high throughput and sustained low latency to provide scale-out/scale-up capabilities so that GPUs are leveraged to the best of their computational capacity. Our experiments verified that all the GPUs across the cluster are used more than 96% on average for training the lane detection using SCNN.

Target audience

Data science incorporates multiple disciplines in IT and business, therefore multiple personas are part of our targeted audience:

- Data scientists need the flexibility to use the tools and libraries of their choice.
- Data engineers need to know how the data flows and where it resides.
- Autonomous driving use-case experts.
- Cloud administrators and architects to set up and manage cloud (Azure) resources.
- A DevOps engineer needs the tools to integrate new AI/ML applications into their continuous integration and continuous deployment (CI/CD) pipelines.
- Business users want to have access to AI/ML applications.

In this document, we describe how Azure NetApp Files, RUN: AI, and Microsoft Azure help each of these roles bring value to business.

Solution technology

This section covers the technology requirements for the lane detection use case by implementing a distributed training solution at scale that fully runs in the Azure cloud. The figure below provides an overview of the solution architecture.

The elements used in this solution are:

- Azure Kubernetes Service (AKS)
- Azure Compute SKUs with NVIDIA GPUs
- Azure NetApp Files
- RUN: AI
- NetApp Trident

Links to all the elements mentioned here are listed in the Additional information section.
Cloud resources and services requirements

The following table lists the hardware components that are required to implement the solution. The cloud components that are used in any implementation of the solution might vary based on customer requirements.

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKS</td>
<td>Minimum of three system nodes and three GPU worker nodes</td>
</tr>
<tr>
<td>Virtual machine (VM) SKU system nodes</td>
<td>Three Standard_DS2_v2</td>
</tr>
<tr>
<td>VM SKU GPU worker nodes</td>
<td>Three Standard_NC6s_v3</td>
</tr>
<tr>
<td>Azure NetApp Files</td>
<td>4TB standard tier</td>
</tr>
</tbody>
</table>

Software requirements

The following table lists the software components that are required to implement the solution. The software components that are used in any implementation of the solution might vary based on customer requirements.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version or other information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKS - Kubernetes version</td>
<td>1.18.14</td>
</tr>
<tr>
<td>RUN:AI CLI</td>
<td>v2.2.25</td>
</tr>
<tr>
<td>RUN:AI Orchestration Kubernetes Operator version</td>
<td>1.0.109</td>
</tr>
<tr>
<td>Horovod</td>
<td>0.21.2</td>
</tr>
<tr>
<td>NetApp Trident</td>
<td>20.01.1</td>
</tr>
<tr>
<td>Helm</td>
<td>3.0.0</td>
</tr>
</tbody>
</table>

Lane detection – Distributed training with RUN:AI

This section provides details on setting up the platform for performing lane detection distributed training at scale using the RUN: AI orchestrator. We discuss installation of all the solution elements and running the distributed training job on the said platform. ML
versioning is completed by using NetApp SnapshotTM linked with RUN: AI experiments for achieving data and model reproducibility. ML versioning plays a crucial role in tracking models, sharing work between team members, reproducibility of results, rolling new model versions to production, and data provenance. NetApp ML version control (Snapshot) can capture point-in-time versions of the data, trained models, and logs associated with each experiment. It has rich API support making it easy to integrate with the RUN: AI platform; you just have to trigger an event based on the training state. You also have to capture the state of the whole experiment without changing anything in the code or the containers running on top of Kubernetes (K8s).

Finally, this technical report wraps up with performance evaluation on multiple GPU-enabled nodes across AKS.

**Distributed training for lane detection use case using the TuSimple dataset**

In this technical report, distributed training is performed on the TuSimple dataset for lane detection. Horovod is used in the training code for conducting data distributed training on multiple GPU nodes simultaneously in the Kubernetes cluster through AKS. Code is packaged as container images for TuSimple data download and processing. Processed data is stored on persistent volumes allocated by NetApp Trident plug-in. For the training, one more container image is created, and it uses the data stored on persistent volumes created during downloading the data.

To submit the data and training job, use RUN: AI for orchestrating the resource allocation and management. RUN: AI allows you to perform Message Passing Interface (MPI) operations which are needed for Horovod. This layout allows multiple GPU nodes to communicate with each other for updating the training weights after every training mini batch. It also enables monitoring of training through the UI and CLI, making it easy to monitor the progress of experiments.

NetApp Snapshot is integrated within the training code and captures the state of data and the trained model for every experiment. This capability enables you to track the version of data and code used, and the associated trained model generated.

**AKS setup and installation**

For setup and installation of the AKS cluster go to [Create an AKS Cluster](#). Then, follow these series of steps:

1. When selecting the type of nodes (whether it be system (CPU) or worker (GPU) nodes), select the following:
   a. Add primary system node named `agentpool` at the `Standard_DS2_v2` size. Use the default three nodes.
   b. Add worker node `gpupool` with the `Standard_NC6s_v3` pool size. Use three nodes minimum for GPU nodes.
Deployment takes 5–10 minutes.

2. After deployment is complete, click Connect to Cluster. To connect to the newly created AKS cluster, install the Kubernetes command-line tool from your local environment (laptop/PC). Visit Install Tools to install it as per your OS.

3. Install Azure CLI on your local environment.

4. To access the AKS cluster from the terminal, first enter `az login` and put in the credentials.

5. Run the following two commands:
   
   ```bash
   az account set --subscription xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
   aks get-credentials --resource-group resourcegroup --name aksclustername
   ```

6. Enter this command in the Azure CLI:
   
   ```bash
   kubectl get nodes
   ```

   If all six nodes are up and running as seen here, your AKS cluster is ready and connected to your local environment.

Create a delegated subnet for Azure NetApp Files

To create a delegated subnet for Azure NetApp Files, follow this series of steps:

1. Navigate to Virtual networks within the Azure portal. Find your newly created virtual network. It should have a prefix such as aks-vnet, as seen here. Click the name of the virtual network.
2. Click Subnets and select +Subnet from the top toolbar.

3. Provide the subnet with a name such as ANF_sn and under the Subnet Delegation heading, select Microsoft.NetApp/volumes. Do not change anything else. Click OK.
Azure NetApp Files volumes are allocated to the application cluster and are consumed as persistent volume claims (PVCs) in Kubernetes. In turn, this allocation provides us the flexibility to map volumes to different services, be it Jupyter notebooks, serverless functions, and so on.

Users of services can consume storage from the platform in many ways. The main benefits of Azure NetApp Files are:

- Provides users with the ability to use snapshots.
- Enables users to store large quantities of data on Azure NetApp Files volumes.
- Procure the performance benefits of Azure NetApp Files volumes when running their models on large sets of files.
Azure NetApp Files setup

To complete the setup of Azure NetApp Files, you must first configure it as described in Quickstart: Set up Azure NetApp Files and create an NFS volume.

However, you may omit the steps to create an NFS volume for Azure NetApp Files as you will create volumes through Trident. Before continuing, be sure that you have:

2. Created an account in Azure NetApp Files.
3. Set up a capacity pool (minimum 4TiB Standard or Premium depending on your needs).

Peering of AKS virtual network and Azure NetApp Files virtual network

Next, peer the AKS virtual network (VNet) with the Azure NetApp Files VNet by following these steps:

1. In the search box at the top of the Azure portal, type virtual networks.
2. Click VNet aks- vnet-name, then enter Peerings in the search field.
3. Click +Add and enter the information provided in the table below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value or description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peering link name</td>
<td>aks-vnet-name_to_anf</td>
</tr>
<tr>
<td>SubscriptionID</td>
<td>Subscription of the Azure NetApp Files VNet to which you’re peering</td>
</tr>
<tr>
<td>VNet peering partner</td>
<td>Azure NetApp Files VNet</td>
</tr>
</tbody>
</table>

Leave all the nonasterisk sections on default

4. Click ADD or OK to add the peering to the virtual network.

For more information, visit Create, change, or delete a virtual network peering.

Trident

Trident is an open-source project that NetApp maintains for application container persistent storage. Trident has been implemented as an external provisioner controller that runs as a pod itself, monitoring volumes and completely automating the provisioning process.

NetApp Trident enables smooth integration with K8s by creating and attaching persistent volumes for storing training datasets and trained models. This capability makes it easier for data scientists and data engineers to use K8s without the hassle of manually storing and managing datasets. Trident also eliminates the need for data scientists to learn managing new data platforms as it integrates the data management-related tasks through the logical API integration.

Install Trident

To install Trident software, complete the following steps:

1. First install helm.
2. Download and extract the Trident 21.01.1 installer.
3. Change the directory to trident-installer.

   cd trident-installer

4. Copy tridentctl to a directory in your system $PATH.

   cp ./tridentctl /usr/local/bin

5. Install Trident on K8s cluster with Helm:
   a. Change directory to helm directory.

      cd helm

   b. Install Trident.

      helm install trident trident-operator-21.01.1.tgz --namespace trident --create-namespace

   c. Check the status of Trident pods the usual K8s way:

      kubectl -n trident get pods

   d. If all the pods are up and running, Trident is installed and you are good to move forward.

Set up Azure NetApp Files back-end and storage class

To set up Azure NetApp Files back-end and storage class, complete the following steps:

1. Switch back to the home directory.

   cd ~

2. Clone the project repository lane-detection-SCNN-horovod.

3. Go to the trident-config directory.
4. Create an Azure Service Principle (the service principle is how Trident communicates with Azure to access your Azure NetApp Files resources).

```bash
az ad sp create-for-rbac --name
```

The output should look like the following example:

```json
{
  "appId": "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "displayName": "netapptrident",
  "name": "http://netapptrident",
  "password": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
}
```

5. Create the Trident backend json file.

6. Using your preferred text editor, complete the following fields from the table below inside the anf-backend.json file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionID</td>
<td>Your Azure Subscription ID</td>
</tr>
<tr>
<td>tenantID</td>
<td>Your Azure Tenant ID (from the output of az ad sp in the previous step)</td>
</tr>
<tr>
<td>clientId</td>
<td>Your appId (from the output of az ad sp in the previous step)</td>
</tr>
<tr>
<td>clientSecret</td>
<td>Your password (from the output of az ad sp in the previous step)</td>
</tr>
</tbody>
</table>

The file should look like the following example:
7. Instruct Trident to create the Azure NetApp Files back-end in the trident namespace, using anf-backend.json as the configuration file as follows:

```
tridentctl create backend -f anf-backend.json -n trident
```

8. Create the storage class:

a. K8s users provision volumes by using PVCs that specify a storage class by name. Instruct K8s to create a storage class azurenetappfiles that will reference the Azure NetApp Files back end created in the previous step using the following:

```
kubectl create -f anf-storage-class.yaml
```

b. Check that storage class is created by using the following command:

```
kubectl get sc azurenetappfiles
```

The output should look like the following example:

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROVISIONER</th>
<th>RECLAIMPOLICY</th>
<th>VOLUMEBINDINGMODE</th>
<th>ALLOWVOLUMEEXPANSION</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>azurenetappfiles</td>
<td>csi.trident.netapp.io</td>
<td>Delete</td>
<td>Immediate</td>
<td>false</td>
<td>98s</td>
</tr>
</tbody>
</table>

**Deploy and set up volume snapshot components on AKS**

If your cluster does not come pre-installed with the correct volume snapshot components, you may manually install these components by running the following steps:
AKS 1.18.14 does not have pre-installed Snapshot Controller.

1. Install Snapshot Beta CRDs by using the following commands:

```
kubectl create -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl create -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
kubectl create -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Install Snapshot Controller by using the following documents from GitHub:

```
```

3. Set up K8s volumesnapshotclass: Before creating a volume snapshot, a volume snapshot class must be set up. Create a volume snapshot class for Azure NetApp Files, and use it to achieve ML versioning by using NetApp Snapshot technology. Create volumesnapshotclass netapp-csi-snapclass and set it to default `volumesnapshotclass` as such:

```
kubectl create -f netapp-volume-snapshot-class.yaml
```

The output should look like the following example:

```
volumesnapshotclass.snapshot.storage.k8s.io/netapp-csi-snapclass created
```

4. Check that the volume Snapshot copy class was created by using the following command:

```
kubectl get volumesnapshotclass
```

The output should look like the following example:

```
NAME                DRIVER           DELETIONPOLICY     AGE
netapp-csi-snapclass csi.trident.netapp.io Delete 63s
```
To install RUN:AI, complete the following steps:

1. **Install RUN:AI cluster on AKS.**
2. Go to app.runai.ai, click create New Project, and name it lane-detection. It will create a namespace on a K8s cluster starting with `runai-` followed by the project name. In this case, the namespace created would be `runai-lane-detection`.

3. **Install RUN:AI CLI.**
4. On your terminal, set lane-detection as a default RUN:AI project by using the following command:

   ```sh
   runai config project lane-detection
   ```

   The output should look like the following example:

   ```text
   Project lane-detection has been set as default project
   ```
5. Create ClusterRole and ClusterRoleBinding for the project namespace (for example, lane-detection) so the default service account belonging to runai-lane-detection namespace has permission to perform volumesnapshot operations during job execution:
   a. List namespaces to check that runai-lane-detection exists by using this command:

   ```
kubectl get namespaces
   ```

   The output should appear like the following example:

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Active</td>
<td>130m</td>
</tr>
<tr>
<td>kube-node-lease</td>
<td>Active</td>
<td>130m</td>
</tr>
<tr>
<td>kube-public</td>
<td>Active</td>
<td>130m</td>
</tr>
<tr>
<td>kube-system</td>
<td>Active</td>
<td>130m</td>
</tr>
<tr>
<td>runai</td>
<td>Active</td>
<td>4m44s</td>
</tr>
<tr>
<td>runai-lane-detection</td>
<td>Active</td>
<td>13s</td>
</tr>
<tr>
<td>trident</td>
<td>Active</td>
<td>102m</td>
</tr>
</tbody>
</table>

6. Create ClusterRole netappsnapshot and ClusterRoleBinding netappsnapshot using the following commands:

   ```
   'kubectl create -f runai-project-snap-role.yaml'
   'kubectl create -f runai-project-snap-role-binding.yaml'
   ```

**Download and process the TuSimple dataset as RUN:AI job**

The process to download and process the TuSimple dataset as a RUN: AI job is optional. It involves the following steps:

1. Build and push the docker image, or omit this step if you want to use an existing docker image (for example, muneer7589/download-tusimple:1.0)
   a. Switch to the home directory:

   ```
   cd ~
   ```

   b. Go to the data directory of the project lane-detection-SCNN-horovod:

   ```
   cd ./lane-detection-SCNN-horovod/data
   ```

   c. Modify build_image.sh shell script and change docker repository to yours. For example, replace muneer7589 with your docker repository name. You could also change the docker image name and
d. Run the script to build the docker image and push it to the docker repository using these commands:

```
chmod +x build_image.sh
./build_image.sh
```

2. Submit the RUN: AI job to download, extract, pre-process, and store the TuSimple lane detection dataset in a `pvc`, which is dynamically created by NetApp Trident:

a. Use the following commands to submit the RUN: AI job:

```
runai submit
   --name download-tusimple-data
   --pvc azurenetappfiles:100Gi:/mnt
   --image muneer7589/download-tusimple:1.0
```

b. Enter the information from the table below to submit the RUN:AI job:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value or description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-name</td>
<td>Name of the job</td>
</tr>
<tr>
<td>Field</td>
<td>Value or description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
</tr>
<tr>
<td>-pvc</td>
<td>PVC of the format [StorageClassName]:Size:ContainerMountPath</td>
</tr>
<tr>
<td></td>
<td>In the above job submission, you are creating an PVC based on-demand using Trident with storage class azurenetappfiles. Persistent volume capacity here is 100Gi and it's mounted at path /mnt.</td>
</tr>
<tr>
<td>-image</td>
<td>Docker image to use when creating the container for this job</td>
</tr>
</tbody>
</table>

The output should look like the following example:

```
The job 'download-tusimple-data' has been submitted successfully
You can run `runai describe job download-tusimple-data -p lane-detection` to check the job status
```

c. List the submitted RUN:AI jobs.

```
rundai list jobs
```

d. Check the submitted job logs.

```
rundai logs download-tusimple-data -t 10
```

e. List the `pvc` created. Use this `pvc` command for training in the next step.

```
kubectl get pvc | grep download-tusimple-data
```

The output should look like the following example:

```
pvc-download-tusimple-data-0 Bound pvc-b8d3874d-2c17-48ca-a445-79f38d316d5 100Gi RWO azurenetappfiles 4m47s
```

f. Check the job in RUN: AI UI (or app.run.ai).
Perform distributed lane detection training using Horovod

Performing distributed lane detection training using Horovod is an optional process. However, here are the steps involved:

1. Build and push the docker image, or skip this step if you want to use the existing docker image (for example, muneer7589/dist-lane-detection:3.1):
   a. Switch to home directory.
      
      ```
      cd ~
      ```
   
   b. Go to the project directory lane-detection-SCNN-horovod.
      
      ```
      cd ./lane-detection-SCNN-horovod
      ```
   
   c. Modify the build_image.sh shell script and change docker repository to yours (for example, replace muneer7589 with your docker repository name). You could also change the docker image name and TAG (dist-lane-detection and 3.1, for example).
d. Run the script to build the docker image and push to the docker repository.

```bash
chmod +x build_image.sh
./build_image.sh
```

2. Submit the RUN: AI job for carrying out distributed training (MPI):
   a. Using submit of RUN: AI for automatically creating PVC in the previous step (for downloading data) only allows you to have RWO access, which does not allow multiple pods or nodes to access the same PVC for distributed training. Update the access mode to ReadWriteMany and use the Kubernetes patch to do so.
   b. First, get the volume name of the PVC by running the following command:

   ```bash
   kubectl get pvc | grep download-tusimple-data
   ```

   c. Patch the volume and update access mode to ReadWriteMany (replace volume name with yours in the following command):

   ```bash
   kubectl patch pv pvc-bb03b74d-2c17-40c4-a445-79f3de8d16d5 -p
   '{"spec":{"accessModes":["ReadWriteMany"]}}'
   ```

   d. Submit the RUN: AI MPI job for executing the distributed training job using information from the table below:
runai submit-mpi
--name dist-lane-detection-training
--large-shm
--processes=3
--gpu 1
--pvc pvc-download-tusimple-data-0:/mnt
--image muneer7589/dist-lane-detection:3.1
-e USE_WORKERS="true"
-e NUM_WORKERS=4
-e BATCH_SIZE=33
-e USE_VAL="false"
-e VAL_BATCH_SIZE=99
-e ENABLE_SNAPSHOT="true"
-e PVC_NAME="pvc-download-tusimple-data-0"

<table>
<thead>
<tr>
<th>Field</th>
<th>Value or description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the distributed training job</td>
</tr>
<tr>
<td>large shm</td>
<td>Mount a large /dev/shm device</td>
</tr>
<tr>
<td></td>
<td>It is a shared file system mounted on RAM and provides large enough shared memory for</td>
</tr>
<tr>
<td></td>
<td>multiple CPU workers to process and load batches into CPU RAM.</td>
</tr>
<tr>
<td>processes</td>
<td>Number of distributed training processes</td>
</tr>
<tr>
<td>gpu</td>
<td>Number of GPUs/processes to allocate for the job</td>
</tr>
<tr>
<td></td>
<td>In this job, there are three GPU worker processes (--processes=3), each allocated</td>
</tr>
<tr>
<td></td>
<td>with a single GPU (--gpu 1)</td>
</tr>
<tr>
<td>pvc</td>
<td>Use existing persistent volume (pvc-download-tusimple-data-0) created by previous</td>
</tr>
<tr>
<td></td>
<td>job (download-tusimple-data) and it is mounted at path /mnt</td>
</tr>
<tr>
<td>image</td>
<td>Docker image to use when creating the container for this job</td>
</tr>
<tr>
<td>Define environment variables to be set in the container</td>
<td></td>
</tr>
<tr>
<td>USE_WORKERS</td>
<td>Setting the argument to true turns on multiprocess data loading</td>
</tr>
<tr>
<td>NUM_WORKERS</td>
<td>Number of data loader worker processes</td>
</tr>
<tr>
<td>BATCH_SIZE</td>
<td>Training batch size</td>
</tr>
<tr>
<td>USE_VAL</td>
<td>Setting the argument to true allows validation</td>
</tr>
<tr>
<td>VAL_BATCH_SIZE</td>
<td>Validation batch size</td>
</tr>
<tr>
<td>Field</td>
<td>Value or description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENABLE_SNAPSHOT</td>
<td>Setting the argument to true enables taking data and trained model snapshots for ML versioning purposes</td>
</tr>
<tr>
<td>PVC_NAME</td>
<td>Name of the pvc to take a snapshot of. In the above job submission, you are taking a snapshot of pvc-download-tusimple-data-0, consisting of dataset and trained models</td>
</tr>
</tbody>
</table>

The output should look like the following example:

The job 'dist-lane-detection-training' has been submitted successfully
You can run `runai describe job dist-lane-detection-training -p lane-detection` to check the job status

e. List the submitted job.

