



Use Cases

NetApp Solutions

NetApp
July 11, 2022

This PDF was generated from <https://docs.netapp.com/us-en/netapp-solutions/ai/ai-protopia-overview.html> on July 11, 2022. Always check docs.netapp.com for the latest.

Table of Contents

Use Cases	1
Responsible AI and confidential inferencing - NetApp AI with Protopia Image Transformation	1
Sentiment analysis with NetApp AI	27
Distributed training in Azure - Click-Through Rate Prediction	45
TR-4896: Distributed training in Azure: Lane detection - Solution design	69
TR-4841: Hybrid Cloud AI Operating System with Data Caching	98
AI Inferencing at the Edge - NetApp with Lenovo ThinkSystem - Solution Design	121
WP-7328: NetApp Conversational AI Using NVIDIA Jarvis	144
TR-4858: NetApp Orchestration Solution with Run:AI	164

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.

[Next: Solution areas.](#)

Solution areas

[Previous: Overview.](#)

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](#).

Next: [Technology overview](#).

Technology overview

[Previous: Solution areas.](#)

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 Cloud Sync

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

NetApp Cloud Data Sense

Driven by powerful AI algorithms, [NetApp Cloud Data Sense](#) 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 Cloud Data Sense 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.

[Next: Test and validation plan.](#)

Test and validation plan

[Previous: Technology overview.](#)

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.

[Next: Test configuration.](#)

Test configuration

[Previous: Test and validation plan.](#)

The following table outlines the solution design validation environment.

Component	Version
Kubernetes	1.21.6
NetApp Astra Trident CSI Driver	22.01.0
NetApp DataOps Toolkit for Kubernetes	2.3.0
NVIDIA Triton Inference Server	21.11-py3

[Next: Test procedure.](#)

Test procedure

[Previous: Test configuration.](#)

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.

Launcher X image-demo-pytorch.ipynb X

Code Python 3 (ipykernel) ⚡

STEP 3-1: Clean (Without obfuscation) detection

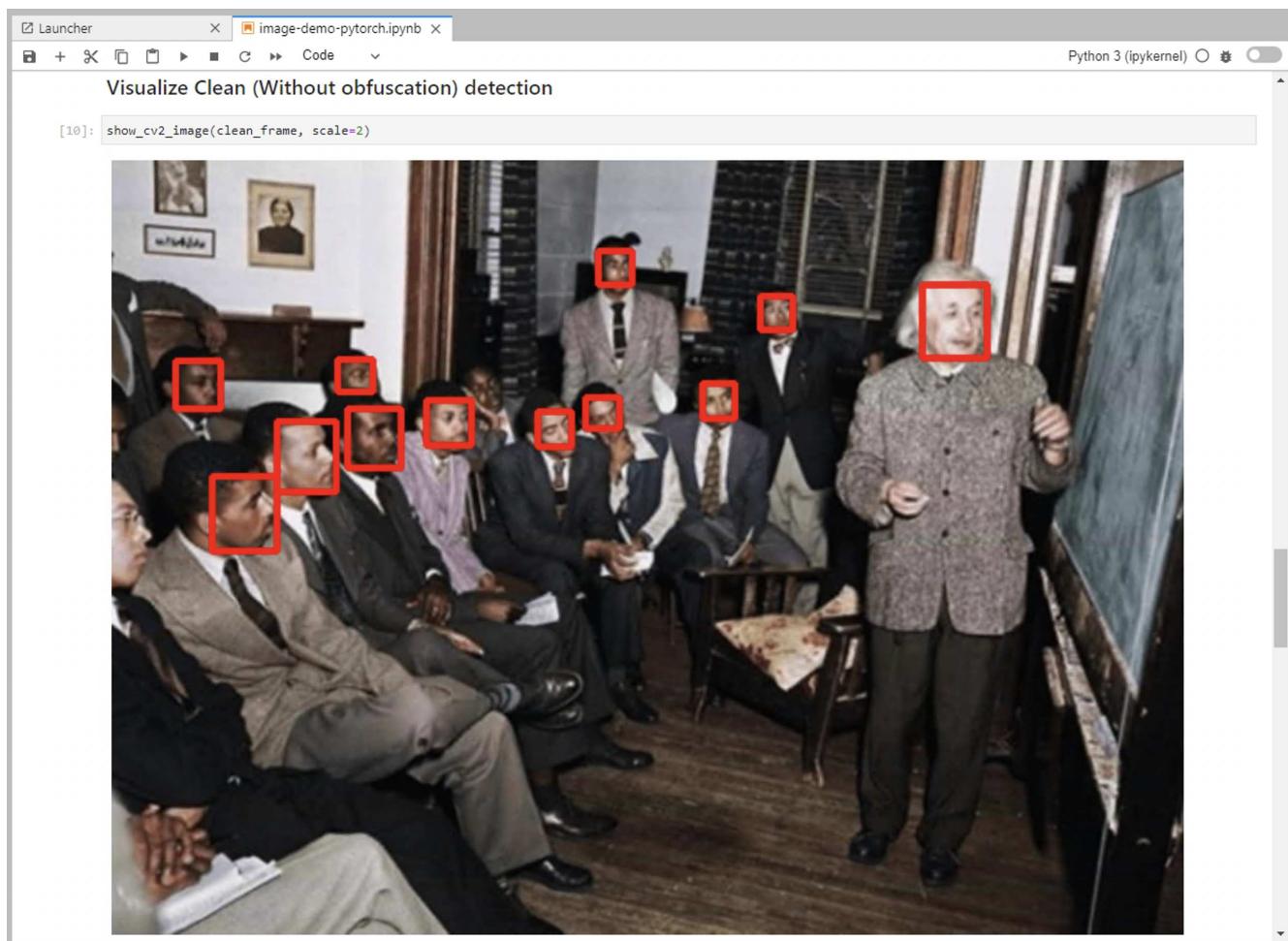
```
[9]: # get current frame
frame = input_image

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

# run forward pass
clean_activation = clean_model.forward_head(preprocessed_input) # runs the first few layers
loc, pred = clean_model.forward_tail(clean_activation) # runs rest of the layers

# postprocess output
clean_pred = (loc.detach().cpu().numpy(), pred.detach().cpu().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)
```