```bash
runai list jobs
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>SERVICE URL(S)</th>
<th>STATUS</th>
<th>AGE</th>
<th>NODE</th>
<th>IMAGE</th>
<th>TYPE</th>
<th>PROJECT</th>
<th>USER</th>
<th>GPUs Allocated (Requested)</th>
<th>PODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>download-tusimple-data</td>
<td></td>
<td>Succeeded</td>
<td>1d</td>
<td>Id</td>
<td>munner7589/download-tusimple:1.0</td>
<td>Train</td>
<td>lane-detection</td>
<td>verronnortina</td>
<td>0 (0)</td>
<td>0</td>
</tr>
<tr>
<td>dist-lane-detection-training</td>
<td>Initiated/1</td>
<td>2m, multiple</td>
<td>munner7589/dist-lane-detection:3.1</td>
<td>Train</td>
<td>lane-detection</td>
<td>root</td>
<td>3 (3)</td>
<td>4 (0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

f. Submitted job logs:

```bash
runai logs dist-lane-detection-training
```

```
root@ai~:~ @runai# runai logs dist-lane-detection-training
Running with 3 workers
2021-03-06 17:29:23.158449: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcutdnn.so.10.1
  + POD_NAME=dist-lane-detection-training-worker-0
  + [ d = - ]
  + shift
  + /opt/kube/kubectl cp /opt/kube/hosts dist-lane-detection-training-worker-0:/etc/hosts_of_nodes
  + POD_NAME=dist-lane-detection-training-worker-2
  + [ d = - ]
  + shift
  + /opt/kube/kubectl cp /opt/kube/hosts dist-lane-detection-training-worker-2:/etc/hosts_of_nodes
  + POD_NAME=dist-lane-detection-training-worker-1
```

```bash
```
g. Check training job in RUN: AI GUI (or app.runai.ai): RUN: AI Dashboard, as seen in the figures below. The first figure details three GPUs allocated for the distributed training job spread across three nodes on AKS, and the second RUN:AI jobs:

```bash
```
h. After the training is finished, check the NetApp Snapshot copy that was created and linked with RUN: AI job.

```
runai logs dist-lane-detection-training --tail 1
```

```
kubectl get volumesnapshots | grep download-tusimple-data-0
```

![Image of a dashboard showing GPU and CPU utilization, and a cluster overview.]

```
kubectl get volumesnapshots | grep download-tusimple-data-0
```

![Image of a dashboard showing running jobs and pending jobs.]

```
kubectl get volumesnapshots | grep download-tusimple-data-0
```
Restore data from the NetApp Snapshot copy

To restore data from the NetApp Snapshot copy, complete the following steps:

1. Switch to home directory.
   ```bash
cd ~
   ```

2. Go to the project directory `lane-detection-SCNN-horovod`.
   ```bash
cd ./lane-detection-SCNN-horovod
   ```

3. Modify `restore-snapshot-pvc.yaml` and update `dataSource` name field to the Snapshot copy from which you want to restore data. You could also change PVC name where the data will be restored to, in this example its `restored-tusimple`.

   ```yaml
   apiVersion: v1
class: PersistentVolumeClaim
metadata:
  name: restored-tusimple
spec:
  storageClassName: azurestabfiles
  dataSource:
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteMany
resources:
  requests:
    storage: 100Gi
   ```

4. Create a new PVC by using `restore-snapshot-pvc.yaml`.
   ```bash
   kubectl create -f restore-snapshot-pvc.yaml
   ```
   The output should look like the following example:

   ```bash
   persistentvolumeclaim/restored-tusimple created
   ```

5. If you want to use the just restored data for training, job submission remains the same as before; only replace the `PVC_NAME` with the restored `PVC_NAME` when submitting the training job, as seen in the following commands:
runai submit-mpi
   --name dist-lane-detection-training
   --large-shm
   --processes=3
   --gpu 1
   --pvc restored-tusimple:/mnt
   --image muneer7589/dist-lane-detection:3.1
   -e USE_WORKERS="true"
   -e NUM_WORKERS=4
   -e BATCH_SIZE=33
   -e USE_VAL="false"
   -e VAL_BATCH_SIZE=99
   -e ENABLE_SNAPSHOT="true"
   -e PVC_NAME="restored-tusimple"

Performance evaluation

To show the linear scalability of the solution, performance tests have been done for two scenarios: one GPU and three GPUs. GPU allocation, GPU and memory utilization, different single- and three- node metrics have been captured during the training on the TuSimple lane detection dataset. Data is increased five- fold just for the sake of analyzing resource utilization during the training processes.

The solution enables customers to start with a small dataset and a few GPUs. When the amount of data and the demand of GPUs increase, customers can dynamically scale out the terabytes in the Standard Tier and quickly scale up to the Premium Tier to get four times the throughput per terabyte without moving any data. This process is further explained in the section, Azure NetApp Files service levels.

Processing time on one GPU was 12 hours and 45 minutes. Processing time on three GPUs across three nodes was approximately 4 hours and 30 minutes.

The figures shown throughout the remainder of this document illustrate examples of performance and scalability based on individual business needs.

The figure below illustrates 1 GPU allocation and memory utilization.
The figure below illustrates single node GPU utilization.

The figure below illustrates single node memory size (16GB).
The figure below illustrates single node GPU count (1).

The figure below illustrates single node GPU allocation (%).

The figure below illustrates three GPUs across three nodes – GPUs allocation and memory.
The figure below illustrates three GPUs across three nodes utilization (%).

The figure below illustrates three GPUs across three nodes memory utilization (%).
**Azure NetApp Files service levels**

You can change the service level of an existing volume by moving the volume to another capacity pool that uses the service level you want for the volume. This existing service-level change for the volume does not require that you migrate data. It also does not affect access to the volume.

**Dynamically change the service level of a volume**

To change the service level of a volume, use the following steps:

1. On the Volumes page, right-click the volume whose service level you want to change. Select Change Pool.

2. In the Change Pool window, select the capacity pool you want to move the volume to. Then, click OK.
Automate service level change

Dynamic Service Level change is currently still in Public Preview, but it is not enabled by default. To enable this feature on the Azure subscription, follow these steps provided in the document “Dynamically change the service level of a volume.”

• You can also use the following commands for Azure: CLI. For more information about changing the pool size of Azure NetApp Files, visit az netappfiles volume: Manage Azure NetApp Files (ANF) volume resources.

```bash
az netappfiles volume pool-change -g mygroup
--account-name myaccname
-pool-name mypoolname
--name myvolname
--new-pool-resource-id mynewresourceid
```

• The `set-aznetappfilesvolumepool` cmdlet shown here can change the pool of an Azure NetApp Files volume. More information about changing volume pool size and Azure PowerShell can be found by visiting Change pool for an Azure NetApp Files volume.
Conclusion

NetApp and RUN: AI have partnered in the creation of this technical report to demonstrate the unique capabilities of the Azure NetApp Files together with the RUN: AI platform for simplifying orchestration of AI workloads. This technical report provides a reference architecture for streamlining the process of both data pipelines and workload orchestration for distributed lane detection training.

In conclusion, with regard to distributed training at scale (especially in a public cloud environment), the resource orchestration and storage component is a critical part of the solution. Making sure that data managing never hinders multiple GPU processing, therefore results in the optimal utilization of GPU cycles. Thus, making the system as cost effective as possible for large-scale distributed training purposes.

Data fabric delivered by NetApp overcomes the challenge by enabling data scientists and data engineers to connect together on-premises and in the cloud to have synchronous data, without performing any manual intervention. In other words, data fabric smooths the process of managing AI workflow spread across multiple locations. It also facilitates on demand-based data availability by bringing data close to compute and performing analysis, training, and validation wherever and whenever needed. This capability not only enables data integration but also protection and security of the entire data pipeline.

Additional information

To learn more about the information that is described in this document, review the following documents and/or websites:

- Dataset: TuSimple
  https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection

- Deep Learning Network Architecture: Spatial Convolutional Neural Network
  https://arxiv.org/abs/1712.06080

- Distributed deep learning training framework: Horovod
  https://horovod.ai/

- RUN: AI container orchestration solution: RUN: AI product introduction
  https://docs.run.ai/home/components/

- RUN: AI installation documentation
TR-4841: Hybrid Cloud AI Operating System with Data Caching

Rick Huang, David Arnette, NetApp
Yochay Ettun, cnvrg.io

The explosive growth of data and the exponential growth of ML and AI have converged to create a zettabyte economy with unique development and implementation challenges.

Although it is a widely known that ML models are data-hungry and require high-performance data storage proximal to compute resources, in practice, it is not so straightforward to implement this model, especially with hybrid cloud and elastic compute instances. Massive quantities of data are usually stored in low-cost data lakes, where high-performance AI compute resources such as GPUs cannot efficiently access it. This problem is aggravated in a hybrid-cloud infrastructure where some workloads operate in the cloud and some are located on-premises or in a different HPC environment entirely.

In this document, we present a novel solution that allows IT professionals and data engineers to create a truly hybrid cloud AI platform with a topology-aware data hub that enables data scientists to instantly and automatically create a cache of their datasets in proximity to their compute resources, wherever they are...
located. As a result, not only can high-performance model training be accomplished, but additional benefits are created, including the collaboration of multiple AI practitioners, who have immediate access to dataset caches, versions, and lineages within a dataset version hub.

Use Case Overview and Problem Statement

Datasets and dataset versions are typically located in a data lake, such as NetApp StorageGrid object-based storage, which offers reduced cost and other operational advantages. Data scientists pull these datasets and engineer them in multiple steps to prepare them for training with a specific model, often creating multiple versions along the way. As the next step, the data scientist must pick optimized compute resources (GPUs, high-end CPU instances, an on-premises cluster, and so on) to run the model. The following figure depicts the lack of dataset proximity in an ML compute environment.

However, multiple training experiments must run in parallel in different compute environments, each of which require a download of the dataset from the data lake, which is an expensive and time-consuming process. Proximity of the dataset to the compute environment (especially for a hybrid cloud) is not guaranteed. In addition, other team members that run their own experiments with the same dataset must go through the same arduous process. Beyond the obvious slow data access, challenges include difficulties tracking dataset versions, dataset sharing, collaboration, and reproducibility.

Customer Requirements

Customer requirements can vary in order to achieve high-performance ML runs while efficiently using resources; for example, customers might require the following:

- Fast access to datasets from each compute instance executing the training model without incurring expensive downloads and data access complexities
• The use any compute instance (GPU or CPU) in the cloud or on-premises without concern for the location of the datasets
• Increased efficiency and productivity by running multiple training experiments in parallel with different compute resources on the same dataset without unnecessary delays and data latency
• Minimized compute instance costs
• Improved reproducibility with tools to keep records of the datasets, their lineage, versions, and other metadata details
• Enhanced sharing and collaboration so that any authorized member of the team can access the datasets and run experiments

To implement dataset caching with NetApp ONTAP data management software, customers must perform the following tasks:

• Configure and set the NFS storage that is closest to the compute resources.
• Determine which dataset and version to cache.
• Monitor the total memory committed to cached datasets and how much NFS storage is available for additional cache commits (for example, cache management).
• Age out of datasets in the cache if they have not been used in certain time. The default is one day; other configuration options are available.

Solution Overview

This section reviews a conventional data science pipeline and its drawbacks. It also presents the architecture of the proposed dataset caching solution.

Conventional Data Science Pipeline and Drawbacks

A typical sequence of ML model development and deployment involves iterative steps that include the following:

• Ingesting data
• Data preprocessing (creating multiple versions of the datasets)
• Running multiple experiments involving hyperparameter optimization, different models, and so on
• Deployment
• Monitoring cnvrg.io has developed a comprehensive platform to automate all tasks from research to deployment. A small sample of dashboard screenshots pertaining to the pipeline is shown in the following figure.
It is very common to have multiple datasets in play from public repositories and private data. In addition, each dataset is likely to have multiple versions resulting from dataset cleanup or feature engineering. A dashboard that provides a dataset hub and a version hub is needed to make sure collaboration and consistency tools are available to the team, as can be seen in the following figure.
The next step in the pipeline is training, which requires multiple parallel instances of training models, each associated with a dataset and a certain compute instance. The binding of a dataset to a certain experiment with a certain compute instance is a challenge because it is possible that some experiments are performed by GPU instances from Amazon Web Services (AWS), while other experiments are performed by DGX-1 or DGX-2 instances on-premises. Other experiments might be executed in CPU servers in GCP, while the dataset location is not in reasonable proximity to the compute resources performing the training. A reasonable proximity would have full 10GbE or more low-latency connectivity from the dataset storage to the compute instance.

It is a common practice for data scientists to download the dataset to the compute instance performing the training and execute the experiment. However, there are several potential problems with this approach:

- When the data scientist downloads the dataset to a compute instance, there are no guarantees that the integrated compute storage is high performance (an example of a high-performance system would be the ONTAP AFF A800 NVMe solution).
- When the downloaded dataset resides in one compute node, storage can become a bottleneck when distributed models are executed over multiple nodes (unlike with NetApp ONTAP high-performance distributed storage).
- The next iteration of the training experiment might be performed in a different compute instance due to queue conflicts or priorities, again creating significant network distance from the dataset to the compute location.
- Other team members executing training experiments on the same compute cluster cannot share this dataset; each performs the (expensive) download of the dataset from an arbitrary location.
- If other datasets or versions of the same dataset are needed for the subsequent training jobs, the data scientists must again perform the (expensive) download of the dataset to the compute instance performing the training. NetApp and cnvrg.io have created a new dataset caching solution that eliminates these
hurdles. The solution creates accelerated execution of the ML pipeline by caching hot datasets on the ONTAP high-performance storage system. With ONTAP NFS, the datasets are cached once (and only once) in a data fabric powered by NetApp (such as AFF A800), which is collocated with the compute. As the NetApp ONTAP NFS high-speed storage can serve multiple ML compute nodes, the performance of the training models is optimized, bringing cost savings, productivity, and operational efficiency to the organization.

Solution Architecture

This solution from NetApp and cnvrg.io provides dataset caching, as shown in the following figure. Dataset caching allows data scientists to pick a desired dataset or dataset version and move it to the ONTAP NFS cache, which lies in proximity to the ML compute cluster. The data scientist can now run multiple experiments without incurring delays or downloads. In addition, all collaborating engineers can use the same dataset with the attached compute cluster (with the freedom to pick any node) without additional downloads from the data lake. The data scientists are offered a dashboard that tracks and monitors all datasets and versions and provides a view of which datasets were cached.

The cnvrg.io platform auto-detects aged datasets that have not been used for a certain time and evicts them from the cache, which maintains free NFS cache space for more frequently used datasets. It is important to note that dataset caching with ONTAP works in the cloud and on-premises, thus providing maximum flexibility.

Concepts and Components

This section covers concepts and components associated with data caching in an ML workflow.
Machine Learning

ML is rapidly becoming essential to many businesses and organizations around the world. Therefore, IT and DevOps teams are now facing the challenge of standardizing ML workloads and provisioning cloud, on-premises, and hybrid compute resources that support the dynamic and intensive workflows that ML jobs and pipelines require.

Container-Based Machine Learning and Kubernetes

Containers are isolated user-space instances that run on top of a shared host operating system kernel. The adoption of containers is rapidly increasing. Containers offer many of the same application sandboxing benefits that virtual machines (VMs) offer. However, because the hypervisor and guest operating system layers that VMs rely on have been eliminated, containers are far more lightweight.

Containers also allow the efficient packaging of application dependencies, run times, and so on directly with an application. The most commonly used container packaging format is the Docker container. An application that has been containerized in the Docker container format can be executed on any machine that can run Docker containers. This is true even if the application’s dependencies are not present on the machine, because all dependencies are packaged in the container itself. For more information, visit the Docker website.

Kubernetes, the popular container orchestrator, allows data scientists to launch flexible, container-based jobs and pipelines. It also enables infrastructure teams to manage and monitor ML workloads in a single managed and cloud-native environment. For more information, visit the Kubernetes website.

cnvrg.io

cnvrg.io is an AI operating system that transforms the way enterprises manage, scale, and accelerate AI and data science development from research to production. The code-first platform is built by data scientists for data scientists and offers flexibility to run on-premises or in the cloud. With model management, MLOps, and continual ML solutions, cnvrg.io brings top-of-the-line technology to data science teams so they can spend less time on DevOps and focus on the real magic—algorithms. Since using cnvrg.io, teams across industries have gotten more models to production resulting in increased business value.

cnvrg.io Meta-Scheduler

cnvrg.io has a unique architecture that allows IT and engineers to attach different compute resources to the same control plane and have cnvrg.io manage ML jobs across all resources. This means that IT can attach multiple on-premises Kubernetes clusters, VM servers, and cloud accounts and run ML workloads on all resources, as shown in the following figure.
cnvrg.io Data Caching

cnvrg.io allows data scientists to define hot and cold dataset versions with its data-caching technology. By default, datasets are stored in a centralized object storage database. Then, data scientists can cache a specific data version on the selected compute resource to save time on download and thereby increase ML development and productivity. Datasets that are cached and are not in use for a few days are automatically cleared from the selected NFS. Caching and clearing the cache can be performed with a single click; no coding, IT, or DevOps work is required.

cnvrg.io Flows and ML Pipelines

cnvrg.io Flows is a tool for building production ML pipelines. Each component in a flow is a script/code running on a selected compute with a base docker image. This design enables data scientists and engineers to build a single pipeline that can run both on-premises and in the cloud. cnvrg.io makes sure data, parameters, and artifacts are moving between the different components. In addition, each flow is monitored and tracked for 100% reproducible data science.

cnvrg.io CORE

cnvrg.io CORE is a free platform for the data science community to help data scientists focus more on data science and less on DevOps. CORE’s flexible infrastructure gives data scientists the control to use any language, AI framework, or compute environment whether on-premises or in the cloud so they can do what they do best, build algorithms. cnvrg.io CORE can be easily installed with a single command on any Kubernetes cluster.

NetApp ONTAP AI

ONTAP AI is a data center reference architecture for ML and deep learning (DL) workloads that uses NetApp AFF storage systems and NVIDIA DGX systems with Tesla V100 GPUs. ONTAP AI is based on the industry-standard NFS file protocol over 100Gb Ethernet, providing customers with a high-performance ML/DL infrastructure that uses standard data center technologies to reduce implementation and administration overhead. Using standardized network and protocols enables ONTAP AI to integrate into hybrid cloud environments while maintaining operational consistency and simplicity. As a prevalidated infrastructure solution, ONTAP AI reduces deployment time and risk and reduces administration overhead significantly, allowing customers to realize faster time to value.

NVIDIA DeepOps

DeepOps is an open source project from NVIDIA that, by using Ansible, automates the deployment of GPU server clusters according to best practices. DeepOps is modular and can be used for various deployment tasks. For this document and the validation exercise that it describes, DeepOps is used to deploy a Kubernetes cluster that consists of GPU server worker nodes. For more information, visit the DeepOps website.

NetApp Trident

Trident is an open source storage orchestrator developed and maintained by NetApp that greatly simplifies the creation, management, and consumption of persistent storage for Kubernetes workloads. Trident itself a Kubernetes-native application—it runs directly within a Kubernetes cluster. With Trident, Kubernetes users (developers, data scientists, Kubernetes administrators, and so on) can create, manage, and interact with persistent storage volumes in the standard Kubernetes format that they are already familiar with. At the same time, they can take advantage of NetApp advanced data management capabilities and a data fabric that is powered by NetApp technology. Trident abstracts away the complexities of persistent storage and makes it simple to consume. For more information, visit the Trident website.
NetApp StorageGRID

NetApp StorageGRID is a software-defined object storage platform designed to meet these needs by providing simple, cloud-like storage that users can access using the S3 protocol. StorageGRID is a scale-out system designed to support multiple nodes across internet-connected sites, regardless of distance. With the intelligent policy engine of StorageGRID, users can choose erasure-coding objects across sites for geo-resiliency or object replication between remote sites to minimize WAN access latency. StorageGrid provides an excellent private-cloud primary object storage data lake in this solution.

NetApp Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP data management software delivers control, protection, and efficiency to user data with the flexibility of public cloud providers including AWS, Google Cloud Platform, and Microsoft Azure. Cloud Volumes ONTAP is cloud-native data management software built on the NetApp ONTAP storage software, providing users with a superior universal storage platform that addresses their cloud data needs. Having the same storage software in the cloud and on-premises provides users with the value of a data fabric without having to train IT staff in all-new methods to manage data.

For customers that are interested in hybrid cloud deployment models, Cloud Volumes ONTAP can provide the same capabilities and class-leading performance in most public clouds to provide a consistent and seamless user experience in any environment.

Hardware and Software Requirements

This section covers the technology requirements for the ONTAP AI solution.

Hardware Requirements

Although hardware requirements depend on specific customer workloads, ONTAP AI can be deployed at any scale for data engineering, model training, and production inferencing from a single GPU up to rack-scale configurations for large-scale ML/DL operations. For more information about ONTAP AI, see the ONTAP AI website.

This solution was validated using a DGX-1 system for compute, a NetApp AFF A800 storage system, and Cisco Nexus 3232C for network connectivity. The AFF A800 used in this validation can support as many as 10 DGX-1 systems for most ML/DL workloads. The following figure shows the ONTAP AI topology used for model training in this validation.
To extend this solution to a public cloud, Cloud Volumes ONTAP can be deployed alongside cloud GPU compute resources and integrated into a hybrid cloud data fabric that enables customers to use whatever resources are appropriate for any given workload.

Software Requirements

The following table shows the specific software versions used in this solution validation.

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>18.04.4 LTS</td>
</tr>
<tr>
<td>NVIDIA DGX OS</td>
<td>4.4.0</td>
</tr>
<tr>
<td>NVIDIA DeepOps</td>
<td>20.02.1</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>1.15</td>
</tr>
<tr>
<td>Helm</td>
<td>3.1.0</td>
</tr>
<tr>
<td>cnvrg.io</td>
<td>3.0.0</td>
</tr>
<tr>
<td>NetApp ONTAP</td>
<td>9.6P4</td>
</tr>
</tbody>
</table>

For this solution validation, Kubernetes was deployed as a single-node cluster on the DGX-1 system. For large-scale deployments, independent Kubernetes master nodes should be deployed to provide high availability of management services as well as reserve valuable DGX resources for ML and DL workloads.

Solution Deployment and Validation Details

The following sections discuss the details of solution deployment and validation.

ONTAP AI Deployment

Deployment of ONTAP AI requires the installation and configuration of networking, compute, and storage hardware. Specific instructions for deployment of the ONTAP AI infrastructure are beyond the scope of this document. For detailed deployment
information, see NVA-1121-DEPLOY: NetApp ONTAP AI, Powered by NVIDIA.

For this solution validation, a single volume was created and mounted to the DGX-1 system. That mount point was then mounted to the containers to make data accessible for training. For large-scale deployments, NetApp Trident automates the creation and mounting of volumes to eliminate administrative overhead and enable end-user management of resources.

Kubernetes Deployment

To deploy and configure your Kubernetes cluster with NVIDIA DeepOps, perform the following tasks from a deployment jump host:

1. Download NVIDIA DeepOps by following the instructions on the Getting Started page on the NVIDIA DeepOps GitHub site.

2. Deploy Kubernetes in your cluster by following the instructions on the Kubernetes Deployment Guide on the NVIDIA DeepOps GitHub site.

   For the DeepOps Kubernetes deployment to work, the same user must exist on all Kubernetes master and worker nodes.

If the deployment fails, change the value of kubectl_localhost to false in deepops/config/group_vars/k8s-cluster.yml and repeat step 2. The Copy kubectl binary to ansible host task, which executes only when the value of kubectl_localhost is true, relies on the fetch Ansible module, which has known memory usage issues. These memory usage issues can sometimes cause the task to fail. If the task fails because of a memory issue, then the remainder of the deployment operation does not complete successfully.

If the deployment completes successfully after you have changed the value of kubectl_localhost to false, then you must manually copy the kubectl binary from a Kubernetes master node to the deployment jump host. You can find the location of the kubectl binary on a specific master node by running the which kubectl command directly on that node.

cnvrg.io Deployment

Deploy cnvrg CORE Using Helm

Helm is the easiest way to quickly deploy cnvrg using any cluster, on-premises, Minikube, or on any cloud cluster (such as AKS, EKS, and GKE). This section describes how cnvrg was installed on an on-premises (DGX-1) instance with Kubernetes installed.

Prerequisites

Before you can complete the installation, you must install and prepare the following dependencies on your local machine:

- Kubectl
- Helm 3.x
- Kubernetes cluster 1.15+


Deploy Using Helm

1. To download the most updated cnvrg helm charts, run the following command:

```
helm repo add cnvrg https://helm.cnvrg.io
helm repo update
```

2. Before you deploy cnvrg, you need the external IP address of the cluster and the name of the node on which you will deploy cnvrg. To deploy cnvrg on an on-premises Kubernetes cluster, run the following command:

```
helm install cnvrg cnvrg/cnvrg --timeout 1500s --wait \ --set
  global.external_ip=<ip_of_cluster> \ --set global.node=<name_of_node>
```

3. Run the `helm install` command. All the services and systems automatically install on your cluster. The process can take up to 15 minutes.

4. The `helm install` command can take up to 10 minutes. When the deployment completes, go to the URL of your newly deployed cnvrg or add the new cluster as a resource inside your organization. The `helm` command informs you of the correct URL.

```
Thank you for installing cnvrg.io!
Your installation of cnvrg.io is now available, and can be reached via:
Talk to our team via email at
```

5. When the status of all the containers is running or complete, cnvrg has been successfully deployed. It should look similar to the following example output:

```
<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnvrg-app-69fbb9df98-6xrgf</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>cnvrg-sidekiq-b9d54d889-5x4fc</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>controller-65895b47d4-s96v6</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>init-app-vs-config-wv9c4</td>
<td>0/1</td>
<td>Completed</td>
<td>0</td>
<td>9m</td>
</tr>
<tr>
<td>init-gateway-vs-config-2zbpp</td>
<td>0/1</td>
<td>Completed</td>
<td>0</td>
<td>9m</td>
</tr>
<tr>
<td>init-minio-vs-config-cd2rg</td>
<td>0/1</td>
<td>Completed</td>
<td>0</td>
<td>9m</td>
</tr>
<tr>
<td>minio-0</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>postgres-0</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>redis-695c49c986-kcbt9</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>seeder-wh655</td>
<td>0/1</td>
<td>Completed</td>
<td>0</td>
<td>2m</td>
</tr>
<tr>
<td>speaker-5sghr</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
</tbody>
</table>
```

Computer Vision Model Training with ResNet50 and the Chest X-ray Dataset

cnvrg.io AI OS was deployed on a Kubernetes setup on a NetApp ONTAP AI architecture powered by the NVIDIA DGX system. For validation, we used the NIH Chest X-ray dataset consisting of de-identified images of
chest x-rays. The images were in the PNG format. The data was provided by the NIH Clinical Center and is available through the NIH download site. We used a 250GB sample of the data with 627,615 images across 15 classes.

The dataset was uploaded to the cnvrg platform and was cached on an NFS export from the NetApp AFF A800 storage system.

Set up the Compute Resources

The cnvrg architecture and meta-scheduling capability allow engineers and IT professionals to attach different compute resources to a single platform. In our setup, we used the same cluster cnvrg that was deployed for running the deep-learning workloads. If you need to attach additional clusters, use the GUI, as shown in the following screenshot.

Load Data

To upload data to the cnvrg platform, you can use the GUI or the cnvrg CLI. For large datasets, NetApp recommends using the CLI because it is a strong, scalable, and reliable tool that can handle a large number of files.

To upload data, complete the following steps:

1. Download the cnvrg CLI.
2. navigate to the x-ray directory.
3. Initialize the dataset in the platform with the cnvrg data init command.
4. Upload all contents of the directory to the central data lake with the cnvrg data sync command. After the data is uploaded to the central object store (StorageGRID, S3, or others), you can browse with the GUI. The following figure shows a loaded chest X-ray fibrosis image PNG file. In addition, cnvrg versions the data so that any model you build can be reproduced down to the data version.
Cach Data

To make training faster and avoid downloading 600k+ files for each model training and experiment, we used the data-caching feature after data was initially uploaded to the central data-lake object store.
After users click Cache, cnvrg downloads the data in its specific commit from the remote object store and caches it on the ONTAP NFS volume. After it completes, the data is available for instant training. In addition, if the data is not used for a few days (for model training or exploration, for example), cnvrg automatically clears the cache.

**Build an ML Pipeline with Cached Data**

cnvrg flows allows you to easily build production ML pipelines. Flows are flexible, can work for any kind of ML use case, and can be created through the GUI or code. Each component in a flow can run on a different compute resource with a different Docker image, which makes it possible to build hybrid cloud and optimized ML pipelines.
Building the Chest X-ray Flow: Setting Data

We added our dataset to a newly created flow. When adding the dataset, you can select the specific version (commit) and indicate whether you want the cached version. In this example, we selected the cached commit.
Building the Chest X-ray Flow: Setting Training Model: ResNet50

In the pipeline, you can add any kind of custom code you want. In cnvrg, there is also the AI library, a reusable ML components collection. In the AI library, there are algorithms, scripts, data sources, and other solutions that can be used in any ML or deep learning flow. In this example, we selected the prebuilt ResNet50 module. We used default parameters such as batch_size:128, epochs:10, and more. These parameters can be viewed in the AI Library docs. The following screenshot shows the new flow with the X-ray dataset connected to ResNet50.
Define the Compute Resource for ResNet50

Each algorithm or component in cnvrg flows can run on a different compute instance, with a different Docker image. In our setup, we wanted to run the training algorithm on the NVIDIA DGX systems with the NetApp ONTAP AI architecture. In the following figure, we selected `gpu-real`, which is a compute template and specification for our on-premises cluster. We also created a queue of templates and selected multiple templates. In this way, if the `gpu-real` resource cannot be allocated (if, for example, other data scientists are using it), then you can enable automatic cloud-bursting by adding a cloud provider template. The following screenshot shows the use of `gpu-real` as a compute node for ResNet50.
Tracking and Monitoring Results

After a flow is executed, cnvrg triggers the tracking and monitoring engine. Each run of a flow is automatically documented and updated in real time. Hyperparameters, metrics, resource usage (GPU utilization, and more), code version, artifacts, logs, and so on are automatically available in the Experiments section, as shown in the following two screenshots.
Conclusion

NetApp and cnvrg.io have partnered to offer customers a complete data management solution for ML and DL software development. ONTAP AI provides high-performance compute and storage for any scale of operation, and cnvrg.io software streamlines data science workflows and improves resource utilization.

Acknowledgments

- Mike Oglesby, Technical Marketing Engineer, NetApp
- Santosh Rao, Senior Technical Director, NetApp

Where to Find Additional Information

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

- Cnvrg.io ([https://cnvrg.io](https://cnvrg.io)):
  - Cnvrg CORE (free ML platform)
    - [https://cnvrg.io/platform/core](https://cnvrg.io/platform/core)
  - Cnvrg docs
    - [https://app.cnvrg.io/docs](https://app.cnvrg.io/docs)

- NVIDIA DGX-1 servers:
  - NVIDIA DGX-1 servers
  - NVIDIA Tesla V100 Tensor Core GPU
  - NVIDIA GPU Cloud (NGC)

- NetApp AFF systems:
  - AFF datasheet
    - [https://www.netapp.com/us/media/d-3582.pdf](https://www.netapp.com/us/media/d-3582.pdf)
  - NetApp FlashAdvantage for AFF
  - ONTAP 9.x documentation
- NetApp FlexGroup technical report

- NetApp persistent storage for containers:
  - NetApp Trident
    https://netapp.io/persistent-storage-provisioner-for-kubernetes/

- NetApp Interoperability Matrix:
  - NetApp Interoperability Matrix Tool
    http://support.netapp.com/matrix

- ONTAP AI networking:
  - Cisco Nexus 3232C Switches
  - Mellanox Spectrum 2000 series switches

- ML framework and tools:
  - DALI
    https://github.com/NVIDIA/DALI
    https://www.tensorflow.org/
    https://eng.uber.com/horovod/
  - Enabling GPUs in the Container Runtime Ecosystem
    https://devblogs.nvidia.com/gpu-containers-runtime/
  - Docker
    https://docs.docker.com
  - Kubernetes
    https://kubernetes.io/docs/home/
  - NVIDIA DeepOps
    https://github.com/NVIDIA/deepops
  - Kubeflow
Several emerging application scenarios, such as advanced driver-assistance systems (ADAS), Industry 4.0, smart cities, and Internet of Things (IoT), require the processing of continuous data streams under a near-zero latency. This document describes a compute and storage architecture to deploy GPU-based artificial intelligence (AI) inferencing on NetApp storage controllers and Lenovo ThinkSystem servers in an edge environment that meets these requirements. This document also provides performance data for the industry standard MLPerf Inference benchmark, evaluating various inference tasks on edge servers equipped with NVIDIA T4 GPUs. We investigate the performance of offline, single stream, and multistream inference scenarios and show that the architecture with a cost-effective shared networked storage system is highly performant and provides a central point for data and model management for multiple edge servers.

Introduction

Companies are increasingly generating massive volumes of data at the network edge. To achieve maximum value from smart sensors and IoT data, organizations are looking for a real-time event streaming solution that enables edge computing. Computationally demanding jobs are therefore increasingly performed at the edge, outside of data centers. AI inference is one of the drivers of this trend. Edge servers provide sufficient computational power for these workloads, especially when using accelerators, but limited storage is often an issue, especially in multiserver environments. In this document we show how you can deploy a shared storage system in the edge environment and how it benefits AI inference workloads without imposing a performance penalty.

This document describes a reference architecture for AI inference at the edge. It combines multiple Lenovo ThinkSystem edge servers with a NetApp storage system to create a solution that is easy to deploy and manage. It is intended to be a baseline guide for practical deployments in various situations, such as the factory floor with multiple cameras and industrial sensors, point-of-sale (POS) systems in retail transactions, or Full Self-Driving (FSD) systems that identify visual anomalies in autonomous vehicles.

This document covers testing and validation of a compute and storage configuration consisting of Lenovo ThinkSystem SE350 Edge Server and an entry-level NetApp AFF and EF-Series storage system. The reference architectures provide an efficient and cost-effective solution for AI deployments while also providing
comprehensive data services, integrated data protection, seamless scalability, and cloud connected data storage with NetApp ONTAP and NetApp SANtricity data management software.

**Target audience**

This document is intended for the following audiences:

- Business leaders and enterprise architects who want to productize AI at the edge.
- Data scientists, data engineers, AI/machine learning (ML) researchers, and developers of AI systems.
- Enterprise architects who design solutions for the development of AI/ML models and applications.
- Data scientists and AI engineers looking for efficient ways to deploy deep learning (DL) and ML models.
- Edge device managers and edge server administrators responsible for deployment and management of edge inferencing models.

**Solution architecture**

This Lenovo ThinkSystem server and NetApp ONTAP or NetApp SANtricity storage solution is designed to handle AI inferencing on large datasets using the processing power of GPUs alongside traditional CPUs. This validation demonstrates high performance and optimal data management with an architecture that uses either single or multiple Lenovo SR350 edge servers interconnected with a single NetApp AFF storage system, as shown in the following two figures.

![Solution architecture diagram](image-url)
The logical architecture overview in the following figure shows the roles of the compute and storage elements in this architecture. Specifically, it shows the following:

- Edge compute devices performing inference on the data it receives from cameras, sensors, and so on.
- A shared storage element that serves multiple purposes:
  - Provides a central location for inference models and other data needed to perform the inference. Compute servers access the storage directly and use inference models across the network without the need to copy them locally.
  - Updated models are pushed here.
  - Archives input data that edge servers receive for later analysis. For example, if the edge devices are connected to cameras, the storage element keeps the videos captured by the cameras.
This NetApp and Lenovo solution offers the following key benefits:

- GPU accelerated computing at the edge.
- Deployment of multiple edge servers backed and managed from a shared storage.
- Robust data protection to meet low recovery point objectives (RPOs) and recovery time objectives (RTOs) with no data loss.
- Optimized data management with NetApp Snapshot copies and clones to streamline development workflows.

**How to use this architecture**

This document validates the design and performance of the proposed architecture. However, we have not tested certain software-level pieces, such as container, workload, or model management and data synchronization with cloud or data center on-premises, because they are specific to a deployment scenario. Here, multiple choices exist.

At the container management level, Kubernetes container management is a good choice and is well supported in either a fully upstream version (Canonical) or in a modified version suitable for enterprise deployments (Red Hat). The NetApp AI Control Plane which uses NetApp Trident and the newly added NetApp DataOps Toolkit provides built-in traceability, data management functions, interfaces, and tools for data scientists and data engineers to integrate with NetApp storage. Kubeflow, the ML toolkit for Kubernetes, provides additional AI capabilities along with a support for model versioning and KFServing on several platforms such as TensorFlow Serving or NVIDIA Triton Inference Server. Another option is NVIDIA EGX platform, which provides workload management along with access to a catalog of GPU-enabled AI inference containers. However, these options might require significant effort and expertise to put them into production and might require the assistance of a third-party independent software vendor (ISV) or consultant.

**Solution areas**

The key benefit of AI inferencing and edge computing is the ability of devices to compute, process, and analyze data with a high level of quality without latency. There are far too many examples of edge computing use cases to describe in this document, but here are a few prominent ones:
**Automobiles: Autonomous vehicles**

The classic edge computing illustration is in the advanced driver-assistance systems (ADAS) in autonomous vehicles (AV). The AI in driverless cars must rapidly process a lot of data from cameras and sensors to be a successful safe driver. Taking too long to interpret between an object and a human can mean life or death, therefore being able to process that data as close to the vehicle as possible is crucial. In this case, one or more edge compute servers handles the input from cameras, RADAR, LiDAR, and other sensors, while shared storage holds inference models and stores input data from sensors.

**Healthcare: Patient monitoring**

One of the greatest impacts of AI and edge computing is its ability to enhance continuous monitoring of patients for chronic diseases both in at-home care and intensive care units (ICUs). Data from edge devices that monitor insulin levels, respiration, neurological activity, cardiac rhythm, and gastrointestinal functions require instantaneous analysis of data that must be acted on immediately because there is limited time to act to save someone’s life.

**Retail: Cashier-less payment**

Edge computing can power AI and ML to help retailers reduce checkout time and increase foot traffic. Cashier-less systems support various components, such as the following:

- Authentication and access. Connecting the physical shopper to a validated account and permitting access to the retail space.
- Inventory monitoring. Using sensors, RFID tags, and computer vision systems to help confirm the selection or deselection of items by shoppers.

Here, each of the edge servers handle each checkout counter and the shared storage system serves as a central synchronization point.

**Financial services: Human safety at kiosks and fraud prevention**

Banking organizations are using AI and edge computing to innovate and create personalized banking experiences. Interactive kiosks using real-time data analytics and AI inferencing now enable ATMs to not only help customers withdraw money, but proactively monitor kiosks through the images captured from cameras to identify risk to human safety or fraudulent behavior. In this scenario, edge compute servers and shared storage systems are connected to interactive kiosks and cameras to help banks collect and process data with AI inference models.

**Manufacturing: Industry 4.0**

The fourth industrial revolution (Industry 4.0) has begun, along with emerging trends such as Smart Factory and 3D printing. To prepare for a data-led future, large-scale machine-to-machine (M2M) communication and IoT are integrated for increased automation without the need for human intervention. Manufacturing is already highly automated and adding AI features is a natural continuation of the long-term trend. AI enables automating operations that can be automated with the help of computer vision and other AI capabilities. You can automate quality control or tasks that rely on human vision or decision making to perform faster analyses of materials on assembly lines in factory floors to help manufacturing plants meet the required ISO standards of safety and quality management. Here, each compute edge server is connected to an array of sensors monitoring the manufacturing process and updated inference models are pushed to the shared storage, as needed.
Telecommunications: Rust detection, tower inspection, and network optimization

The telecommunications industry uses computer vision and AI techniques to process images that automatically detect rust and identify cell towers that contain corrosion and, therefore, require further inspection. The use of drone images and AI models to identify distinct regions of a tower to analyze rust, surface cracks, and corrosion has increased in recent years. The demand continues to grow for AI technologies that enable telecommunication infrastructure and cell towers to be inspected efficiently, assessed regularly for degradation, and repaired promptly when required.

Additionally, another emerging use case in telecommunication is the use of AI and ML algorithms to predict data traffic patterns, detect 5G-capable devices, and automate and augment multiple-input and multiple-output (MIMO) energy management. MIMO hardware is used at radio towers to increase network capacity; however, this comes with additional energy costs. ML models for “MIMO sleep mode” deployed at cell sites can predict the efficient use of radios and help reduce energy consumption costs for mobile network operators (MNOs). AI inferencing and edge computing solutions help MNOs reduce the amount of data transmitted back-and-forth to data centers, lower their TCO, optimize network operations, and improve overall performance for end users.

Technology overview

This section describes the technological foundation for this AI solution.

NetApp AFF systems

State-of-the-art NetApp AFF storage systems enable AI inference deployments at the edge to meet enterprise storage requirements with industry-leading performance, superior flexibility, cloud integration, and best-in class data management. Designed specifically for flash, NetApp AFF systems help accelerate, manage, and protect business-critical data.

- Entry-level NetApp AFF storage systems are based on FAS2750 hardware and SSD flash media
- Two controllers in HA configuration

NetApp entry-level AFF C190 storage systems support the following features:

- A maximum drive count of 24x 960GB SSDs
- Two possible configurations:
• Ethernet (10GbE): 4x 10GBASE-T (RJ-45) ports
• Unified (16Gb FC or 10GbE): 4x unified target adapter 2 (UTA2) ports
• A maximum of 50.5TB effective capacity

For NAS workloads, a single entry-level AFF C190 system supports throughput of 4.4GBps for sequential reads and 230K IOPS for small random reads at latencies of 1ms or less.

NetApp AFF A220

NetApp also offers other entry-level storage systems that provide higher performance and scalability for larger-scale deployments. For NAS workloads, a single entry-level AFF A220 system supports:

• Throughput of 6.2GBps for sequential reads
• 375K IOPS for small random reads at latencies of 1ms or less
• Maximum drive count of 144x 960GB, 3.8TB, or 7.6TB SSDs
• AFF A220 scales to larger than 1PB of effective capacity

NetApp AFF A250

• Maximum effective capacity is 35PB with maximum scale out 2-24 nodes (12 HA pairs)
• Provides ≥ 45% performance increase over AFF A220
• 440k IOPS random reads @1ms
• Built on the latest NetApp ONTAP release: ONTAP 9.8
• Leverages two 25Gb Ethernet for HA and cluster interconnect

NetApp E-Series EF Systems

The EF-Series is a family of entry-level and mid-range all-flash SAN storage arrays that can accelerate access to your data and help you derive value from it faster with NetApp SANtricity software. These systems offer both SAS and NVMe flash storage and provide you with affordable to extreme IOPS, response times under 100 microseconds, and bandwidth up to 44GBps—making them ideal for mixed workloads and demanding applications such as AI inferencing and high-performance computing (HPC).