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.

Launcher image-demo-pytorch.ipynb Python 3 (ipykernel)

STEP 3-2: Protopia AI (With obfuscation) detection

```
[11]: # 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.forward_head(preprocessed_input) # runs the first few layers
#####
# SINGLE ADDITIONAL LINE FOR PRIVATE INFERENCE #
#####
noisy_activation = noisy_model.forward_noise(not_noisy_activation)
#####
loc, pred = noisy_model.forward_tail(noisy_activation) # runs rest of the layers

# postprocess output
noisy_pred = (loc.detach().cpu().numpy(), pred.detach().cpu().numpy())
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_activation)
noisy_reconstruction = noisy_reconstruction.detach().cpu().numpy()[0]
noisy_reconstruction = unprocess_output(
    noisy_reconstruction, (input_image_width, input_image_height), True
).astype(np.uint8)

# draw rectangles
for (x1, y1, x2, y2, s) in noisy_outputs[0]:
    x1, y1 = int(x1), int(y1)
    x2, y2 = int(x2), int(y2)
    cv2.rectangle(noisy_reconstruction, (x1, y1), (x2, y2), (0, 0, 255), 4)
```

Launcher image-demo-pytorch.ipynb Python 3 (ipykernel)

Visualize Protopia AI (With obfuscation) detection

```
[12]: 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.

```
$ vi inference-job-raw.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-inference-raw
  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
          command: ["python3", "run-accuracy-measurement.py", "--dataset",
                    "/data/netapp-face-detection/FDDB"]
          resources:
            limits:
              nvidia.com/gpu: 2
      volumeMounts:
        - mountPath: /data
          name: data
        - mountPath: /dev/shm
          name: dshm
      restartPolicy: Never
$ kubectl create -f inference-job-raw.yaml
job.batch/netapp-inference-raw created
```

5. Confirm that the inferencing job completed successfully.