The following figure shows the NetApp EF280 storage system.
NetApp EF280

- 32Gb/16Gb FC, 25Gb/10Gb iSCSI, and 12Gb SAS support
- Maximum effective capacity is 96 drives totaling 1.5PB
- Throughput of 10GBps (sequential reads)
- 300K IOPs (random reads)
- The NetApp EF280 is the lowest cost all-flash array (AFA) in the NetApp portfolio

NetApp EF300

- 24x NVMe SSD drives for a total capacity of 367TB
- Expansion options totaling 240x NL-SAS HDDs, 96x SAS SSDs, or a combination
- 100Gb NVMe/IB, NVMe/RoCE, iSER/IB, and SRP/IB
- 32Gb NVME/FC, FCP
- 25Gb iSCSI
- 20GBps (sequential reads)
- 670K IOPs (random reads)

For more information, see the NetApp EF-Series NetApp EF-Series all-flash arrays EF600, F300, EF570, and EF280 datasheet.

NetApp ONTAP 9

ONTAP 9.8.1, the latest generation of storage management software from NetApp, enables businesses to modernize infrastructure and transition to a cloud-ready data center. Leveraging industry-leading data management capabilities, ONTAP enables the management and protection of data with a single set of tools, regardless of where that data resides. You can also move data freely to wherever it is needed: the edge, the core, or the cloud. ONTAP 9.8.1 includes numerous features that simplify data management, accelerate and protect critical data, and enable next generation infrastructure capabilities across hybrid cloud architectures.

Simplify data management

Data management is crucial to enterprise IT operations so that appropriate resources are used for applications and datasets. ONTAP includes the following features to streamline and simplify operations and reduce the total cost of operation:

- **Inline data compaction and expanded deduplication.** Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity. This applies to data stored locally and data tiered to the cloud.

- **Minimum, maximum, and adaptive quality of service (AQoS).** Granular quality of service (QoS) controls help maintain performance levels for critical applications in highly shared environments.

- **NetApp FabricPool.** This feature provides automatic tiering of cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID storage solution. For more information about FabricPool, see TR-4598.

Accelerate and protect data

ONTAP 9 delivers superior levels of performance and data protection and extends these capabilities in the following ways:
• **Performance and lower latency.** ONTAP offers the highest possible throughput at the lowest possible latency.

• **Data protection.** ONTAP provides built-in data protection capabilities with common management across all platforms.

• **NetApp Volume Encryption (NVE).** ONTAP offers native volume-level encryption with both onboard and External Key Management support.

• **Multitenancy and multifactor authentication.** ONTAP enables sharing of infrastructure resources with the highest levels of security.

**Future-proof infrastructure**

ONTAP 9 helps meet demanding and constantly changing business needs with the following features:

• **Seamless scaling and nondisruptive operations.** ONTAP supports the nondisruptive addition of capacity to existing controllers and to scale-out clusters. Customers can upgrade to the latest technologies, such as NVMe and 32Gb FC, without costly data migrations or outages.

• **Cloud connection.** ONTAP is the most cloud-connected storage management software, with options for software-defined storage (ONTAP Select) and cloud-native instances (NetApp Cloud Volumes Service) in all public clouds.

• **Integration with emerging applications.** ONTAP offers enterprise-grade data services for next generation platforms and applications, such as autonomous vehicles, smart cities, and Industry 4.0, by using the same infrastructure that supports existing enterprise apps.

**NetApp SANtricity**

NetApp SANtricity is designed to deliver industry-leading performance, reliability, and simplicity to E-Series hybrid-flash and EF-Series all-flash arrays. Achieve maximum performance and utilization of your E-Series hybrid-flash and EF-Series all-flash arrays for heavy-workload applications, including data analytics, video surveillance, and backup and recovery. With SANtricity, configuration tweaking, maintenance, capacity expansion, and other tasks can be completed while the storage stays online. SANtricity also provides superior data protection, proactive monitoring, and certified security—all accessible through the easy-to-use, on-box System Manager interface. To learn more, see the NetApp E-Series SANtricity Software datasheet.

**Performance optimized**

Performance-optimized SANtricity software delivers data—with high IOPs, high throughput, and low latency—to all your data analytics, video surveillance, and backup apps. Accelerate performance for high-IOPS, low-latency applications and high-bandwidth, high-throughput applications.

**Maximize uptime**

Complete all your management tasks while the storage stays online. Tweak configurations, perform maintenance, or expand capacity without disrupting I/O. Realize best-in-class reliability with automated features, online configuration, state-of-the-art Dynamic Disk Pools (DPP) technology, and more.

**Rest easy**

SANtricity software delivers superior data protection, proactive monitoring, and certified security—all through the easy-to-use, on-box System Manager interface. Simplify storage-management chores. Gain the flexibility you need for advanced tuning of all E-Series storage systems. Manage your NetApp E-Series system—anytime, anywhere. Our on-box, web-based interface streamlines your management workflow.
NetApp Trident

Trident from NetApp is an open-source dynamic storage orchestrator for Docker and Kubernetes that simplifies the creation, management, and consumption of persistent storage. Trident, a Kubernetes native application, runs directly within a Kubernetes cluster. Trident enables customers to seamlessly deploy DL container images onto NetApp storage and provides an enterprise-grade experience for AI container deployments. Kubernetes users (such as ML developers and data scientists) can create, manage, and automate orchestration and cloning to take advantage of NetApp advanced data management capabilities powered by NetApp technology.

NetApp BlueXP Copy and Sync

BlueXP Copy and Sync is a NetApp service for rapid and secure data synchronization. Whether you need to transfer files between on-premises NFS or SMB file shares, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, Amazon Simple Storage Service (Amazon S3), Amazon Elastic File System (Amazon EFS), Azure Blob, Google Cloud Storage, or IBM Cloud Object Storage, BlueXP Copy and Sync moves the files where you need them quickly and securely. After your data is transferred, it is fully available for use on both source and target. BlueXP Copy and Sync continuously synchronizes the data, based on your predefined schedule, moving only the deltas, so time and money spent on data replication is minimized. BlueXP Copy and Sync is a software as a service (SaaS) tool that is extremely simple to set up and use. Data transfers that are triggered by BlueXP Copy and Sync are carried out by data brokers. You can deploy BlueXP Copy and Sync data brokers in AWS, Azure, Google Cloud Platform, or on-premises.

Lenovo ThinkSystem servers

Lenovo ThinkSystem servers feature innovative hardware, software, and services that solve customers’ challenges today and deliver an evolutionary, fit-for-purpose, modular design approach to address tomorrow’s challenges. These servers capitalize on best-in-class, industry-standard technologies coupled with differentiated Lenovo innovations to provide the greatest possible flexibility in x86 servers.

Key advantages of deploying Lenovo ThinkSystem servers include:

- Highly scalable, modular designs to grow with your business
- Industry-leading resilience to save hours of costly unscheduled downtime
- Fast flash technologies for lower latencies, quicker response times, and smarter data management in real time

In the AI area, Lenovo is taking a practical approach to helping enterprises understand and adopt the benefits of ML and AI for their workloads. Lenovo customers can explore and evaluate Lenovo AI offerings in Lenovo AI Innovation Centers to fully understand the value for their particular use case. To improve time to value, this customer-centric approach gives customers proof of concept for solution development platforms that are ready to use and optimized for AI.

Lenovo ThinkSystem SE350 Edge Server

Edge computing allows data from IoT devices to be analyzed at the edge of the network before being sent to the data center or cloud. The Lenovo ThinkSystem SE350, as shown in the figure below, is designed for the unique requirements for deployment at the edge, with a focus on flexibility, connectivity, security, and remote manageability in a compact ruggedized and environmentally hardened form factor.

Featuring the Intel Xeon D processor with the flexibility to support acceleration for edge AI workloads, the SE350 is purpose-built for addressing the challenge of server deployments in a variety of environments outside the data center.
MLPerf

MLPerf is the industry-leading benchmark suite for evaluating AI performance. It covers many areas of applied AI including image classification, object detection, medical imaging, and natural language processing (NLP). In this validation, we used Inference v0.7 workloads, which is the latest iteration of the MLPerf Inference at the completion of this validation. The MLPerf Inference v0.7 suite includes four new benchmarks for data center and edge systems:

- **BERT.** Bi-directional Encoder Representation from Transformers (BERT) fine-tuned for question answering by using the SQuAD dataset.
- **DLRM.** Deep Learning Recommendation Model (DLRM) is a personalization and recommendation model that is trained to optimize click-through rates (CTR).
- **3D U-Net.** 3D U-Net architecture is trained on the Brain Tumor Segmentation (BraTS) dataset.
- **RNN-T.** Recurrent Neural Network Transducer (RNN-T) is an automatic speech recognition (ASR) model.
that is trained on a subset of LibriSpeech. MLPerf Inference results and code are publicly available and released under Apache license. MLPerf Inference has an Edge division, which supports the following scenarios:

- **Single stream.** This scenario mimics systems where responsiveness is a critical factor, such as offline AI queries performed on smartphones. Individual queries are sent to the system and response times are recorded. 90th percentile latency of all the responses is reported as the result.

- **Multistream.** This benchmark is for systems that process input from multiple sensors. During the test, queries are sent at a fixed time interval. A QoS constraint (maximum allowed latency) is imposed. The test reports the number of streams that the system can process while meeting the QoS constraint.

- **Offline.** This is the simplest scenario covering batch processing applications and the metric is throughput in samples per second. All data is available to the system and the benchmark measures the time it takes to process all the samples.

Lenovo has published MLPerf Inference scores for SE350 with T4, the server used in this document. See the results at [https://mlperf.org/inference-results-0-7/](https://mlperf.org/inference-results-0-7/) in the “Edge, Closed Division” section in entry #0.7-145.

**Test plan**

This document follows MLPerf Inference v0.7 code, MLPerf Inference v1.1 code, and rules. We ran MLPerf benchmarks designed for inference at the edge as defined in the follow table.

<table>
<thead>
<tr>
<th>Area</th>
<th>Task</th>
<th>Model</th>
<th>Dataset</th>
<th>QSL size</th>
<th>Quality</th>
<th>Multistream latency constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision</td>
<td>Image classification</td>
<td>Resnet50v1.5</td>
<td>ImageNet (224x224)</td>
<td>1024</td>
<td>99% of FP32</td>
<td>50ms</td>
</tr>
<tr>
<td>Vision</td>
<td>Object detection (large)</td>
<td>SSD-ResNet34</td>
<td>COCO (1200x1200)</td>
<td>64</td>
<td>99% of FP32</td>
<td>66ms</td>
</tr>
<tr>
<td>Vision</td>
<td>Object detection (small)</td>
<td>SSD-MobileNetsv1</td>
<td>COCO (300x300)</td>
<td>256</td>
<td>99% of FP32</td>
<td>50ms</td>
</tr>
<tr>
<td>Vision</td>
<td>Medical image segmentation</td>
<td>3D UNET</td>
<td>BraTS 2019 (224x224x160)</td>
<td>16</td>
<td>99% and 99.9% of FP32</td>
<td>n/a</td>
</tr>
<tr>
<td>Speech</td>
<td>Speech-to-text</td>
<td>RNNT</td>
<td>Librispeech dev-clean</td>
<td>2513</td>
<td>99% of FP32</td>
<td>n/a</td>
</tr>
<tr>
<td>Language</td>
<td>Language processing</td>
<td>BERT</td>
<td>SQuAD v1.1</td>
<td>10833</td>
<td>99% of FP32</td>
<td>n/a</td>
</tr>
</tbody>
</table>

The following table presents Edge benchmark scenarios.

<table>
<thead>
<tr>
<th>Area</th>
<th>Task</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision</td>
<td>Image classification</td>
<td>Single stream, offline, multistream</td>
</tr>
<tr>
<td>Vision</td>
<td>Object detection (large)</td>
<td>Single stream, offline, multistream</td>
</tr>
</tbody>
</table>
### Area | Task | Scenarios
--- | --- | ---
Vision | Object detection (small) | Single stream, offline, multistream
Vision | Medical image segmentation | Single stream, offline
Speech | Speech-to-text | Single stream, offline
Language | Language processing | Single stream, offline

We performed these benchmarks using the networked storage architecture developed in this validation and compared results to those from local runs on the edge servers previously submitted to MLPerf. The comparison is to determine how much impact the shared storage has on inference performance.

#### Test configuration

The following figure shows the test configuration. We used the NetApp AFF C190 storage system and two Lenovo ThinkSystem SE350 servers (each with one NVIDIA T4 accelerator). These components are connected through a 10GbE network switch. The network storage holds validation/test datasets and pretrained models. The servers provide computational capability, and the storage is accessed over NFS protocol.

This section describes the tested configurations, the network infrastructure, the SE350 server, and the storage provisioning details. The following table lists the base components for the solution architecture.

<table>
<thead>
<tr>
<th>Solution components</th>
<th>Details</th>
</tr>
</thead>
</table>
| Lenovo ThinkSystem servers | • 2x SE350 servers each with one NVIDIA T4 GPU card  
• Each server contains one Intel Xeon D-2123IT CPU with four physical cores running at 2.20GHz and 128GB RAM |
| Entry-level NetApp AFF storage system (HA pair) | • NetApp ONTAP 9 software  
• 24x 960GB SSDs  
• NFS protocol  
• One interface group per controller, with four logical IP addresses for mount points |
The following table lists the storage configuration: AFF C190 with 2RU, 24 drive slots.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Aggregate</th>
<th>FlexGroup volume</th>
<th>Aggregatesize</th>
<th>Volumesize</th>
<th>Operating systemmount point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller1</td>
<td>Aggr1</td>
<td>/netapplenovo_AI_fg</td>
<td>8.42TiB</td>
<td>15TB</td>
<td>/netapp_lenovo_fg</td>
</tr>
<tr>
<td>Controller2</td>
<td>Aggr2</td>
<td></td>
<td>8.42TiB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The /netappLenovo_AI_fg folder contains the datasets used for model validation.

The figure below shows the test configuration. We used the NetApp EF280 storage system and two Lenovo ThinkSystem SE350 servers (each with one NVIDIA T4 accelerator). These components are connected through a 10GbE network switch. The network storage holds validation/test datasets and pretrained models. The servers provide computational capability, and the storage is accessed over NFS protocol.

The following table lists the storage configuration for EF280.
<table>
<thead>
<tr>
<th>Controller</th>
<th>Volume Group</th>
<th>Volume</th>
<th>Volumesize</th>
<th>DDPsize</th>
<th>Connection method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller1</td>
<td>DDP1</td>
<td>Volume 1</td>
<td>8.42TiB</td>
<td>16TB</td>
<td>SE350-1 to iSCSI LUN 0</td>
</tr>
<tr>
<td>Controller2</td>
<td></td>
<td>Volume 2</td>
<td>8.42TiB</td>
<td></td>
<td>SE350-2 to iSCSI LUN 1</td>
</tr>
</tbody>
</table>

**Test procedure**

This section describes the test procedures used to validate this solution.

**Operating system and AI inference setup**

For AFF C190, we used Ubuntu 18.04 with NVIDIA drivers and docker with support for NVIDIA GPUs and used MLPerf code available as a part of the Lenovo submission to MLPerf Inference v0.7.

For EF280, we used Ubuntu 20.04 with NVIDIA drivers and docker with support for NVIDIA GPUs and MLPerf code available as a part of the Lenovo submission to MLPerf Inference v1.1.

To set up the AI inference, follow these steps:

1. Download datasets that require registration, the ImageNet 2012 Validation set, Criteo Terabyte dataset, and BraTS 2019 Training set, and then unzip the files.

2. Create a working directory with at least 1TB and define environmental variable MLPERF_SCRATCH_PATH referring to the directory.

   You should share this directory on the shared storage for the network storage use case, or the local disk when testing with local data.
3. Run the make `prebuild` command, which builds and launches the docker container for the required inference tasks.

The following commands are all executed from within the running docker container:

- Download pretrained AI models for MLPerf Inference tasks: `make download_model`
- Download additional datasets that are freely downloadable: `make download_data`
- Preprocess the data: `make preprocess_data`
- Run: `make build`.
- Build inference engines optimized for the GPU in compute servers: `make generate_engines`
- To run Inference workloads, run the following (one command):

```
make run_harness RUN_ARGS="--benchmarks=<BENCHMARKS> --scenarios=<SCENARIOS>"
```

**AI inference runs**

Three types of runs were executed:

- Single server AI inference using local storage
- Single server AI inference using network storage
- Multi-server AI inference using network storage

**Test results**

A multitude of tests were run to evaluate the performance of the proposed architecture.

There are six different workloads (image classification, object detection [small], object detection [large], medical imaging, speech-to-text, and natural language processing [NLP]), which you can run in three different scenarios: offline, single stream, and multistream.

The last scenario is implemented only for image classification and object detection.

This gives 15 possible workloads, which were all tested under three different setups:

- Single server/local storage
- Single server/network storage
- Multi-server/network storage

The results are described in the following sections.

**AI inference in offline scenario for AFF**

In this scenario, all the data was available to the server and the time it took to process all the samples was measured. We report bandwidths in samples per second as the results of the tests. When more than one compute server was used, we report total bandwidth summed over all the servers. The results for all three use
cases are shown in the figure below. For the two-server case, we report combined bandwidth from both servers.

The results show that network storage does not negatively affect the performance—the change is minimal and for some tasks, none is found. When adding the second server, the total bandwidth either exactly doubles, or at worst, the change is less than 1%.

**AI inference in a single stream scenario for AFF**

This benchmark measures latency. For the multiple computational server case, we report the average latency. The results for the suite of tasks are given in the figure below. For the two-server case, we report the average latency from both servers.
The results, again, show that the network storage is sufficient to handle the tasks. The difference between local and network storage in the one server case is minimal or none. Similarly, when two servers use the same storage, the latency on both servers stays the same or changes by a very small amount.

**AI inference in multistream scenario for AFF**

In this case, the result is the number of streams that the system can handle while satisfying the QoS constraint. Thus, the result is always an integer. For more than one server, we report the total number of streams summed over all the servers. Not all workloads support this scenario, but we have executed those that do. The results of our tests are summarized in the figure below. For the two-server case, we report the combined number of streams from both servers.
The results show perfect performance of the setup—local and networking storage give the same results and adding the second server doubles the number of streams the proposed setup can handle.

**Test results for EF**

A multitude of tests were run to evaluate the performance of the proposed architecture. There are six different workloads (image classification, object detection [small], object detection [large], medical imaging, speech-to-text, and natural language processing [NLP]), which were run in two different scenarios: offline and single stream. The results are described in the following sections.

**AI inference in offline scenario for EF**

In this scenario, all the data was available to the server and the time it took to process all the samples was measured. We report bandwidths in samples per second as the results of the tests. For single node runs we report average from both servers, while for two server runs we report total bandwidth summed over all the servers. The results for use cases are shown in the figure below.
The results show that network storage does not negatively affect the performance—the change is minimal and for some tasks, none is found. When adding the second server, the total bandwidth either exactly doubles, or at worst, the change is less than 1%.

**AI inference in a single stream scenario for EF**

This benchmark measures latency. For all cases, we report average latency across all servers involved in the runs. The results for the suite of tasks are given.
The results show again that the network storage is sufficient to handle the tasks. The difference between the local and network storage in the one server case is minimal or none. Similarly, when two servers use the same storage, the latency on both servers stays the same or changes by a very small amount.

**Architecture sizing options**

You can adjust the setup used for the validation to fit other use cases.

**Compute server**

We used an Intel Xeon D-2123IT CPU, which is the lowest level of CPU supported in SE350, with four physical cores and 60W TDP. While the server does not support replacing CPUs, it can be ordered with a more powerful CPU. The top CPU supported is Intel Xeon D-2183IT with 16 cores, 100W running at 2.20GHz. This increases the CPU computational capability considerably. While CPU was not a bottleneck for running the inference workloads themselves, it helps with data processing and other tasks related to inference. At present, NVIDIA T4 is the only GPU available for edge use cases; therefore, currently, there is no ability to upgrade or downgrade the GPU.

**Shared storage**

For testing and validation, the NetApp AFF C190 system, which has maximum storage capacity of 50.5TB, a throughput of 4.4GBps for sequential reads, and 230K IOPS for small random reads, was used for the purpose of this document and is proven to be well-suited for edge inference workloads.

However, if you require more storage capacity or faster networking speeds, you should use the NetApp AFF A220 or NetApp AFF A250 storage systems. In addition, the NetApp EF280 system, which has a maximum capacity of 1.5PB, bandwidth 10GBps was also used for the purpose of this solution validation. If you prefer more storage capacity with higher bandwidth, NetApp EF300 can be used.
Conclusion

AI-driven automation and edge computing is a leading approach to help business organizations achieve digital transformation and maximize operational efficiency and safety. With edge computing, data is processed much faster because it does not have to travel to and from a data center. Therefore, the cost associated with sending data back and forth to data centers or the cloud is diminished. Lower latency and increased speed can be beneficial when businesses must make decisions in near-real time using AI inferencing models deployed at the edge.

NetApp storage systems deliver the same or better performance as local SSD storage and offer the following benefits to data scientists, data engineers, AI/ML developers, and business or IT decision makers:

• Effortless sharing of data between AI systems, analytics, and other critical business systems. This data sharing reduces infrastructure overhead, improves performance, and streamlines data management across the enterprise.
• Independently scalable compute and storage to minimize costs and improve resource usage.
• Streamlined development and deployment workflows using integrated Snapshot copies and clones for instantaneous and space-efficient user workspaces, integrated version control, and automated deployment.
• Enterprise-grade data protection for disaster recovery and business continuity. The NetApp and Lenovo solution presented in this document is a flexible, scale-out architecture that is ideal for enterprise-grade AI inference deployments at the edge.

Acknowledgments

• J.J. Falkanger, Sr. Manager, HPC & AI Solutions, Lenovo
• Dave Arnette, Technical Marketing Engineer, NetApp
• Joey Parnell, Tech Lead E-Series AI Solutions, NetApp
• Cody Harryman, QA Engineer, NetApp

Where to find additional information

To learn more about the information described in this document, refer to the following documents and/or websites:

• NetApp AFF A-Series arrays product page
  https://www.netapp.com/data-storage/aff-a-series/
• NetApp ONTAP data management software—ONTAP 9 information library
• TR-4727: NetApp EF-Series Introduction
• NetApp E-Series SANtricity Software Datasheet
WP-7328: NetApp Conversational AI Using NVIDIA Jarvis

Rick Huang, Sung-Han Lin, NetApp
Davide Onofrio, NVIDIA

The NVIDIA DGX family of systems is made up of the world’s first integrated artificial intelligence (AI)-based systems that are purpose-built for enterprise AI. NetApp AFF storage systems deliver extreme performance and industry-leading hybrid cloud data-management capabilities. NetApp and NVIDIA have partnered to create the NetApp ONTAP AI reference architecture, a turnkey solution for AI and machine learning (ML) workloads that provides enterprise-class performance, reliability, and support.

This white paper gives directional guidance to customers building conversational AI systems in support of different use cases in various industry verticals. It includes information about the deployment of the system using NVIDIA Jarvis. The tests were performed using an NVIDIA DGX Station and a NetApp AFF A220 storage system.

The target audience for the solution includes the following groups:

- Enterprise architects who design solutions for the development of AI models and software for conversational AI use cases such as a virtual retail assistant
- Data scientists looking for efficient ways to achieve language modeling development goals
- Data engineers in charge of maintaining and processing text data such as customer questions and dialogue transcripts
- Executive and IT decision makers and business leaders interested in transforming the conversational AI experience and achieving the fastest time to market from AI initiatives
Solution Overview

NetApp ONTAP AI and BlueXP Copy and Sync

The NetApp ONTAP AI architecture, powered by NVIDIA DGX systems and NetApp cloud-connected storage systems, was developed and verified by NetApp and NVIDIA. This reference architecture gives IT organizations the following advantages:

- Eliminates design complexities
- Enables independent scaling of compute and storage
- Enables customers to start small and scale seamlessly
- Offers a range of storage options for various performance and cost points

NetApp ONTAP AI tightly integrates DGX systems and NetApp AFF A220 storage systems with state-of-the-art networking. NetApp ONTAP AI and DGX systems simplify AI deployments by eliminating design complexity and guesswork. Customers can start small and grow their systems in an uninterrupted manner while intelligently managing data from the edge to the core to the cloud and back.

NetApp BlueXP Copy and Sync enables you to move data easily over various protocols, whether it’s between two NFS shares, two CIFS shares, or one file share and Amazon S3, Amazon Elastic File System (EFS), or Azure Blob storage. Active-active operation means that you can continue to work with both source and target at the same time, incrementally synchronizing data changes when required. By enabling you to move and incrementally synchronize data between any source and destination system, whether on-premises or cloud-based, BlueXP Copy and Sync opens up a wide variety of new ways in which you can use data. Migrating data between on-premises systems, cloud on-boarding and cloud migration, or collaboration and data analytics all become easily achievable. The figure below shows available sources and destinations.

In conversational AI systems, developers can leverage BlueXP Copy and Sync to archive conversation history from the cloud to data centers to enable offline training of natural language processing (NLP) models. By training models to recognize more intents, the conversational AI system will be better equipped to manage more complex questions from end-users.

NVIDIA Jarvis Multimodal Framework
NVIDIA Jarvis is an end-to-end framework for building conversational AI services. It includes the following GPU-optimized services:

- Automatic speech recognition (ASR)
- Natural language understanding (NLU)
- Integration with domain-specific fulfillment services
- Text-to-speech (TTS)
- Computer vision (CV)

Jarvis-based services use state-of-the-art deep learning models to address the complex and challenging task of real-time conversational AI. To enable real-time, natural interaction with an end user, the models need to complete computation in under 300 milliseconds. Natural interactions are challenging, requiring multimodal sensory integration. Model pipelines are also complex and require coordination across the above services.

Jarvis is a fully accelerated, application framework for building multimodal conversational AI services that use an end-to-end deep learning pipeline. The Jarvis framework includes pretrained conversational AI models, tools, and optimized end-to-end services for speech, vision, and NLU tasks. In addition to AI services, Jarvis enables you to fuse vision, audio, and other sensor inputs simultaneously to deliver capabilities such as multi-user, multi-context conversations in applications such as virtual assistants, multi-user diarization, and call center assistants.

NVIDIA NeMo

NVIDIA NeMo is an open-source Python toolkit for building, training, and fine-tuning GPU-accelerated state-of-the-art conversational AI models using easy-to-use application programming interfaces (APIs). NeMo runs mixed precision compute using Tensor Cores in NVIDIA GPUs and can scale up to multiple GPUs easily to deliver the highest training performance possible. NeMo is used to build models for real-time ASR, NLP, and TTS applications such as video call transcriptions, intelligent video assistants, and automated call center support across different industry verticals, including healthcare, finance, retail, and telecommunications.

We used NeMo to train models that recognize complex intents from user questions in archived conversation history. This training extends the capabilities of the retail virtual assistant beyond what Jarvis supports as delivered.

Retail Use Case Summary

Using NVIDIA Jarvis, we built a virtual retail assistant that accepts speech or text input and answers questions regarding weather, points-of-interest, and inventory pricing. The conversational AI system is able to remember conversation flow, for example, ask a follow-up question if the user does not specify location for weather or points-of-interest. The system also recognizes complex entities such as “Thai food” or “laptop memory.” It understands natural language questions like “will it rain next week in Los Angeles?” A demonstration of the retail virtual assistant can be found in Customize States and Flows for Retail Use Case.

Solution Technology

The following figure illustrates the proposed conversational AI system architecture. You can interact with the system with either speech signal or text input. If spoken input is detected, Jarvis AI-as-service (AIaaS) performs ASR to produce text for Dialog Manager. Dialog Manager remembers states of conversation, routes text to corresponding services, and passes commands to Fulfillment Engine. Jarvis NLP Service takes in text, recognizes intents and entities, and outputs those intents and entity slots back to Dialog Manager, which then sends Action to Fulfillment Engine. Fulfillment Engine consists of third-party
APIs or SQL databases that answer user queries. After receiving Result from Fulfillment Engine, Dialog Manager routes text to Jarvis TTS AIaaS to produce an audio response for the end-user. We can archive conversation history, annotate sentences with intents and slots for NeMo training such that NLP Service improves as more users interact with the system.

Hardware Requirements

This solution was validated using one DGX Station and one AFF A220 storage system. Jarvis requires either a T4 or V100 GPU to perform deep neural network computations.

The following table lists the hardware components that are required to implement the solution as tested.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4 or V100 GPU</td>
<td>1</td>
</tr>
<tr>
<td>NVIDIA DGX Station</td>
<td>1</td>
</tr>
</tbody>
</table>

Software Requirements

The following table lists the software components that are required to implement the solution as tested.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version or Other Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetApp ONTAP data management software</td>
<td>9.6</td>
</tr>
<tr>
<td>Cisco NX-OS switch firmware</td>
<td>7.0(3)I6(1)</td>
</tr>
<tr>
<td>NVIDIA DGX OS</td>
<td>4.0.4 - Ubuntu 18.04 LTS</td>
</tr>
<tr>
<td>NVIDIA Jarvis Framework</td>
<td>EA v0.2</td>
</tr>
</tbody>
</table>
Overview

This section provides detail on the implementation of the virtual retail assistant.

Jarvis Deployment

You can sign up for Jarvis Early Access program to gain access to Jarvis containers on NVIDIA GPU Cloud (NGC). After receiving credentials from NVIDIA, you can deploy Jarvis using the following steps:

1. Sign-on to NGC.
2. Set your organization on NGC: ea-2-jarvis.
3. Locate Jarvis EA v0.2 assets: Jarvis containers are in Private Registry > Organization Containers.
4. Select Jarvis: navigate to Model Scripts and click Jarvis Quick Start
5. Verify that all assets are working properly.
6. Find the documentation to build your own applications: PDFs can be found in Model Scripts > Jarvis Documentation > File Browser.

Customize States and Flows for Retail Use Case

You can customize States and Flows of Dialog Manager for your specific use cases. In our retail example, we have the following four yaml files to direct the conversation according to different intents.

Se the following list of file names and description of each file:

- **main_flow.yml**: Defines the main conversation flows and states and directs the flow to the other three yaml files when necessary.
- **retail_flow.yml**: Contains states related to retail or points-of-interest questions. The system either provides the information of the nearest store, or the price of a given item.
- **weather_flow.yml**: Contains states related to weather questions. If the location cannot be determined, the system asks a follow up question to clarify.
- **error_flow.yml**: Handles cases where user intents do not fall into the above three yaml files. After displaying an error message, the system re-routes back to accepting user questions. The following sections contain the detailed definitions for these yaml files.

**main_flow.yml**

```yaml
name: JarvisRetail
intent_transitions:
```
jarvis_error: error
price_check: retail_price_check
inventory_check: retail_inventory_check
store_location: retail_store_location
weather.weather: weather
weather.temperature: temperature
weather.sunny: sunny
weather.cloudy: cloudy
weather.snow: snow
weather.rainfall: rain
weather.snow_yes_no: snowfall
weather.rainfall_yes_no: rainfall
weather.temperature_yes_no: tempyesno
weather.humidity: humidity
weather.humidity_yes_no: humidity
navigation.startnavigationpoi: retail # Transitions should be context
and slot based. Redirecting for now.
navigation.geteta: retail
navigation.showdirection: retail
navigation.showmappoi: idk_what_you_talkin_about
nomatch.none: idk_what_you_talkin_about
states:
  init:
    type: message_text
    properties:
      text: "Hi, welcome to NARA retail and weather service. How can I help you?"
  input_intent:
    type: input_context
    properties:
      nlp_type: jarvis
      entities:
        intent: dontcare
      # This state is executed if the intent was not understood
  dont_get_the_intent:
    type: message_text_random
    properties:
      responses:
        - "Sorry I didn't get that! Please come again."
        - "I beg your pardon! Say that again?"
        - "Are we talking about weather? What would you like to know?"
        - "Sorry I know only about the weather"
        - "You can ask me about the weather, the rainfall, the temperature, I don't know much more"
    delay: 0
  transitions:
next_state: input_intent
idk_what_you_talkin_about:
  type: message_text_random
  properties:
  responses:
  - "Sorry I didn't get that! Please come again."
  - "I beg your pardon! Say that again?"
  - "Are we talking about retail or weather? What would you like to know?"
  - "Sorry I know only about retail and the weather"
  - "You can ask me about retail information or the weather, the rainfall, the temperature. I don't know much more."
  delay: 0
  transitions:
  next_state: input_intent
error:
  type: change_context
  properties:
  update_keys:
    intent: 'error'
  transitions:
  flow: error_flow
retail_inventory_check:
  type: change_context
  properties:
  update_keys:
    intent: 'retail_inventory_check'
  transitions:
  flow: retail_flow
retail_price_check:
  type: change_context
  properties:
  update_keys:
    intent: 'check_item_price'
  transitions:
  flow: retail_flow
retail_store_location:
  type: change_context
  properties:
  update_keys:
    intent: 'find_the_store'
  transitions:
  flow: retail_flow
weather:
  type: change_context
  properties:
update_keys:
  intent: 'weather'
transitions:
  flow: weather_flow
temperature:
  type: change_context
  properties:
    update_keys:
      intent: 'temperature'
  transitions:
    flow: weather_flow
rainfall:
  type: change_context
  properties:
    update_keys:
      intent: 'rainfall'
  transitions:
    flow: weather_flow
sunny:
  type: change_context
  properties:
    update_keys:
      intent: 'sunny'
  transitions:
    flow: weather_flow
cloudy:
  type: change_context
  properties:
    update_keys:
      intent: 'cloudy'
  transitions:
    flow: weather_flow
snow:
  type: change_context
  properties:
    update_keys:
      intent: 'snow'
  transitions:
    flow: weather_flow
rain:
  type: change_context
  properties:
    update_keys:
      intent: 'rain'
  transitions:
    flow: weather_flow
snowfall:
  type: change_context
  properties:
    update_keys:
      intent: 'snowfall'
  transitions:
    flow: weather_flow

tempyesno:
  type: change_context
  properties:
    update_keys:
      intent: 'tempyesno'
  transitions:
    flow: weather_flow

humidity:
  type: change_context
  properties:
    update_keys:
      intent: 'humidity'
  transitions:
    flow: weather_flow

dend_state:
  type: reset
  transitions:
    next_state: init

retail_flow.yml

name: retail_flow
states:
  store_location:
    type: conditional_exists
    properties:
      key: '{{location}}'
    transitions:
      exists: retail_state
      notexists: ask_retail_location

retail_state:
  type: Retail
  properties:
  transitions:
    next_state: output_retail

output_retail:
  type: message_text
  properties:
For which location? I can find the closest store near you.

I did not understand the location. Can you please repeat?
that?

transitions:
    next_state: input_intent

weather_flow.yml

name: weather_flow
states:
    check_weather_location:
        type: conditional_exists
        properties:
            key: '{{location}}'
        transitions:
            exists: weather_state
            notexists: ask_weather_location
weather_state:
    type: Weather
    properties:
    transitions:
        next_state: output_weather
output_weather:
    type: message_text
    properties:
        text: '{{weather_status}}'
    transitions:
        next_state: input_intent
ask_weather_location:
    type: message_text
    properties:
        text: "For which location?"
    transitions:
        next_state: input_weather_location
input_weather_location:
    type: input_user
    properties:
        nlp_type: jarvis
        entities:
            slot: location
            require_match: true
    transitions:
        match: weather_state
        notmatch: check_jarvis_error
output_weather_acknowledge:
    type: message_text_random
    properties:
responses:
- 'ok in {{location}}'
- 'the weather in {{location}}'
- 'I always wanted to go in {{location}}'

delay: 0
transitions:
  next_state: weather_state
output_weather_notlocation:
  type: message_text
  properties:
    text: "I did not understand the location, can you please repeat?"
  transitions:
    next_state: input_intent
check_jarvis_error:
  type: conditional_exists
  properties:
    key: '{{jarvis_error}}'
  transitions:
    exists: show_jarvis_api_error
    notexists: output_weather_notlocation
show_jarvis_api_error:
  type: message_text
  properties:
    text: "I am having troubled understanding right now. Come again on that, else check jarvis services?"
  transitions:
    next_state: input_intent
Connect to Third-Party APIs as Fulfillment Engine

We connected the following third-party APIs as a Fulfillment Engine to answer questions:

- **WeatherStack API**: returns weather, temperature, rainfall, and snow in a given location.
- **Yelp Fusion API**: returns the nearest store information in a given location.
- **eBay Python SDK**: returns the price of a given item.

**NetApp Retail Assistant Demonstration**

We recorded a demonstration video of NetApp Retail Assistant (NARA).

**Video demonstration of NARA**

Video demonstration of NARA
Use NetApp BlueXP Copy and Sync to Archive Conversation History

By dumping conversation history into a CSV file once a day, we can then leverage BlueXP Copy and Sync to download the log files into local storage. The following figure shows the architecture of having Jarvis deployed on-premises and in public clouds, while using BlueXP Copy and Sync to send conversation history for NeMo training. Details of NeMo training can be found in the section Expand Intent Models Using NeMo Training.
NVIDIA NeMo is a toolkit built by NVIDIA for creating conversational AI applications. This toolkit includes collections of pre-trained modules for ASR, NLP, and TTS, enabling researchers and data scientists to easily compose complex neural network architectures and put more focus on designing their own applications.

As shown in the previous example, NARA can only handle a limited type of question. This is because the pre-trained NLP model only trains on these types of questions. If we want to enable NARA to handle a broader range of questions, we need to retrain it with our own datasets. Thus, here, we demonstrate how we can use NeMo to extend the NLP model to satisfy the requirements. We start by converting the log collected from NARA into the format for NeMo, and then train with the dataset to enhance the NLP model.

**Model**

Our goal is to enable NARA to sort the items based on user preferences. For instance, we might ask NARA to suggest the highest-rated sushi restaurant or might want NARA to look up the jeans with the lowest price. To this end, we use the intent detection and slot filling model provided in NeMo as our training model. This model allows NARA to understand the intent of searching preference.

**Data Preparation**

To train the model, we collect the dataset for this type of question, and convert it to the NeMo format. Here, we listed the files we use to train the model.

**dict.intents.csv**

This file lists all the intents we want the NeMo to understand. Here, we have two primary intents and one intent only used to categorize the questions that do not fit into any of the primary intents.

```
price_check
find_the_store
unknown
```

**dict.slots.csv**

This file lists all the slots we can label on our training questions.

```
B-store.type
B-store.name
B-store.status
B-store.hour.start
B-store.hour.end
B-store.hour.day
B-item.type
B-item.name
B-item.color
B-item.size
B-item.quantity
```
This is the main training dataset. Each line starts with the question following the intent category listing in the file.
dict.intent.csv. The label is enumerated starting from zero.

**train_slots.tsv**

```
20 46 24 25 6 32 6
52 52 24 6
23 52 14 40 52 25 6 32 6
...
```

**Train the Model**

docker pull nvcr.io/nvidia/nemo:v0.10

We then use the following command to launch the container. In this command, we limit the container to use a single GPU (GPU ID = 1) since this is a lightweight training exercise. We also map our local workspace /workspace/nemo/ to the folder inside container /nemo.

```
NV_GPU='1' docker run --runtime=nvidia -it --shm-size=16g \
    --network=host --ulimit memlock=-1 --ulimit \
    stack=67108864 \
    -v /workspace/nemo:/nemo \
    --rm nvcr.io/nvidia/nemo:v0.10
```

Inside the container, if we want to start from the original pre-trained BERT model, we can use the following command to start the training procedure. data_dir is the argument to set up the path of the training data. work_dir allows you to configure where you want to store the checkpoint files.

```
cd examples/nlp/intent_detection_slot_tagging/ 
python joint_intent_slot_with_bert.py \
    --data_dir /nemo/training_data\ 
    --work_dir /nemo/log
```

If we have new training datasets and want to improve the previous model, we can use the following command to continue from the point we stopped. checkpoint_dir takes the path to the previous checkpoints folder.

```
cd examples/nlp/intent_detection_slot_tagging/ 
python joint_intent_slot_infer.py \
    --data_dir /nemo/training_data \
    --checkpoint_dir /nemo/log/2020-05-04_18-34-20/checkpoints/ \
    --eval_file_prefix test
```
### Inference the Model

We need to validate the performance of the trained model after a certain number of epochs. The following command allows us to test the query one-by-one. For instance, in this command, we want to check if our model can properly identify the intention of the query *where can I get the best pasta*.

```bash
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_infer_b1.py \
   --checkpoint_dir /nemo/log/2020-05-29_23-50-58/checkpoints/ \
   --query "where can i get the best pasta" \
   --data_dir /nemo/training_data/ \
   --num_epochs=50
```

Then, the following is the output from the inference. In the output, we can see that our trained model can properly predict the intention *find_the_store*, and return the keywords we are interested in. With these keywords, we enable the NARA to search for what users want and do a more precise search.

```
[NeMo I 2020-05-30 00:06:54 actions:728] Evaluating batch 0 out of 1
[NeMo I 2020-05-30 00:06:55 inference_utils:34] Query: where can i get the best pasta
[NeMo I 2020-05-30 00:06:55 inference_utils:36] Predicted intent: 1 find_the_store
[NeMo I 2020-05-30 00:06:55 inference_utils:50] where B-interrogative.location
[NeMo I 2020-05-30 00:06:55 inference_utils:50] can O
[NeMo I 2020-05-30 00:06:55 inference_utils:50] i O
[NeMo I 2020-05-30 00:06:55 inference_utils:50] get B-verb
[NeMo I 2020-05-30 00:06:55 inference_utils:50] the B-article
[NeMo I 2020-05-30 00:06:55 inference_utils:50] best B-rating.high
[NeMo I 2020-05-30 00:06:55 inference_utils:50] pasta B-item.type
```

### Conclusion

A true conversational AI system engages in human-like dialogue, understands context, and provides intelligent responses. Such AI models are often huge and highly complex. With NVIDIA GPUs and NetApp storage, massive, state-of-the-art language models can be trained and optimized to run inference rapidly. This is a major stride towards ending the trade-off between an AI model that is fast versus one that is large and complex. GPU-optimized language understanding models can be integrated into AI applications for industries such as healthcare, retail, and financial services, powering advanced digital voice assistants in smart speakers and customer service lines. These high-quality conversational AI systems allow businesses across verticals to provide previously unattainable personalized services when engaging with customers.