```
$ kubectl -n inference logs netapp-inference-raw-255sp
100%|██████████| 89/89 [00:52<00:00, 1.68it/s]
Reading Predictions : 100%|██████████| 10/10 [00:01<00:00, 6.23it/s]
Predicting ... : 100%|██████████| 10/10 [00:16<00:00, 1.64s/it]
===== Results =====
FDDB-fold-1 Val AP: 0.9491256561145955
FDDB-fold-2 Val AP: 0.9205024466101926
FDDB-fold-3 Val AP: 0.9253013871078468
FDDB-fold-4 Val AP: 0.9399781485863011
FDDB-fold-5 Val AP: 0.9504280149478732
FDDB-fold-6 Val AP: 0.9416473519339292
FDDB-fold-7 Val AP: 0.9241631566241117
FDDB-fold-8 Val AP: 0.9072663297546659
FDDB-fold-9 Val AP: 0.9339648715035469
FDDB-fold-10 Val AP: 0.9447707905560152
FDDB Dataset Average AP: 0.9337148153739079
=====
mAP: 0.9337148153739079
```

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", "$(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: 0.9101178994188819
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 an 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],
```

```

    "FP32"))
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.

```
# 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,
```

```

        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
noisy_pred = (loc_numpy, pred_numpy)
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_activation)
noisy_reconstruction = noisy_reconstruction.detach().cpu().numpy()[0]
noisy_reconstruction = unpreprocess_output(
    noisy_reconstruction, (input_image_width, input_image_height), True
).astype(np.uint8)
# draw rectangles
for (x1, y1, x2, y2, s) in noisy_outputs[0]:
    x1, y1 = int(x1), int(y1)
    x2, y2 = int(x2), int(y2)
    cv2.rectangle(noisy_reconstruction, (x1, y1), (x2, y2), (0, 0, 255),
4)

```

[Next: Inferencing accuracy comparison.](#)

Inferencing accuracy comparison

[Previous: Test procedure.](#)

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.

Component	Details
Model	FaceBoxes (PyTorch) -
Dataset	FDDB dataset

Protopia obfuscation	ALPHA	Accuracy
No	N/A	0.9337148153739079
Yes	0.05	0.9028766627325002
Yes	0.1	0.9024301009661478
Yes	0.2	0.9081836283186224
Yes	0.4	0.9073066107482036
Yes	0.6	0.8847816568680239
Yes	0.8	0.8841195749171925
Yes	0.9	0.8455427675252052
Yes	0.95	0.8455427675252052

[Next: Obfuscation speed.](#)

Obfuscation speed

[Previous: Inferencing accuracy comparison.](#)

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.

[Next: Conclusion.](#)

Conclusion

[Previous: Obfuscation speed.](#)

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.

Next: [Where to find additional information, acknowledgements, and version history](#).

Where to find additional information, acknowledgements, and version history

Previous: [Conclusion](#).

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

<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>

- NetApp Persistent Storage for Containers—NetApp Trident

<https://netapp.io/persistent-storage-provisioner-for-kubernetes/>

- NetApp DataOps Toolkit

<https://github.com/NetApp/netapp-dataops-toolkit>

- NetApp Persistent Storage for Containers—NetApp Astra Trident

<https://netapp.io/persistent-storage-provisioner-for-kubernetes/>

- Protopia AI—Confidential Inference