Jarvis enables the deployment of use cases such as virtual assistants, digital avatars, multimodal sensor fusion (CV fused with ASR/NLP/TTS), or any ASR/NLP/TTS/CV stand-alone use case, such as transcription.
We built a virtual retail assistant that can answer questions regarding weather, points-of-interest, and inventory pricing. We also demonstrated how to improve the natural language understanding capabilities of the conversational AI system by archiving conversation history using BlueXP Copy and Sync and training NeMo models on new data.

Acknowledgments

The authors gratefully acknowledge the contributions that were made to this white paper by our esteemed colleagues from NVIDIA: Davide Onofrio, Alex Qi, Sicong Ji, Marty Jain, and Robert Sohigian. The authors would also like to acknowledge the contributions of key NetApp team members: Santosh Rao, David Arnette, Michael Oglesby, Brent Davis, Andy Sayare, Erik Mulder, and Mike McNamara.

Our 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, see the following resources:

• NVIDIA DGX Station, V100 GPU, GPU Cloud
  ◦ NVIDIA DGX Station
  ◦ NVIDIA V100 Tensor Core GPU
  ◦ NVIDIA NGC
• NVIDIA Jarvis Multimodal Framework
  ◦ NVIDIA Jarvis
    https://developer.nvidia.com/nvidia-jarvis
  ◦ NVIDIA Jarvis Early Access
    https://developer.nvidia.com/nvidia-jarvis-early-access
• NVIDIA NeMo
  ◦ NVIDIA NeMo
    https://developer.nvidia.com/nvidia-nemo
  ◦ Developer Guide
    https://nvidia.github.io/NeMo/
• NetApp AFF systems
  ◦ NetApp AFF A-Series Datasheet
  ◦ NetApp Flash Advantage for All Flash FAS
  ◦ ONTAP 9 Information Library
NetApp AFF storage systems deliver extreme performance and industry-leading hybrid cloud data-management capabilities. NetApp and Run:AI have partnered to demonstrate the unique capabilities of the NetApp ONTAP AI solution for artificial intelligence (AI) and machine learning (ML) workloads that provides enterprise-class performance, reliability, and support. Run:AI orchestration of AI workloads adds a Kubernetes-based scheduling and resource utilization platform to help researchers manage and optimize GPU utilization. Together with the NVIDIA DGX systems, the combined solution from NetApp, NVIDIA, and Run:AI provide an infrastructure stack that is purpose-built for enterprise AI workloads. This technical report gives directional guidance to customers building conversational AI systems in support of various use cases and industry verticals. It includes information about the deployment of Run:AI and a NetApp AFF A800 storage system and serves as a reference architecture for the simplest way to achieve fast, successful deployment of AI initiatives.

The target audience for the solution includes the following groups:

- Enterprise architects who design solutions for the development of AI models and software for Kubernetes-based use cases such as containerized microservices
- Data scientists looking for efficient ways to achieve efficient model development goals in a cluster environment with multiple teams and projects
- Data engineers in charge of maintaining and running production models
- Executive and IT decision makers and business leaders who would like to create the optimal Kubernetes cluster resource utilization experience and achieve the fastest time to market from AI initiatives

Solution Overview

NetApp ONTAP AI and AI Control Plane

The NetApp ONTAP AI architecture, developed and verified by NetApp and NVIDIA, is powered by NVIDIA DGX systems and NetApp cloud-connected storage systems. This reference architecture gives IT
organizations the following advantages:

- Eliminates design complexities
- Enables independent scaling of compute and storage
- Enables customers to start small and scale seamlessly
- Offers a range of storage options for various performance and cost points

NetApp ONTAP AI tightly integrates DGX systems and NetApp AFF A800 storage systems with state-of-the-art networking. NetApp ONTAP AI and DGX systems simplify AI deployments by eliminating design complexity and guesswork. Customers can start small and grow their systems in an uninterrupted manner while intelligently managing data from the edge to the core to the cloud and back.

NetApp AI Control Plane is a full stack AI, ML, and deep learning (DL) data and experiment management solution for data scientists and data engineers. As organizations increase their use of AI, they face many challenges, including workload scalability and data availability. NetApp AI Control Plane addresses these challenges through functionalities, such as rapidly cloning a data namespace just as you would a Git repo, and defining and implementing AI training workflows that incorporate the near-instant creation of data and model baselines for traceability and versioning. With NetApp AI Control Plane, you can seamlessly replicate data across sites and regions and swiftly provision Jupyter Notebook workspaces with access to massive datasets.

Run:AI Platform for AI Workload Orchestration

Run:AI has built the world's first orchestration and virtualization platform for AI infrastructure. By abstracting workloads from the underlying hardware, Run:AI creates a shared pool of GPU resources that can be dynamically provisioned, enabling efficient orchestration of AI workloads and optimized use of GPUs. Data scientists can seamlessly consume massive amounts of GPU power to improve and accelerate their research while IT teams retain centralized, cross-site control and real-time visibility over resource provisioning, queuing, and utilization. The Run:AI platform is built on top of Kubernetes, enabling simple integration with existing IT and data science workflows.

The Run:AI platform provides the following benefits:

- **Faster time to innovation.** By using Run:AI resource pooling, queueing, and prioritization mechanisms together with a NetApp storage system, researchers are removed from infrastructure management hassles and can focus exclusively on data science. Run:AI and NetApp customers increase productivity by running as many workloads as they need without compute or data pipeline bottlenecks.

- **Increased team productivity.** Run:AI fairness algorithms guarantee that all users and teams get their fair share of resources. Policies around priority projects can be preset, and the platform enables dynamic allocation of resources from one user or team to another, helping users to get timely access to coveted GPU resources.

- **Improved GPU utilization.** The Run:AI Scheduler enables users to easily make use of fractional GPUs, integer GPUs, and multiple nodes of GPUs for distributed training on Kubernetes. In this way, AI workloads run based on your needs, not capacity. Data science teams are able to run more AI experiments on the same infrastructure.

Solution Technology

This solution was implemented with one NetApp AFF A800 system, two DGX-1 servers, and two Cisco Nexus 3232C 100GbE-switches. Each DGX-1 server is connected to the Nexus switches with four 100GbE connections that are used for inter-GPU communications by using remote direct memory access (RDMA) over Converged
Ethernet (RoCE). Traditional IP communications for NFS storage access also occur on these links. Each storage controller is connected to the network switches by using four 100GbE-links. The following figure shows the ONTAP AI solution architecture used in this technical report for all testing scenarios.

![ONTAP AI solution architecture](image)

**Hardware Used in This Solution**

This solution was validated using the ONTAP AI reference architecture two DGX-1 nodes and one AFF A800 storage system. See NVA-1121 for more details about the infrastructure used in this validation.

The following table lists the hardware components that are required to implement the solution as tested.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGX-1 systems</td>
<td>2</td>
</tr>
<tr>
<td>AFF A800</td>
<td>1</td>
</tr>
<tr>
<td>Nexus 3232C switches</td>
<td>2</td>
</tr>
</tbody>
</table>

**Software Requirements**

This solution was validated using a basic Kubernetes deployment with the Run:AI operator installed. Kubernetes was deployed using the NVIDIA DeepOps deployment engine, which deploys all required components for a production-ready environment. DeepOps automatically deployed NetApp Trident for persistent storage integration with the k8s environment, and default storage classes were created so containers leverage storage from the AFF A800 storage system. For more information on Trident with Kubernetes on ONTAP AI, see TR-4798.

The following table lists the software components that are required to implement the solution as tested.
### Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Version or Other Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetApp ONTAP data management software</td>
<td>9.6p4</td>
</tr>
<tr>
<td>Cisco NX-OS switch firmware</td>
<td>7.0(3)l6(1)</td>
</tr>
<tr>
<td>NVIDIA DGX OS</td>
<td>4.0.4 - Ubuntu 18.04 LTS</td>
</tr>
<tr>
<td>Kubernetes version</td>
<td>1.17</td>
</tr>
<tr>
<td>Trident version</td>
<td>20.04.0</td>
</tr>
<tr>
<td>Run:AI CLI</td>
<td>v2.1.13</td>
</tr>
<tr>
<td>Run:AI Orchestration Kubernetes Operator version</td>
<td>1.0.39</td>
</tr>
<tr>
<td>Docker container platform</td>
<td>18.06.1-ce [e68fc7a]</td>
</tr>
</tbody>
</table>

### Additional software requirements for Run:AI
Additional software requirements for Run:AI can be found at Run:AI GPU cluster prerequisites.

### Optimal Cluster and GPU Utilization with Run:AI

The following sections provide details on the Run:AI installation, test scenarios, and results performed in this validation.

We validated the operation and performance of this system by using industry standard benchmark tools, including TensorFlow benchmarks. The ImageNet dataset was used to train ResNet-50, which is a famous Convolutional Neural Network (CNN) DL model for image classification. ResNet-50 delivers an accurate training result with a faster processing time, which enabled us to drive a sufficient demand on the storage.

### Run:AI Installation

To install Run:AI, complete the following steps:

1. Install the Kubernetes cluster using DeepOps and configure the NetApp default storage class.
2. Prepare GPU nodes:
   a. Verify that NVIDIA drivers are installed on GPU nodes.
   b. Verify that nvidia-docker is installed and configured as the default docker runtime.
3. Install Run:AI:
   a. Log into the Run:AI Admin UI to create the cluster.
   b. Download the created runai-operator-<clustername>.yaml file.
   c. Apply the operator configuration to the Kubernetes cluster.
      ```
      kubectl apply -f runai-operator-<clustername>.yaml
      ```
4. Verify the installation:
   a. Go to https://app.run.ai/.
   b. Go to the Overview dashboard.
   c. Verify that the number of GPUs on the top right reflects the expected number of GPUs and the GPU nodes are all in the list of servers. For more information about Run:AI deployment, see installing Run:AI...
Run:AI Dashboards and Views

After installing Run:AI on your Kubernetes cluster and configuring the containers correctly, you see the following dashboards and views on https://app.run.ai in your browser, as shown in the following figure.

There are 16 total GPUs in the cluster provided by two DGX-1 nodes. You can see the number of nodes, the total available GPUs, the allocated GPUs that are assigned with workloads, the total number of running jobs, pending jobs, and idle allocated GPUs. On the right side, the bar diagram shows GPUs per Project, which summarizes how different teams are using the cluster resource. In the middle is the list of currently running jobs with job details, including job name, project, user, job type, the node each job is running on, the number of GPU(s) allocated for that job, the current run time of the job, job progress in percentage, and the GPU utilization for that job. Note that the cluster is under-utilized (GPU utilization at 23%) because there are only three running jobs submitted by a single team (team-a).

In the following section, we show how to create multiple teams in the Projects tab and allocate GPUs for each team to maximize cluster usage and manage resources when there are many users per cluster. The test scenarios mimic enterprise environments in which memory and GPU resources are shared among training, inferencing, and interactive workloads.

Creating Projects for Data Science Teams and Allocating GPUs

Researchers can submit workloads through the Run:AI CLI, Kubeflow, or similar processes. To streamline resource allocation and create prioritization, Run:AI introduces the concept of Projects. Projects are quota entities that associate a project name with GPU allocation and preferences. It is a simple and convenient way to manage multiple data science teams.
A researcher submitting a workload must associate a project with a workload request. The Run:AI scheduler compares the request against the current allocations and the project and determines whether the workload can be allocated resources or whether it should remain in a pending state.

As a system administrator, you can set the following parameters in the Run:AI Projects tab:

- **Model projects.** Set a project per user, set a project per team of users, and set a project per a real organizational project.

- **Project quotas.** Each project is associated with a quota of GPUs that can be allocated for this project at the same time. This is a guaranteed quota in the sense that researchers using this project are guaranteed to get this number of GPUs no matter what the status in the cluster is. As a rule, the sum of the project allocation should be equal to the number of GPUs in the cluster. Beyond that, a user of this project can receive an over-quota. As long as GPUs are unused, a researcher using this project can get more GPUs. We demonstrate over-quota testing scenarios and fairness considerations in *Achieving High Cluster Utilization with Over-Quota GPU Allocation, Basic Resource Allocation Fairness, and Over-Quota Fairness.*

- Create a new project, update an existing project, and delete an existing project.

- **Limit jobs to run on specific node groups.** You can assign specific projects to run only on specific nodes. This is useful when the project team needs specialized hardware, for example, with enough memory. Alternatively, a project team might be the owner of specific hardware that was acquired with a specialized budget, or when you might need to direct builds or interactive workloads to work on weaker hardware and direct longer training or unattended workloads to faster nodes. For commands to group nodes and set affinity for a specific project, see the Run:AI Documentation.

- **Limit the duration of interactive jobs.** Researchers frequently forget to close interactive jobs. This might lead to a waste of resources. Some organizations prefer to limit the duration of interactive jobs and close them automatically.

The following figure shows the Projects view with four teams created. Each team is assigned a different number of GPUs to account for different workloads, with the total number of GPUs equal to that of the total available GPUs in a cluster consisting of two DGX-1s.

### Submitting Jobs in Run:AI CLI

This section provides the detail on basic Run:AI commands that you can use to run any Kubernetes job. It is divided into three parts according to workload type. AI/ML/DL workloads can be divided into two generic types:

- **Unattended training sessions.** With these types of workloads, the data scientist prepares a self-running workload and sends it for execution. During the execution, the customer can examine the results. This type
of workload is often used in production or when model development is at a stage where no human intervention is required.

- **Interactive build sessions.** With these types of workloads, the data scientist opens an interactive session with Bash, Jupyter Notebook, remote PyCharm, or similar IDEs and accesses GPU resources directly. We include a third scenario for running interactive workloads with connected ports to reveal an internal port to the container user.

**Unattended Training Workloads**

After setting up projects and allocating GPU(s), you can run any Kubernetes workload using the following command at the command line:

```bash
$ runai project set team-a runai submit hyper1 -i gcr.io/run-ai-demo/quickstart -g 1
```

This command starts an unattended training job for team-a with an allocation of a single GPU. The job is based on a sample Docker image, `gcr.io/run-ai-demo/quickstart`. We named the job `hyper1`. You can then monitor the job’s progress by running the following command:

```bash
$ runai list
```

The following figure shows the result of the `runai list` command. Typical statuses you might see include the following:

- **ContainerCreating.** The docker container is being downloaded from the cloud repository.
- **Pending.** The job is waiting to be scheduled.
- **Running.** The job is running.

To get an additional status on your job, run the following command:

```bash
$ runai get hyper1
```

To view the logs of the job, run the `runai logs <job-name>` command:

```bash
$ runai logs hyper1
```

In this example, you should see the log of a running DL session, including the current training epoch, ETA, loss function value, accuracy, and time elapsed for each step.

You can view the cluster status on the Run:AI UI at [https://app.run.ai/](https://app.run.ai/). Under Dashboards > Overview, you can monitor GPU utilization.
To stop this workload, run the following command:

```
$ runai delte hyper1
```

This command stops the training workload. You can verify this action by running `runai list` again. For more detail, see launching unattended training workloads.

### Interactive Build Workloads

After setting up projects and allocating GPU(s) you can run an interactive build workload using the following command at the command line:

```
$ runai submit build1 -i python -g 1 --interactive --command sleep --args infinity
```

The job is based on a sample docker image python. We named the job `build1`.

The `--interactive` flag means that the job does not have a start or end. It is the researcher’s responsibility to close the job. The administrator can define a time limit for interactive jobs after which they are terminated by the system.

The `--g 1` flag allocates a single GPU to this job. The command and argument provided is `--command sleep --args infinity`. You must provide a command, or the container starts and then exits immediately.

The following commands work similarly to the commands described in Unattended Training Workloads:

- `runai list`: Shows the name, status, age, node, image, project, user, and GPUs for jobs.
- `runai get build1`: Displays additional status on the job `build1`.
- `runai delete build1`: Stops the interactive workload `build1`. To get a bash shell to the container, the following command:

```
$ runai bash build1
```

This provides a direct shell into the computer. Data scientists can then develop or finetune their models within the container.

You can view the cluster status on the Run:AI UI at https://app.run.ai. For more detail, see starting and using interactive build workloads.

### Interactive Workloads with Connected Ports

As an extension of interactive build workloads, you can reveal internal ports to the container user when starting a container with the Run:AI CLI. This is useful for cloud environments, working with Jupyter Notebooks, or connecting to other microservices. Ingress allows access to Kubernetes services from outside the Kubernetes cluster. You can configure access by creating a collection of rules that define which inbound connections reach which services.
For better management of external access to the services in a cluster, we suggest that cluster administrators install Ingress and configure LoadBalancer.

To use Ingress as a service type, run the following command to set the method type and the ports when submitting your workload:

```
$ runai submit test-ingress -i jupyter/base-notebook -g 1 \
   --interactive --service-type=ingress --port 8888 \ 
   --args="--NotebookApp.base_url=test-ingress" --command=start-notebook.sh
```

After the container starts successfully, execute `runai list` to see the SERVICE URL(S) with which to access the Jupyter Notebook. The URL is composed of the ingress endpoint, the job name, and the port. For example, see `https://10.255.174.13/test-ingress-8888`.

For more details, see launching an interactive build workload with connected ports.

**Achieving High Cluster Utilization**

In this section, we emulate a realistic scenario in which four data science teams each submit their own workloads to demonstrate the Run:AI orchestration solution that achieves high cluster utilization while maintaining prioritization and balancing GPU resources. We start by using the ResNet-50 benchmark described in the section ResNet-50 with ImageNet Dataset Benchmark Summary:

```
$ runai submit netapp1 -i netapp/tensorflow-tf1-py3:20.01.0 --local-image 
  --large-shm -v /mnt:/mnt -v /tmp:/tmp --command python --args 
  "/netapp/scripts/run.py" --args "-- 
  dataset_dir=/mnt/mount_0/dataset/imagenet/imagenet_original/" --args "--num_mounts=2"  --args "--dgx_version=dgx1" --args "--num_devices=1" -g 1
```

We ran the same ResNet-50 benchmark as in NVA-1121. We used the flag `--local-image` for containers not residing in the public docker repository. We mounted the directories /mnt and /tmp on the host DGX-1 node to /mnt and /tmp to the container, respectively. The dataset is at NetApp AFFA800 with the `dataset_dir` argument pointing to the directory. Both `--num_devices=1` and `-g 1` mean that we allocate one GPU for this job. The former is an argument for the `run.py` script, while the latter is a flag for the `runai submit` command.

The following figure shows a system overview dashboard with 97% GPU utilization and all sixteen available GPUs allocated. You can easily see how many GPUs are allocated for each team in the GPUs/Project bar chart. The Running Jobs pane shows the current running job names, project, user, type, node, GPUs consumed, run time, progress, and utilization details. A list of workloads in queue with their wait time is shown in Pending Jobs. Finally, the Nodes box offers GPU numbers and utilization for individual DGX-1 nodes in the cluster.
Fractional GPU Allocation for Less Demanding or Interactive Workloads

When researchers and developers are working on their models, whether in the development, hyperparameter tuning, or debugging stages, such workloads usually require fewer computational resources. It is therefore more efficient to provision fractional GPU and memory such that the same GPU can simultaneously be allocated to other workloads. Run:AI’s orchestration solution provides a fractional GPU sharing system for containerized workloads on Kubernetes. The system supports workloads running CUDA programs and is especially suited for lightweight AI tasks such as inference and model building. The fractional GPU system transparently gives data science and AI engineering teams the ability to run multiple workloads simultaneously on a single GPU. This enables companies to run more workloads, such as computer vision, voice recognition, and natural language processing on the same hardware, thus lowering costs.

Run:AI’s fractional GPU system effectively creates virtualized logical GPUs with their own memory and computing space that containers can use and access as if they were self-contained processors. This enables several workloads to run in containers side-by-side on the same GPU without interfering with each other. The solution is transparent, simple, and portable and it requires no changes to the containers themselves.

A typical usecase could see two to eight jobs running on the same GPU, meaning that you could do eight times the work with the same hardware.

For the job frac05 belonging to project team-d in the following figure, we can see that the number of GPUs allocated was 0.50. This is further verified by the nvidia-smi command, which shows that the GPU memory available to the container was 16,255MB: half of the 32GB per V100 GPU in the DGX-1 node.
Achieving High Cluster Utilization with Over-Quota GPU Allocation

In this section and in the sections Basic Resource Allocation Fairness, and Over-Quota Fairness, we have devised advanced testing scenarios to demonstrate the Run:AI orchestration capabilities for complex workload management, automatic preemptive scheduling, and over-quota GPU provisioning. We did this to achieve high cluster-resource usage and optimize enterprise-level data science team productivity in an ONTAP AI environment.

For these three sections, set the following projects and quotas:

<table>
<thead>
<tr>
<th>Project</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-a</td>
<td>4</td>
</tr>
<tr>
<td>team-b</td>
<td>2</td>
</tr>
<tr>
<td>team-c</td>
<td>2</td>
</tr>
<tr>
<td>team-d</td>
<td>8</td>
</tr>
</tbody>
</table>

In addition, we use the following containers for these three sections:

- Jupyter Notebook: jupyter/base-notebook
- Run:AI quickstart: gcr.io/run-ai-demo/quickstart

We set the following goals for this test scenario:
• Show the simplicity of resource provisioning and how resources are abstracted from users
• Show how users can easily provision fractions of a GPU and integer number of GPUs
• Show how the system eliminates compute bottlenecks by allowing teams or users to go over their resource quota if there are free GPUs in the cluster
• Show how data pipeline bottlenecks are eliminated by using the NetApp solution when running compute-intensive jobs, such as the NetApp container
• Show how multiple types of containers are running using the system
  ◦ Jupyter Notebook
  ◦ Run:AI container
• Show high utilization when the cluster is full

For details on the actual command sequence executed during the testing, see Testing Details for Section 4.8.