<https://protopia.ai/blog/protopia-ai-takes-on-the-missing-link-in-ai-privacy-confidential-inference/>

- NetApp Cloud Sync

https://docs.netapp.com/us-en/occm/concept_cloud_sync.html#how-cloud-sync-works

- NVIDIA Triton Inference Server

<https://developer.nvidia.com/nvidia-triton-inference-server>

- NVIDIA Triton Inference Server Documentation

<https://docs.nvidia.com/deeplearning/triton-inference-server/index.html>

- FaceBoxes in PyTorch

<https://github.com/zisianw/FaceBoxes.PyTorch>

Acknowledgments

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

Version history

Version	Date	Document Version History
Version 1.0	May 2022	Initial release.

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.

[Next: Use cases.](#)

Use cases

[Previous: Support center analytics.](#)

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.

[Next: Architecture.](#)

Architecture

[Previous: Use cases.](#)

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 Cloud 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 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.

Response latency tests	Time (milliseconds)
Data processing	10
Inferencing	10

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

Hardware	Requirements
OS	Linux x86_64
GPU memory (ASR)	Streaming models: ~5600 MB Non-streaming models: ~3100 MB
GPU memory (NLP)	~500MB per BERT model

NVIDIA TAO Toolkit hardware

Hardware	Requirements
System RAM	32GB
GPU RAM	32GB
CPU	8 core
GPU	NVIDIA (A100, V100 and RTX 30x0)
SSD	100GB

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 Cloud Sync

[Cloud 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

Cloud 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. Cloud 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. Cloud Sync is a software as a service (SaaS) tool that is simple to set up and use. Data transfers that are triggered by Cloud Sync are carried out by data brokers. You can deploy Cloud 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.

Host machine	Requirements
RIVA (formerly JARVIS)	1.4.0
TAO Toolkit (formerly Transfer Learning Toolkit)	3.0
ONTAP	9.9.1
DGX OS	5.1
DOTK	2.0.0

NVIDIA RIVA Software

Software	Requirements
Docker	>19.02 (with nvidia-docker installed)>=19.03 if not using DGX
NVIDIA Driver	465.19.01+ 418.40+, 440.33+, 450.51+, 460.27+ for Data Center GPUs
Container OS	Ubuntu 20.04
CUDA	11.3.0
cuBLAS	11.5.1.101
cuDNN	8.2.0.41
NCCL	2.9.6
TensorRT	7.2.3.4
Triton Inference Server	2.9.0

NVIDIA TAO Toolkit software

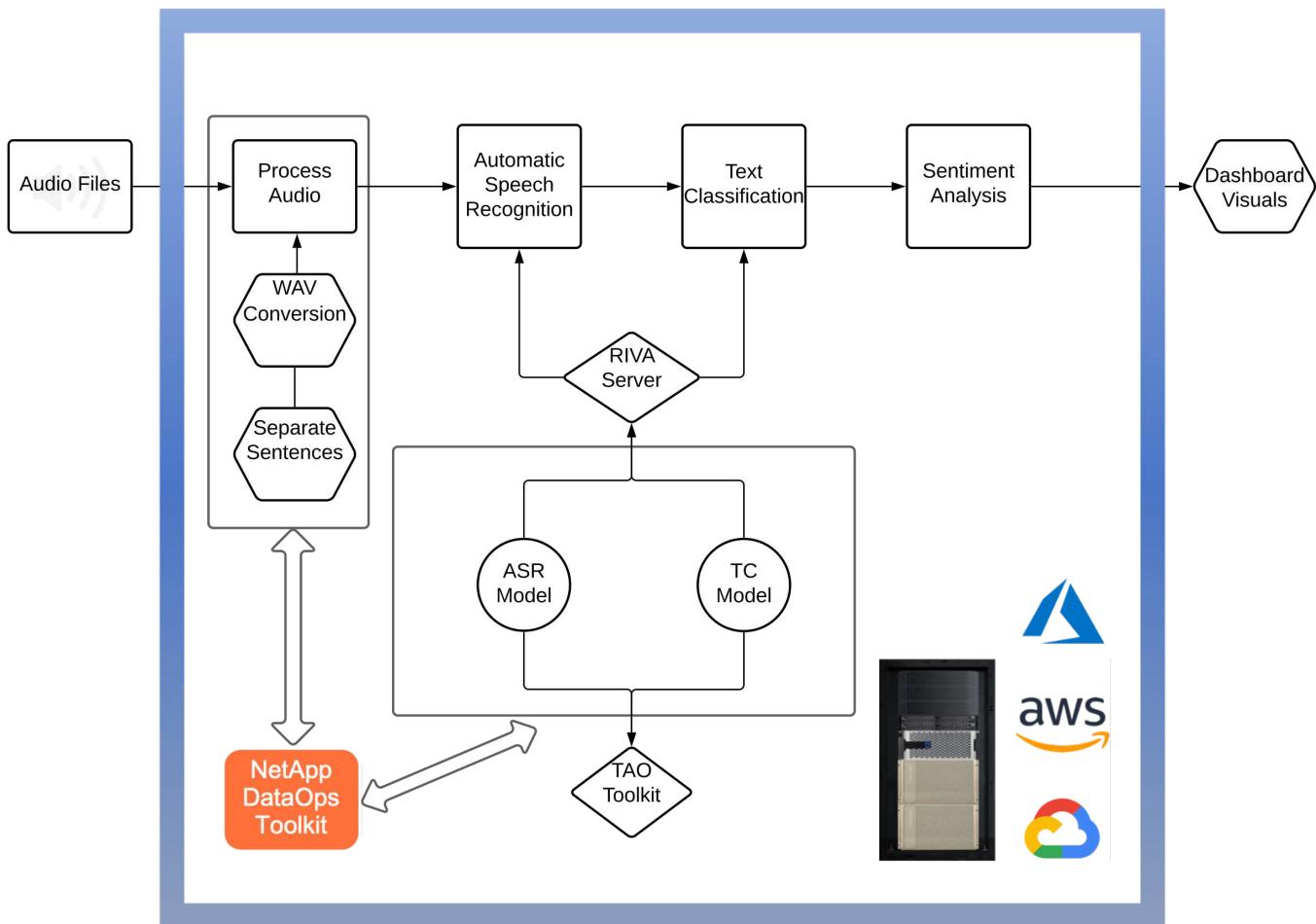
Software	Requirements
Ubuntu 18.04 LTS	18.04
python	>=3.6.9
docker-ce	>19.03.5
docker-API	1.40
nvidia-container-toolkit	>1.3.0-1
nvidia-container-runtime	3.4.0-1

Software	Requirements
nvidia-docker2	2.5.0-1
nvidia-driver	>455
python-pip	>21.06
nvidia-pyindex	Latest version

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.

[Next: Design considerations.](#)

Design considerations

[Previous: Architecture.](#)

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.

[Next: Deploying support-center sentiment analysis.](#)

Deploying support center sentiment analysis

[Previous: Design considerations.](#)

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.

```
python3 -m pip install netapp-dataops-traditional
```

2. Configure the data management

```
netapp_dataops_cli.py config
```

NVIDIA NGC configuration: Support center sentiment analysis

To set up [NVIDIA NGC](#), complete the following steps:

1. Download the NGC.

```
wget -O ngccli_linux.zip  
https://ngc.nvidia.com/downloads/ngccli_linux.zip && unzip -o  
ngccli_linux.zip && chmod u+x 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.

[Next: Validation results.](#)

Validation results

[Previous: Deploying support-center sentiment analysis.](#)

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

Test case	Details
Test number	Pipeline sentiment metric
Test prerequisites	Fine-tuned models for speech-to-text and sentiment analysis models
Expected outcome	The sentiment metric of the fine-tuned model performs better than the original pretrained model.

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.

Next: [Videos and demos](#).

Videos and demos

Previous: [Validation results](#).

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.

Call Center - Model Transfer Learning and Fine-Tuning

TAO Toolkit is a python based AI toolkit for taking purpose-built pre-trained AI models and customizing them with your own data. Transfer learning extracts learned features from an existing neural network to a new one. Transfer learning is often used when creating a large training dataset is not feasible in order to enhance the base performance of state-of-the-art models.

For this call center solution, the speech-to-text and sentiment analysis models are fine-tuned on call center data to augment the model performance on business specific terminology.

For more information on the TAO Toolkit, please visit [here](#).



Installing necessary dependencies

For ease of use, please install TAO Toolkit inside a python virtual environment. We recommend performing this step first and then launching the notebook from the virtual environment. Please refer to the README for these instructions.

Next: Conclusion.

Conclusion

Previous: [Videos and demos](#).

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.

Next: [Where to find additional information](#).

Where to find additional information

Previous: [Conclusion](#).

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

<https://www.netapp.com/pdf.html?item=/media/32792-DS-4145-NVIDIA-Base-Command-Platform-with-NetApp.pdf>

- NetApp for AI 10 Good Reasons infographic

<https://www.netapp.com/us/media/netapp-ai-10-good-reasons.pdf>

- AI in Healthcare: Deep learning to identify COVID-19 lesions in lung CT scans white paper

<https://www.netapp.com/pdf.html?item=/media/31240-WP-7342.pdf>

- AI in Healthcare: Monitoring face mask usage in healthcare settings white paper

<https://www.netapp.com/pdf.html?item=/media/37490-NA-611-Monitoring-face-mask-usage-in-healthcare-settings.pdf>