When all 13 workloads are submitted, you can see a list of container names and GPUs allocated, as shown in the following figure. We have seven training and six interactive jobs, simulating four data science teams, each with their own models running or in development. For interactive jobs, individual developers are using Jupyter Notebooks to write or debug their code. Thus, it is suitable to provision GPU fractions without using too many cluster resources.

The results of this testing scenario show the following:

• The cluster should be full: 16/16 GPUs are used.
• High cluster utilization.
• More experiments than GPUs due to fractional allocation.
  • team-d is not using all their quota; therefore, team-b and team-c can use additional GPUs for their experiments, leading to faster time to innovation.

Basic Resource Allocation Fairness

In this section, we show that, when team-d asks for more GPUs (they are under their quota), the system pauses the workloads of team-b and team-c and moves them into a pending state in a fair-share manner.

For details including job submissions, container images used, and command sequences executed, see the section Testing Details for Section 4.9.

The following figure shows the resulting cluster utilization, GPUs allocated per team, and pending jobs due to automatic load balancing and preemptive scheduling. We can observe that when the total number of GPUs
requested by all team workloads exceeds the total available GPUs in the cluster, Run:AI’s internal fairness algorithm pauses one job each for team-b and team-c because they have met their project quota. This provides overall high cluster utilization while data science teams still work under resource constraints set by an administrator.

The results of this testing scenario demonstrate the following:

- **Automatic load balancing.** The system automatically balances the quota of the GPUs, such that each team is now using their quota. The workloads that were paused belong to teams that were over their quota.

- **Fair share pause.** The system chooses to stop the workload of one team that was over their quota and then stop the workload of the other team. Run:AI has internal fairness algorithms.

**Over-Quota Fairness**

In this section, we expand the scenario in which multiple teams submit workloads and exceed their quota. In this way, we demonstrate how Run:AI’s fairness algorithm allocates cluster resources according to the ratio of preset quotas.

Goals for this test scenario:

- Show queuing mechanism when multiple teams are requesting GPUs over their quota.
- Show how the system distributes a fair share of the cluster between multiple teams that are over their quota according to the ratio between their quotas, so that the team with the larger quota gets a larger share of the spare capacity.

At the end of **Basic Resource Allocation Fairness**, there are two workloads queued: one for team-b and one for team-c. In this section, we queue additional workloads.
For details including job submissions, container images used, and command sequences executed, see Testing Details for section 4.10.

When all jobs are submitted according to the section Testing Details for section 4.10, the system dashboard shows that team-a, team-b, and team-c all have more GPUs than their preset quota. team-a occupies four more GPUs than its preset soft quota (four), whereas team-b and team-c each occupy two more GPUs than their soft quota (two). The ratio of over-quota GPUs allocated is equal to that of their preset quota. This is because the system used the preset quota as a reference of priority and provisioned accordingly when multiple teams request more GPUs, exceeding their quota. Such automatic load balancing provides fairness and prioritization when enterprise data science teams are actively engaged in AI model development and production.

The results of this testing scenario show the following:

- The system starts to de-queue the workloads of other teams.
- The order of the de-queueing is decided according to fairness algorithms, such that team-b and team-c get the same amount of over-quota GPUs (since they have a similar quota), and team-a gets a double amount of GPUs since their quota is two times higher than the quota of team-b and team-c.
- All the allocation is done automatically.

Therefore, the system should stabilize on the following states:

<table>
<thead>
<tr>
<th>Project</th>
<th>GPUs allocated</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-a</td>
<td>8/4</td>
<td>Four GPUs over the quota. Empty queue.</td>
</tr>
</tbody>
</table>

![System Dashboard](image.png)
<table>
<thead>
<tr>
<th>Project</th>
<th>GPUs allocated</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-b</td>
<td>4/2</td>
<td>Two GPUs over the quota. One workload queued.</td>
</tr>
<tr>
<td>team-c</td>
<td>4/2</td>
<td>Two GPUs over the quota. One workload queued.</td>
</tr>
<tr>
<td>team-d</td>
<td>0/8</td>
<td>Not using GPUs at all, no queued workloads.</td>
</tr>
</tbody>
</table>

The following figure shows the GPU allocation per project over time in the Run:AI Analytics dashboard for the sections Achieving High Cluster Utilization with Over-Quota GPU Allocation, Basic Resource Allocation Fairness, and Over-Quota Fairness. Each line in the figure indicates the number of GPUs provisioned for a given data science team at any time. We can see that the system dynamically allocates GPUs according to workloads submitted. This allows teams to go over quota when there are available GPUs in the cluster, and then preempt jobs according to fairness, before finally reaching a stable state for all four teams.

**Saving Data to a Trident-Provisioned PersistentVolume**

NetApp Trident is a fully supported open source project designed to help you meet the sophisticated persistence demands of your containerized applications. You can read and write data to a Trident-provisioned Kubernetes PersistentVolume (PV) with the added benefit of data tiering, encryption, NetApp Snapshot technology, compliance, and high performance offered by NetApp ONTAP data management software.

**Reusing PVCs in an Existing Namespace**

For larger AI projects, it might be more efficient for different containers to read and write data to the same Kubernetes PV. To reuse a Kubernetes Persistent Volume Claim (PVC), the user must have already created a PVC. See the NetApp Trident documentation for details on creating a PVC. Here is an example of reusing an existing PVC:
Run the following command to see the status of job `pvc-test` for project `team-a`:

```
$ runai get pvc-test -p team-a
```

You should see the PV `/tmp/pvc1mount` mounted to `team-a` job `pvc-test`. In this way, multiple containers can read from the same volume, which is useful when there are multiple competing models in development or in production. Data scientists can build an ensemble of models and then combine prediction results by majority voting or other techniques.

Use the following to access the container shell:

```
$ runai bash pvc-test -p team-a
```

You can then check the mounted volume and access your data within the container.

This capability of reusing PVCs works with NetApp FlexVol volumes and NetApp ONTAP FlexGroup volumes, enabling data engineers more flexible and robust data management options to leverage your data fabric powered by NetApp.

**Conclusion**

NetApp and Run:AI have partnered in this technical report to demonstrate the unique capabilities of the NetApp ONTAP AI solution together with the Run:AI Platform for simplifying orchestration of AI workloads. The preceding steps provide a reference architecture to streamline the process of data pipelines and workload orchestration for deep learning. Customers looking to implement these solutions are encouraged to reach out to NetApp and Run:AI for more information.

**Testing Details for Section 4.8**

This section contains the testing details for the section *Achieving High Cluster Utilization with Over-Quota GPU Allocation*.

Submit jobs in the following order:

<table>
<thead>
<tr>
<th>Project</th>
<th>Image</th>
<th># GPUs</th>
<th>Total</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-a</td>
<td>Jupyter</td>
<td>1</td>
<td>1/4</td>
<td>–</td>
</tr>
<tr>
<td>team-a</td>
<td>NetApp</td>
<td>1</td>
<td>2/4</td>
<td>–</td>
</tr>
<tr>
<td>team-a</td>
<td>Run:AI</td>
<td>2</td>
<td>4/4</td>
<td>Using all their quota</td>
</tr>
<tr>
<td>team-b</td>
<td>Run:AI</td>
<td>0.6</td>
<td>0.6/2</td>
<td>Fractional GPU</td>
</tr>
<tr>
<td>Project</td>
<td>Image</td>
<td># GPUs</td>
<td>Total</td>
<td>Comment</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------</td>
<td>-------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>team-b</td>
<td>Run:AI</td>
<td>0.4</td>
<td>1/2</td>
<td>Fractional GPU</td>
</tr>
<tr>
<td>team-b</td>
<td>NetApp</td>
<td>1</td>
<td>2/2</td>
<td></td>
</tr>
<tr>
<td>team-b</td>
<td>NetApp</td>
<td>2</td>
<td>4/2</td>
<td>Two over quota</td>
</tr>
<tr>
<td>team-c</td>
<td>Run:AI</td>
<td>0.5</td>
<td>0.5/2</td>
<td>Fractional GPU</td>
</tr>
<tr>
<td>team-c</td>
<td>Run:AI</td>
<td>0.3</td>
<td>0.8/2</td>
<td>Fractional GPU</td>
</tr>
<tr>
<td>team-c</td>
<td>Run:AI</td>
<td>0.2</td>
<td>1/2</td>
<td>Fractional GPU</td>
</tr>
<tr>
<td>team-c</td>
<td>NetApp</td>
<td>2</td>
<td>3/2</td>
<td>One over quota</td>
</tr>
<tr>
<td>team-c</td>
<td>NetApp</td>
<td>1</td>
<td>4/2</td>
<td>Two over quota</td>
</tr>
<tr>
<td>team-d</td>
<td>NetApp</td>
<td>4</td>
<td>4/8</td>
<td>Using half of their quota</td>
</tr>
</tbody>
</table>

Command structure:

```
$ runai submit <job-name> -p <project-name> -g <#GPUs> -i <image-name>
```

Actual command sequence used in testing:

```
$ runai submit a-1-l-jupyter -i jupyter/base-notebook -g 1 \
   --interactive --service-type=ingress --port 8888 \
   --args="--NotebookApp.base_url=team-a-test-ingress" --command=start \
   -notebook.sh -p team-a
$ runai submit a-2-g -i gcr.io/run-ai-demo/quickstart -g 1 -p team-a
$ runai submit a-2-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-a
$ runai submit b-1-g06 -i gcr.io/run-ai-demo/quickstart -g 0.6 \
   --interactive -p team-b
$ runai submit b-2-g04 -i gcr.io/run-ai-demo/quickstart -g 0.4 \
   --interactive -p team-b
$ runai submit b-3-g -i gcr.io/run-ai-demo/quickstart -g 1 -p team-b
$ runai submit b-4-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-b
$ runai submit c-1-g05 -i gcr.io/run-ai-demo/quickstart -g 0.5 \
   --interactive -p team-c
$ runai submit c-2-g03 -i gcr.io/run-ai-demo/quickstart -g 0.3 \
   --interactive -p team-c
$ runai submit c-3-g02 -i gcr.io/run-ai-demo/quickstart -g 0.2 \
   --interactive -p team-c
$ runai submit c-4-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-c
$ runai submit c-5-g -i gcr.io/run-ai-demo/quickstart -g 1 -p team-c
$ runai submit d-1-gggg -i gcr.io/run-ai-demo/quickstart -g 4 -p team-d
```

At this point, you should have the following states:
### Testing Details for Section 4.9

This section contains testing details for the section **Basic Resource Allocation Fairness**.

Submit jobs in the following order:

<table>
<thead>
<tr>
<th>Project</th>
<th># GPUs</th>
<th>Total</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-d</td>
<td>2</td>
<td>6/8</td>
<td>Team-b/c workload pauses and moves to pending.</td>
</tr>
<tr>
<td>team-d</td>
<td>2</td>
<td>8/8</td>
<td>Other team (b/c) workloads pause and move to pending.</td>
</tr>
</tbody>
</table>

See the following executed command sequence:

```
$ runai submit d-2-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-d
runai submit d-3-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-d
```

At this point, you should have the following states:

<table>
<thead>
<tr>
<th>Project</th>
<th>GPUs Allocated</th>
<th>Workloads Queued</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-a</td>
<td>4/4</td>
<td>None</td>
</tr>
<tr>
<td>team-b</td>
<td>2/2</td>
<td>None</td>
</tr>
<tr>
<td>team-c</td>
<td>2/2</td>
<td>None</td>
</tr>
<tr>
<td>team-d</td>
<td>8/8</td>
<td>None</td>
</tr>
</tbody>
</table>

See the section **Basic Resource Allocation Fairness** for a discussion on the proceeding testing scenario.

### Testing Details for Section 4.10

This section contains testing details for the section **Over-Quota Fairness**.

Submit jobs in the following order for team-a, team-b, and team-c:
See the following executed command sequence:

```
$ runai submit a-3-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-a$
runai submit a-4-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-a$
runai submit b-5-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-b$
runai submit c-6-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-c
```

At this point, you should have the following states:

<table>
<thead>
<tr>
<th>Project</th>
<th>GPUs Allocated</th>
<th>Workloads Queued</th>
</tr>
</thead>
<tbody>
<tr>
<td>team-a</td>
<td>4/4</td>
<td>Two workloads asking for GPUs two each</td>
</tr>
<tr>
<td>team-b</td>
<td>2/2</td>
<td>Two workloads asking for two GPUs each</td>
</tr>
<tr>
<td>team-c</td>
<td>2/2</td>
<td>Two workloads asking for two GPUs each</td>
</tr>
<tr>
<td>team-d</td>
<td>8/8</td>
<td>None</td>
</tr>
</tbody>
</table>

Next, delete all the workloads for team-d:

```
$ runai delete -p team-d d-1-gggg d-2-gg d-3-gg
```

See the section *Over-Quota Fairness*, for discussions on the proceeding testing scenario.

**Where to Find Additional Information**

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

- NVIDIA DGX Systems
  - NVIDIA DGX-1 System
  - NVIDIA V100 Tensor Core GPU
  - NVIDIA NGC
• Run:AI container orchestration solution
  ▪ Run:AI product introduction
    https://docs.run.ai/home/components/
  ▪ Run:AI installation documentation
    https://docs.run.ai/Administrator/Cluster-Setup/Installing-Run-AI-on-an-on-premise-Kubernetes-Cluster/
  ▪ Submitting jobs in Run:AI CLI
    https://docs.run.ai/Researcher/Walkthroughs/Walkthrough-Launch-Unattended-Training-Workloads-/  
                                   https://docs.run.ai/Researcher/Walkthroughs/Walkthrough-Start-and-Use-Interactive-Build-Workloads-/ 
  ▪ Allocating GPU fractions in Run:AI CLI
    https://docs.run.ai/Researcher/Walkthroughs/Walkthrough-Using-GPU-Fractions/

• NetApp AI Control Plane
  ▪ Technical report
  ▪ Short-form demo
    https://youtu.be/gfr_sO27Rvo
  ▪ GitHub repository
    https://github.com/NetApp/kubeflow_jupyter_pipeline

• NetApp AFF systems
  ▪ NetApp AFF A-Series Datasheet
  ▪ NetApp Flash Advantage for All Flash FAS
  ▪ ONTAP 9 Information Library
  ▪ NetApp ONTAP FlexGroup Volumes technical report

• NetApp ONTAP AI
  ▪ ONTAP AI with DGX-1 and Cisco Networking Design Guide
  ▪ ONTAP AI with DGX-1 and Cisco Networking Deployment Guide
  ▪ ONTAP AI with DGX-1 and Mellanox Networking Design Guide
  ▪ ONTAP AI with DGX-2 Design Guide

TR-4799-DESIGN: NetApp ONTAP AI reference architecture for autonomous driving workloads

David Arnette and Sung-Han Lin, NetApp

The NVIDIA DGX family of systems is the world’s first integrated artificial intelligence (AI) platform that is purpose-built for enterprise AI. NetApp AFF storage systems deliver
extreme performance and industry-leading hybrid cloud data-management capabilities. NetApp and NVIDIA have partnered to create the NetApp ONTAP AI reference architecture to offer customers a turnkey solution for supporting AI and machine learning (ML) workloads with enterprise-class performance, reliability, and support.

TR-4799-DESIGN: NetApp ONTAP AI reference architecture for autonomous driving workloads

**TR-4811: NetApp ONTAP AI reference architecture for healthcare: Diagnostic imaging - Solution design**

Rick Huang, Sung-Han Lin, Sathish Thyagarajan, NetApp
Jacci Cenci, NVIDIA

This reference architecture offers guidelines for customers building artificial intelligence (AI) infrastructure using NVIDIA DGX-2 systems and NetApp AFF storage for healthcare use cases. It includes information about the high-level workflows used in the development of deep learning (DL) models for medical diagnostic imaging, validated test cases, and results. It also includes sizing recommendations for customer deployments.

TR-4811: NetApp ONTAP AI reference architecture for healthcare: Diagnostic imaging - Solution design

**TR-4807: NetApp ONTAP AI reference architecture for financial services workloads - Solution design**

Karthikeyan Nagalingam, Sung-Han Lin, NetApp
Jacci Cenci, NVIDIA

This reference architecture offers guidelines for customers who are building artificial intelligence infrastructure using NVIDIA DGX-1 systems and NetApp AFF storage for financial sector use cases. It includes information about the high-level workflows used in the development of deep learning models for financial services test cases and results. It also includes sizing recommendations for customer deployments.

TR-4807: NetApp ONTAP AI reference architecture for financial services workloads - Solution design

**Generative AI and NetApp Value**

Author: Sathish Thyagarajan, NetApp

**Abstract**

The demand for generative artificial intelligence (AI) is driving disruption across industries, enhancing business creativity and product innovation. Many organizations are using generative AI to build new product features, improve engineering productivity and prototype AI powered applications that deliver better results and consumer experiences. Generative AI such as Generative Pre-trained Transformers (GPT) use neural networks to create new content, as diverse as text, audio, and video. Given the extreme scale and massive datasets involved with large language models (LLMs), it is crucial to architect a robust AI infrastructure that takes advantage of the compelling data storage features of on-premises, hybrid and multicloud deployment options and reduce risks associated with data mobility, data protection and governance before companies can design AI solutions. This paper describes these considerations and the corresponding NetApp® AI capabilities.
that enable seamless data management and data movement across the AI data pipeline for training, retraining, fine-tuning, and inferencing generative AI models.

Executive Summary

Most recently after the launch of ChatGPT, a spin-off of GPT-3 in November 2022, new AI tools used to generate text, code, image, or even therapeutic proteins in response to user prompts have gained significant fame. This indicates users can make a request using natural language and AI will interpret and generate text, such as news articles or product descriptions that reflect user request or produce code, music, speech, visual effects, and 3D assets using algorithms trained on already existing data. As a result, phrases like Stable Diffusion, Hallucinations, Prompt Engineering and Value Alignment are rapidly emerging in the design of AI systems. These self-supervised or semi-supervised machine learning (ML) models are becoming widely available as pre-trained foundation models (FM) via cloud service providers and other AI firms/vendors, which are being adopted by various business establishments across industries for a wide range of downstream NLP (natural language processing) tasks. As asserted by research analyst firms like McKinsey – “Generative AI’s impact on productivity could add trillions of dollars in value to the global economy.” While companies are reimagining AI as thought partners to humans and FMs are broadening simultaneously to what businesses and institutions can do with generative AI, the opportunities to manage massive volumes of data will continue to grow. This document presents introductory information on generative AI and the design concepts in relation to NetApp capabilities that bring value to NetApp customers, both on-premises and hybrid or multicloud environments.

So, what’s in it for customers to use NetApp in their AI environments? NetApp helps organizations meet the complexities created by rapid data and cloud growth, multi-cloud management, and the adoption of next-generation technologies, such as AI. NetApp has combined various capabilities into intelligent data management software and storage infrastructure that have been well balanced with high-performance optimized for AI workloads. Generative AI solutions like LLMs need to read and process their source datasets from storage into memory numerous times to foster intelligence. NetApp has been a leader in data mobility, data governance and data security technologies across the edge-to-core-to-cloud ecosystem, serving enterprise customers build at-scale AI solutions. NetApp, with a strong network of partners has been helping chief data officers, AI engineers, enterprise architects and data scientists in the design of a free-flowing data pipeline for data preparation, data protection, and strategic data management responsibilities of AI model training and inferencing, optimizing the performance and scalability of the AI/ML lifecycle. NetApp data technologies and capabilities such as NetApp® ONTAP AI® for deep learning data pipeline, NetApp® SnapMirror® for transporting data seamlessly and efficiently between storage endpoints, and NetApp® FlexCache® for real-time rendering when the data flow shifts from batch to real-time and data engineering happens at prompt time, bring value to the deployment of real-time Generative AI models. As enterprises of all types embrace new AI tools, they face data challenges from the edge to the data center to the cloud that demand for scalable, responsible and explainable AI solutions. As the data authority on hybrid and multi cloud, NetApp is committed to building a network of partners and joint solutions that can help with all aspects of constructing a data pipeline and data lakes for generative AI model training (pre-training), fine-tuning, context-based inferencing and model decay monitoring of LLMs.

What is Generative AI?

Generative AI is changing how we create content, generate new design concepts, and explore novel compositions. It illustrates neural network frameworks like Generative Adversarial Network (GAN), Variational Autoencoders (VAE), and Generative Pre-Trained Transformers (GPT), which can generate new content like text, code, images, audio, video, and synthetic data. Transformer-based models like OpenAI’s Chat-GPT, Google’s Bard, Hugging Face’s BLOOM, and Meta’s LLaMA have emerged as the foundational technology underpinning many advances in large language models. Likewise, OpenAI’s Dall-E, Meta’s CM3eon, and Google’s Imagen are examples for text-to-image diffusion models which offer customers an unprecedented degree of photorealism to create new, complex images from scratch or edit existing images to generate high-quality context-aware images using dataset augmentation and text-to-image synthesis linking textual and visual semantics. Digital artists are starting to apply a combination of rendering technologies like NeRF (Neural
Radiance Field) with generative AI to convert static 2D images into immersive 3D scenes. In general, LLMs are broadly characterized by four parameters: (1) Size of the model (typically in billions of parameters); (2) Size of the training dataset; (3) Cost of training, and (4) Model performance after training. LLMs also fall mainly into three transformer architectures. (i) Encoder-only models. E.g. BERT (Google, 2018); (ii) Encoder-Decoder E.g. BART (Meta, 2020) and (iii) Decoder-only models. E.g. LLaMA (Meta, 2023), PaLM-E (Google, 2023).

Depending on the business requirement, irrespective of which architecture a company chooses the number of model parameters (N) and the number of tokens (D) in the training dataset generally determine the baseline cost of training (pre-training) or fine-tuning an LLM.

**Enterprise Use Cases and Downstream NLP Tasks**

Businesses across industries are uncovering more and more potential for AI to extract and produce new forms of value from existing data for business operations, sales, marketing, and legal services. According to IDC (International Data Corporation) market intelligence on global generative AI use cases and investments, knowledge management in software development and product design is to be the most impacted, followed by storyline creation for marketing and code generation for developers. In healthcare, clinical research organizations are breaking new ground in medicine. Pretrained models like ProteinBERT incorporate Gene Ontology (GO) annotations to rapidly design protein structures for medical drugs, representing a significant milestone in drug discovery, bioinformatics, and molecular biology. Biotech firms have initiated human trials for generative AI-discovered medicine, that aims to treat diseases like pulmonary fibrosis (IPF), a lung disease that causes irreversible scarring of lung tissue.

**Figure 1: Use cases driving Generative AI**

<table>
<thead>
<tr>
<th>Chatbots</th>
<th>Drug discovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text generation</td>
<td>Genome model expression</td>
</tr>
<tr>
<td>Image generation</td>
<td>Classification</td>
</tr>
<tr>
<td>Code generation</td>
<td>Speech-to-Text</td>
</tr>
</tbody>
</table>

Increases in automation adoption driven by generative AI is also changing the supply & demand of work activities for many occupations. As per McKinsey the US labor market (diagram below) has gone through a rapid transition, which may only continue when factoring in the impact of AI.

Source: McKinsey & Company
Role of storage in generative AI

LLMs rely largely on deep learning, GPUs, and compute. However, when GPU buffer fills up, the data needs to be written quickly to storage. While some AI models are small enough to execute in memory, LLMs require high IOPS and high throughput storage to provide fast access to large datasets, especially if it involves billions of tokens or millions of images. For a typical GPU memory requirement of an LLM, the memory needed to train a model with 1 billion parameters could go up to 80GB @32-bit full precision. In which case, Meta’s LLaMA 2, a family of LLMs ranging in scale from 7 billion to 70 billion parameters, may require 70x80, approx. 5600GB or 5.6TB of GPU RAM. Furthermore, the amount of memory you need is directly proportional to the maximum number of tokens you want to generate. For example, if you want to generate outputs of up to 512 tokens (about 380 words), you need "512MB". It may seem inconsequential – but, if you want to run bigger batches it starts to add up. Therefore, making it very expensive for organizations training or fine-tuning LLMs in memory, thus making storage a cornerstone for generative AI.

Three primary approaches to LLMs

For most businesses, based on current trends, the approach to deploying LLMs can be condensed into 3 basic scenarios. As described in a recent “Harvard Business Review” article: (1) Training (pre-training) an LLM from
scratch – costly and requires expert AI/ML skills; (2) Fine-tuning a foundation model with enterprise data – complex, yet feasible; (3) Using retrieval-augmented generation (RAG) to query document repositories, APIs and vector databases that contain company data. Each of these has tradeoffs between the effort, iteration speed, cost-efficiency and model accuracy in their implementations, used to solving different types of problems (diagram below).

Figure 3: Problem Types

Foundation Models

A foundation model (FM) also known as base model is a large AI model (LLM) trained on vast quantities of unlabeled data, using self-supervision at scale, generally adapted for a wide range of downstream NLP tasks. Since the training data is not labelled by humans, the model emerges rather than being explicitly encoded. This means the model can generate stories or a narrative of its own without being explicitly programmed to do so. Hence an important characteristic of FM is homogenization, which means the same method is used in many domains. However, with personalization and fine-tuning techniques, FMs integrated into products appearing these days are not only good at generating text, text-to-images, and text-to-code, but also for explaining domain specific tasks or debugging code. For instance, FMs like OpenAI’s Codex or Meta’s Code Llama can generate code in multiple programming languages based on natural language descriptions of a programming task. These models are proficient in over a dozen programming languages including Python, C#, JavaScript, Perl, Ruby, and SQL. They understand the user’s intent and generate specific code that accomplishes the desired task useful for software development, code optimization, and automation of programming tasks.

Fine-tuning, domain-specificity, and retraining

One of the common practices with LLM deployment following data preparation and data pre-processing is to select a pre-trained model that has been trained on a large and diverse dataset. In the context of fine-tuning this can be an open-source large language model such as "Meta’s Llama 2" trained on 70 billion parameters and 2 trillion tokens. Once the pre-trained model is selected, the next step is to fine-tune it on the domain-specific data. This involves adjusting the model’s parameters and training it on the new data to adapt to a specific domain and task. For example, BloombergGPT, a proprietary LLM trained on a wide range of financial data serving the financial industry. Domain-specific models designed and trained for a specific task generally have higher accuracy and performance within their scope, but low transferability across other tasks or domains. When business environment and data change over a period, the prediction accuracy of the FM could begin to decline when compared to their performance during testing. This is when retraining or fine-tuning the model becomes crucial. Model retraining in traditional AI/ML refers to updating a deployed ML model with new data, generally performed to eliminate two types of drifts that occur. (1) Concept drift – when the link between the input variables and the target variables changes over time, since the description of what we want to predict changes, the model can produce inaccurate predictions. (2) Data drift – occurs when the characteristics of the input data change, like changes in customer habits or behavior over time and therefore the model’s inability to respond to such changes. In a similar fashion, retraining applies to FMs/LLMs, however it can be a lot costlier.
(in $millions), therefore not something most organizations might consider. It is under active research, still emerging in the realm of LLMOps. So instead of re-training, when model decay occurs in fine-tuned FMs, businesses may opt for fine-tuning again (lot cheaper) with a newer dataset. For a cost perspective, listed below is an example of a model-price table from Azure-OpenAI Services. For each task category, customers can fine-tune and evaluate models on specific datasets.

Source: Microsoft Azure

<table>
<thead>
<tr>
<th>Model</th>
<th>Per 1000 token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text-Ada</td>
<td>$0.0001</td>
</tr>
<tr>
<td>GPT-3.5 Turbo</td>
<td>$0.003</td>
</tr>
<tr>
<td>GPT-4</td>
<td>$0.06</td>
</tr>
<tr>
<td>Text-Davinci</td>
<td>$0.02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Per 100 images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dall-E</td>
<td>$2</td>
</tr>
</tbody>
</table>

Prompt engineering and Inferencing

Prompt engineering refers to the effective methods of how to communicate with LLMs to perform desired tasks without updating the model weights. As important as AI model training and fine-tuning is to NLP applications, inferencing is equally important, where the trained models respond to user prompts. The system requirements for inferencing are generally much more on the read performance of the AI storage system that feeds data from LLMs to the GPUs as it needs to be able to apply billions of stored model parameters to produce the best response.

LLMOps, Model Monitoring and Vectorstores

Like traditional Machine Learning Ops (MLOps), Large Language Model Operations (LLMOps) also require the collaboration of data scientists and DevOps engineers with tools and best practices for the management of LLMs in production environments. However, the workflow and tech stack for LLMs could vary in some ways. For instance, LLM pipelines built using frameworks like LangChain string together multiple LLM API calls to external embedding endpoints such as vectorstores or vector databases. The use of an embedding endpoint and vectorstore for downstream connectors (like to a vector database) represents a significant development in how data is stored and accessed. As opposed to traditional ML models that are developed from scratch, LLMs often rely on transfer learning since these models start with FMs that are fine-tuned with new data to improve performance in a more specific domain. Therefore, it is crucial LLMOps deliver the capabilities of risk management and model decay monitoring.

Risks and Ethics in the age of Generative AI

“ChatGPT – it’s slick but still spews nonsense.”– MIT Tech Review. Garbage in–garbage out, has always been the challenging case with computing. The only difference with generative AI is that it excels at making the garbage highly credible, leading to inaccurate outcomes. LLMs are prone to invent facts to fit the narrative it’s building. Therefore, companies that see generative AI as a great opportunity to lower their costs with AI equivalents need to efficiently detect deep fakes, reduce biases, and lower risks to keep the systems honest and ethical. A free-flowing data pipeline with a robust AI infrastructure that supports data mobility, data quality, data governance and data protection via end-to-end encryption and AI guardrails is eminent in the design of
responsible and explainable generative AI models.

Customer scenario and NetApp

Figure 3: Machine Learning/Large Language Model Workflow

Are we training or fine-tuning? The question of whether to (a) train an LLM model from scratch, fine-tune a pre-trained FM, or use RAG to retrieve data from document repositories outside a foundation model and augment prompts, and (b) either by leveraging open-source LLMs (e.g., Llama 2) or proprietary FMs (E.g., ChatGPT, Bard, AWS Bedrock) is a strategic decision for organizations. Each approach has a tradeoff between cost-efficiency, data gravity, operations, model accuracy and management of LLMs.

NetApp as a company embraces AI internally in its work culture and in its approach to product design and engineering efforts. For instance, NetApp’s autonomous ransomware protection is built using AI and machine learning. It provides early detection of file system anomalies to help identify threats before they impact operations. Second, NetApp uses predictive AI for its business operations like sales and inventory forecasting and chatbots to assist customers in call center product support services, tech specs, warranty, service manuals, and more. Third, NetApp brings customer value to the AI data pipeline and ML/LLM workflow via products and solutions serving customers building predictive AI solutions such as demand forecasting, medical imaging, sentiment analysis, and generative AI solutions like GANs for industrial images anomaly detection in manufacturing sector and anti-money laundering and fraud detection in banking & financial services with NetApp products and capabilities like NetApp® ONTAP AI®, NetApp® SnapMirror®, and NetApp® FlexCache®.

NetApp capabilities

The movement and management of data in generative AI applications such as chatbot, code generation, image generation or genome model expression can span across the edge, private data center, and hybrid multicloud ecosystem. For instance, a real-time AI-bot helping a passenger upgrade his or her airline ticket to business class from an end-user app exposed via APIs of pre-trained models such as ChatGPT cannot achieve that task by itself since the passenger information is not publicly available on the internet. The API
requires access to the passenger’s personal info and ticket info from the airline carrier which may exist in a hybrid or multicloud ecosystem. A similar scenario might apply to scientists sharing a drug molecule and patient data via an end-user application that uses LLMs to accomplish clinical trials across drug discovery involving one-to-many bio-medical research institutions. Sensitive data that gets passed to FMs or LLMs may include PII, financial information, health information, biometric data, location data, communications data, online behavior, and legal information. In such an event of real-time rendering, prompt execution and edge inferencing there is data movement from end-user app to storage endpoints via open source or proprietary LLM models to a data center on premises or public cloud platforms. In all such scenarios, data mobility and data protection are crucial for the AI operations involving LLMs which rely on large training datasets and movement of such data.

Figure 4: Generative AI - LLM Data Pipeline

NetApp’s portfolio of storage infrastructure, data and cloud services is powered by intelligent data management software.

Data Preparation: The first pillar of the LLM tech stack is largely untouched from the older traditional ML stack. Data preprocessing in AI pipeline is necessary to normalize and cleanse the data before training or fine-tuning. This step includes connectors to ingest data wherever it may reside in the form of an Amazon S3 tier or in on-premises storage systems such as a file store or an object store like NetApp StorageGRID.

NetApp® ONTAP is the foundational technology that underpins NetApp’s critical storage solutions in the data center and the cloud. ONTAP includes various data management and protection features and capabilities, including automatic ransomware protection against cyber-attacks, built-in data transport features, and storage efficiency capabilities for a range of architectures from on-premises, hybrid, multiclouds in NAS, SAN, object, and software defined storage (SDS) situations of LLM deployments.

NetApp® ONTAP AI® for deep learning model training. NetApp® ONTAP® supports NVIDIA GPU Direct Storage™ with the use of NFS over RDMA for NetApp customers with ONTAP storage cluster and NVIDIA DGX compute nodes. It offers a cost-efficient performance to read and process source datasets from storage into memory numerous times to foster intelligence, enabling organizations with training, fine-tuning, and scaling access to LLMs.
NetApp® FlexCache® is a remote caching capability that simplifies file distribution and caches only the actively read data. This can be useful for LLM training, re-training, and fine tuning, bringing value to customers with business requirements like real-time rendering and LLM inferencing.

NetApp® SnapMirror is an ONTAP feature that replicates volume snapshots between any two ONTAP systems. This feature optimally transfers data at the edge to your on-premises data center or to the cloud. SnapMirror can be used for moving data securely and efficiently between on-premises and hyperscaler clouds, when customers want to develop generative AI in clouds with RAG containing enterprise data. It efficiently transfers only changes, saving bandwidth and speeding replication, thus bringing essential data mobility features during the operations of training, re-training, and fine-tuning of FMs or LLMs.

NetApp® SnapLock brings immutable disk capability on ONTAP-based storage systems for dataset versioning. The microcore architecture is designed to protect customer data with FPolicy™ Zero Trust engine. NetApp ensures customer data is available by resisting denial-of-service (DoS) attacks when an attacker interacts with an LLM in a particularly resource-consuming way.

NetApp® Cloud Data Sense helps identify, map, and classify personal information present in enterprise datasets, enact policies, meet privacy requirements on premises or in the cloud, help improve security posture and comply with regulations.

NetApp® BlueXP™ classification, powered by Cloud Data Sense. Customers can automatically scan, analyze, categorize, and act on data across data estate, detect security risks, optimize storage, and accelerate cloud deployments. It combines storage and data services via its unified control plane. Customers can use GPU instances for computation, and hybrid multicloud environments for cold storage tiering and for archives and backups.

NetApp File-Object Duality. NetApp ONTAP enables dual-protocol access for NFS and S3. With this solution, customers can access NFS data from Amazon AWS SageMaker notebooks via S3 buckets from NetApp Cloud Volumes ONTAP. This offers flexibility to customers who need easy access to heterogenous data sources with the ability to share data from both NFS and S3. For e.g., fine-tuning FMs like Meta’s Llama 2 text-generation models on SageMaker with access to file-object buckets.

NetApp® Cloud Sync service offers a simple and secure way to migrate data to any target, in the cloud or on-premises. Cloud Sync seamlessly transfers and synchronizes data between on-premises or cloud storage, NAS, and object stores.

NetApp XCP is a client software that enables fast and reliable any-to-NetApp and NetApp-to-NetApp data migrations. XCP also provides the capability of moving bulk data efficiently from Hadoop HDFS file systems into ONTAP NFS, S3 or StorageGRID and XCP file analytics provides visibility into the file system.

NetApp® DataOps Toolkit is a Python library that makes it simple for data scientists, DevOps, and data engineers to perform various data management tasks, such as near-instantaneously provisioning, cloning, or snapshotting a data volume or JupyterLab workspace that are backed by high-performance scale-out NetApp storage.

NetApp’s product security. LLMs may inadvertently reveal confidential data in their responses, thus a concern to CISOs who study the vulnerabilities associated with AI applications leveraging LLMs. As outlined by OWASP (Open Worldwide Application Security Project), security issues such as data poisoning, data leakage, denial of service and prompt injections within LLMs can impact businesses from data exposure to unauthorized access serving attackers. Data storage requirements should include integrity checks and immutable snapshots for structured, semi-structured, and unstructured data. NetApp Snapshots and SnapLock are being used for dataset versioning. It brings strict role-based access control (RBAC), as well as secure protocols, and industry standard encryption for securing both data at rest and in transit. Cloud Insights and Cloud Data Sense together offer capabilities to help you forensically identify the source of the threat and prioritize which data to restore.
ONTAP AI with DGX BasePOD

NetApp® ONTAP® AI reference architecture with NVIDIA DGX BasePOD is a scalable architecture for machine learning (ML) and artificial intelligence (AI) workloads. For the critical training phase of LLMs, data is typically copied from the data storage into the training cluster at regular intervals. The servers that are used in this phase use GPUs to parallelize computations, creating a tremendous appetite for data. Meeting the raw I/O bandwidth needs is crucial for maintaining high GPU utilization.

ONTAP AI with NVIDIA AI Enterprise

NVIDIA AI Enterprise is an end-to-end, cloud-native suite of AI and data analytics software that is optimized, certified, and supported by NVIDIA to run on VMware vSphere with NVIDIA-Certified Systems. This software facilitates the simple and rapid deployment, management, and scaling of AI workloads in the modern hybrid cloud environment. NVIDIA AI Enterprise, powered by NetApp and VMware, delivers enterprise-class AI workload and data management in a simplified, familiar package.

1P Cloud Platforms

Fully managed cloud storage offerings are available natively on Microsoft Azure as Azure NetApp Files (ANF), on AWS as Amazon FSx for NetApp ONTAP (FSxN), and on Google as Google Cloud NetApp Volumes (GNCV). 1P is a managed, high-performance file system that enables customers to run highly available AI workloads with improved data security in public clouds, for fine-tuning LLMs/FMs with cloud native ML platforms like AWS SageMaker, Azure-OpenAI Services, and Google’s Vertex AI.

NetApp Partner Solution Suite

In addition to its core data products, technologies and capabilities, NetApp also collaborates closely with a robust network of AI partners to bring added value to customers.

NVIDIA Guardrails in AI systems serve as safeguards to ensure the ethical and responsible use of AI technologies. AI developers can choose to define the behavior of LLM-powered applications on specific topics and prevent them from engaging in discussions on unwanted topics. Guardrails, an open-source toolkit, provides the ability to connect an LLM to other services, seamlessly and securely for building trustworthy, safe, and secure LLM conversational systems.

Domino Data Lab provides versatile, enterprise-grade tools for building and productizing Generative AI - fast, safe, and economical, wherever you are in your AI journey. With Domino’s Enterprise MLOps Platform, data scientists can use preferred tools and all their data, train and deploy models easily anywhere and manage risk and cost effectively - all from one control center.

Modzy for Edge AI. NetApp® and Modzy have partnered together to deliver AI at scale to any type of data, including imagery, audio, text, and tables. Modzy is an MLOps platform for deploying, integrating, and running AI models, offers data scientists the capabilities of model monitoring, drift detection and explainability, with an integrated solution for seamless LLM inference.

Run:AI and NetApp have partnered to demonstrate the unique capabilities of the NetApp ONTAP AI solution with the Run:AI cluster management platform for simplifying orchestration of AI workloads. It automatically splits and joins GPU resources, designed to scale your data processing pipelines to hundreds of machines with built-in integration frameworks for Spark, Ray, Dask, and Rapids.

Conclusion

Generative AI can produce effective results only when the model is trained on reams of quality data. While LLMs have achieved remarkable milestones, it is critical to recognize its limitations, design challenges and risks associated with data mobility and data quality. LLMs rely on large and disparate training datasets from
heterogenous data sources. Inaccurate outcomes or biased results generated by the models can put both businesses and consumers in jeopardy. These risks can correspond to constraints for LLMs emerging potentially from data management challenges associated with data quality, data security, and data mobility. NetApp helps organizations meet the complexities created by rapid data growth, data mobility, multi-cloud management, and the adoption of AI. At scale AI infrastructure and efficient data management is crucial to defining the success of AI applications like generative AI. It is critical customers cover all the deployment scenarios without compromising on the ability to expand as enterprises need to while maintaining cost-efficiency, data governance and ethical AI practices in control. NetApp is constantly working to help customers simplify and accelerate their AI deployments.

**TR-4785: AI Deployment with NetApp E-Series and BeeGFS**

Nagalakshmi Raju, Daniel Landes, Nathan Swartz, Amine Bennani, NetApp

Artificial intelligence (AI), machine learning (ML), and deep learning (DL) applications involve large datasets and high computations. To run these workloads successfully, you need an agile infrastructure that allows you to scale out both storage and compute nodes seamlessly. This report includes the steps for running an AI training model in a distributed mode, which allows seamless scale-out of compute and storage nodes. The report also includes various performance metrics to show how a solution combining NetApp E-Series storage with the BeeGFS parallel file system provides a flexible, cost-effective, and simple solution for AI workloads.

**NVA-1150-DESIGN: Quantum StorNext with NetApp E-Series systems design guide**

Ryan Rodine, NetApp

This document provides details on how to design a StorNext parallel file system solution with NetApp E-Series storage systems. This solution covers the NetApp EF280 all-flash array, the NetApp EF300 all-flash NVMe array, the EF600 all-flash NVMe array, and the NetApp E5760 hybrid system. It offers performance characterization based on Frametest benchmarking, a tool that is widely used for testing in the media and entertainment industry.

**NVA-1150-DEPLOY: Quantum StorNext with NetApp E-Series systems deployment guide**

Ryan Rodine, NetApp

This document provides details on how to deploy a StorNext parallel file system solution with NetApp E-Series storage systems. This solution covers the NetApp EF280 all-flash array, the NetApp EF300 all-flash NVMe array, the NetApp EF600 all-flash NVMe array, and the NetApp E5760 hybrid system. It offers performance characterization based on Frametest benchmarking, a tool that is widely used for testing in the media and entertainment industry.