- AI in Healthcare: Diagnostic Imaging Technical Report

<https://www.netapp.com/pdf.html?item=/media/7395-tr4811.pdf>

- AI for Retail: NetApp Conversational AI using NVIDIA RIVA
https://docs.netapp.com/us-en/netapp-solutions/ai/cainvidia_executive_summary.html
- NetApp ONTAP AI solution brief
<https://www.netapp.com/pdf.html?item=/media/6736-sb-3939.pdf>
- NetApp DataOps Toolkit solution brief
<https://www.netapp.com/pdf.html?item=/media/21480-SB-4111-1220-NA-Data-Science-Toolkit.pdf>
- NetApp AI Control Plane solution brief
<https://www.netapp.com/pdf.html?item=/media/6737-sb-4055.pdf>
- Transforming Industry with Data Drive AI eBook
<https://www.netapp.com/us/media/na-337.pdf>
- NetApp EF-Series AI solution brief
<https://www.netapp.com/pdf.html?item=/media/26708-SB-4136-NetApp-AI-E-Series.pdf>
- NetApp AI and Lenovo ThinkSystem for AI Inferencing solution brief
<https://www.netapp.com/pdf.html?item=/media/25316-SB-4129.pdf>
- NetApp AI and Lenovo ThinkSystem for enterprise AI and ML solution brief
<https://www.netapp.com/pdf.html?item=/media/25317-SB-4128.pdf>
- NetApp and NVIDIA – Redefining What is Possible with AI video
<https://www.youtube.com/watch?v=38xw65SteUc>

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.



[Next: Technology overview.](#)

Technology overview

[Previous: Introduction.](#)

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.



[Next: Software requirements.](#)

Software requirements

[Previous: Technology overview.](#)

The following table lists the software requirements needed for this solution.

Software	Version
Azure Kubernetes Service	1.18.14
RAPIDS and Dask container image	Repository: "rapidsai/rapidsai" Tag: 0.17-cuda11.0-runtime-ubuntu18.04
NetApp Trident	20.01.1
Helm	3.0.0

[Next: Cloud resource requirements.](#)

Cloud resource requirements

[Previous: Software requirements.](#)

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.

Resources	Type/version
Azure Kubernetes Service	1.18.14
Agent node	3x Standard_DS2_v2
GPU node	3x Standard_NC6s_v3
Azure NetApp Files	Standard capacity pool
Capacity in TB	4

Next: [Click-through rate prediction use case summary](#).

Click-through rate prediction use case summary

Previous: [Cloud resource requirements](#).

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.

[Next: Install and set up the aks cluster.](#)

Setup

Install and set up the AKS cluster

[Previous: Click-through rate prediction use case summary.](#)

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 gppool.

Name	Mode	OS type	Node count	Node size
agentpool	System	Linux	3	Standard_DS2_v2
gppool	User	Linux	3	Standard_NC6s_v3

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 xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
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

```
verronmartina@verron-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
verronmartina@verron-mac-0 ~ %
```

Next: [Create a delegated subnet for Azure NetApp Files](#).

Create a delegated subnet for Azure NetApp Files

[Previous: Install and set up the AKS cluster.](#)

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.

Microsoft Azure

Search resources, services, and docs (G+/-)

Dashboard > Virtual networks

seanlucealive (Default Directory)

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.

Name	Resource group	Location	Subscription
aks-vnet-22885919	MC_sluce.rg_TridentDemo_eastus2	East US 2	AzureSub01

No grouping List view

3. Click Subnets and click +Subnet from the top toolbar.



Microsoft Azure

Search resources, services, and docs (G+/-)

Dashboard > Virtual networks > aks-vnet-22885919

aks-vnet-22885919 | Subnets

+ Subnet + Gateway subnet Refresh Manage users Delete

Search subnets

Name	IPv4	IPv6 (many availab...)	Delegated to	Security group
aks-subnet	10.240.0.0/16 (65530 av...)	-	-	aks-agentpool-2288591...

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.



Add subnet

X

Name *

ANF.sn



Subnet address range * ⓘ

10.0.0.0/24

10.0.0.0 - 10.0.0.255 (251 + 5 Azure reserved addresses)

Add IPv6 address space ⓘ

NAT gateway ⓘ

None



Network security group

None



Route table

None



SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. [Learn more](#)

Services ⓘ

0 selected



SUBNET DELEGATION

Delegate subnet to a service ⓘ

Microsoft.Netapp/volumes



OK

Cancel

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.

Next: Peer AKS vnet and Azure NetApp Files vnet.

Peer AKS VNet and Azure NetApp Files VNet

Previous: [Create a delegated subnet for Azure NetApp Files.](#)

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](#).

Next: [Install Trident](#).

Install Trident

Previous: [Peer AKS VNet and Azure NetApp Files VNet](#).

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.

```
$ helm install trident trident-operator-21.01.1.tgz --namespace  
trident --create-namespace
```

c. Check the status of Trident pods.

```
$ 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.

a. Create an Azure Service Principle.

The service principal is how Trident communicates with Azure to manipulate your Azure NetApp Files resources.

```
$ az ad sp create-for-rbac --name ""
```

The output should look like the following example:

```
{  
  "appId": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
  "displayName": "netapptrident",  
  "name": "",  
  "password": "xxxxxxxxxxxxxxxx.xxxxxxxxxxxxxx",  
  "tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
}
```

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:

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "fakec765-4774-fake-ae98-a721add4fake",
    "tenantID": "fakef836-edc1-fake-bff9-b2d865eefake",
    "clientID": "fake0f63-bf8e-fake-8076-8de91e57fake",
    "clientSecret": "SECRET",
    "location": "westeurope",
    "serviceLevel": "Standard",
    "virtualNetwork": "anf-vnet",
    "subnet": "default",
    "nfsMountOptions": "vers=3,proto=tcp",
    "limitVolumeSize": "500Gi",
    "defaults": {
        "exportRule": "0.0.0.0/0",
        "size": "200Gi"
    }
}
```

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:

```
$tridentctl create backend -f anf-backend.json -n trident
```

NAME	STORAGE DRIVER	UUID	STATE	VOLUMES
azurenappfiles_86181	azure-netapp-files	2ca85462-59ac-4946-be05-c03f5575a2ad	online	0

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 `azurenappfiles` 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: azurenetaappfiles
  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 azurenetaappfiles
```

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	AGE
azurenetaappfiles	csi.trident.netapp.io	Delete	Immediate	false	98s

Next: Set up Dask with RAPIDS deployment on AKS using Helm.

Set up Dask with RAPIDS deployment on AKS using Helm

Previous: Install Trident.

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: azurenetaappfiles
```

- b. Apply the YAML file to your Kubernetes cluster.

```
kubectl -n rapids-dask apply -f <your yaml file>
```

3. Clone the `rapidsai` git repository (<https://github.com/rapidsai/helm-chart>).

```
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.

```
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
```

[Next: Azure NetApp Files performance tiers.](#)

Azure NetApp Files performance tiers

[Previous: Set up Dask with RAPIDS deployment on AKS using Helm.](#)

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.

NFSv3	10.28.254.4:/norootfor	Standard	pool0	...
NFSv4.1	NAS-735a.docs.lab:/for	Premium		
NFSv4.1	NAS-735a.docs.lab:/krt	Premium		
NFSv3	10.28.254.4:/moveme0	Premium		
NFSv3	10.28.254.4:/placeholder	Premium		

- Resize
- Edit
- Change pool
- Delete

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:

```

Set-AzNetAppFilesVolumePool
-ResourceGroupName "MyRG"
-AccountName "MyAnfAccount"
-PoolName "MyAnfPool"
-Name "MyAnfVolume"
-NewPoolResourceId 7d6e4069-6c78-6c61-7bf6-c60968e45fbf

```

[Next: Libraries for data processing and model training.](#)

Click through rate prediction data processing and model training

Libraries for data processing and model training

[Previous: Azure NetApp Files performance tiers.](#)

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.

Libraries/framework	Description
Dask cuML	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.
Dask cuDF	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.
Scikit Learn	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.

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.

[Next: Load Criteo Click Logs day 15 in Pandas and train a scikit-learn random forest model.](#)

Load Criteo Click Logs day 15 in Pandas and train a scikit-learn random forest model

[Previous: Libraries for data processing and model training.](#)

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 notebook CTR-PandasRF-collated.ipynb creates a Pandas DataFrame that contains the first 50 million rows and generates a scikit-learn random forest model.

```
%%time
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](#).

```

# testing data, last 1M rows in day15
test_file = '/data/day_15_test'
with open(test_file) as g:
    print(g.readline())

# DataFrame processing for test data
test_df = pd.read_csv(test_file, delimiter='\t', names=header)
test_df_sliced = test_df.iloc[:, 0:14]
test_Y = test_df_sliced.pop('col1')
test_df_sliced = test_df_sliced.astype(np.float32).fillna(0)
# prediction & calculating error
pred_df = rf_model.predict(test_df_sliced)
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(test_Y, pred_df))

```

[Next: Load Day 15 in Dask and train a Dask cuML random forest model.](#)

Load Day 15 in Dask and train a Dask cuML random forest model

[Previous: Load Criteo Click Logs day 15 in Pandas and train a scikit-learn 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:

```

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

```

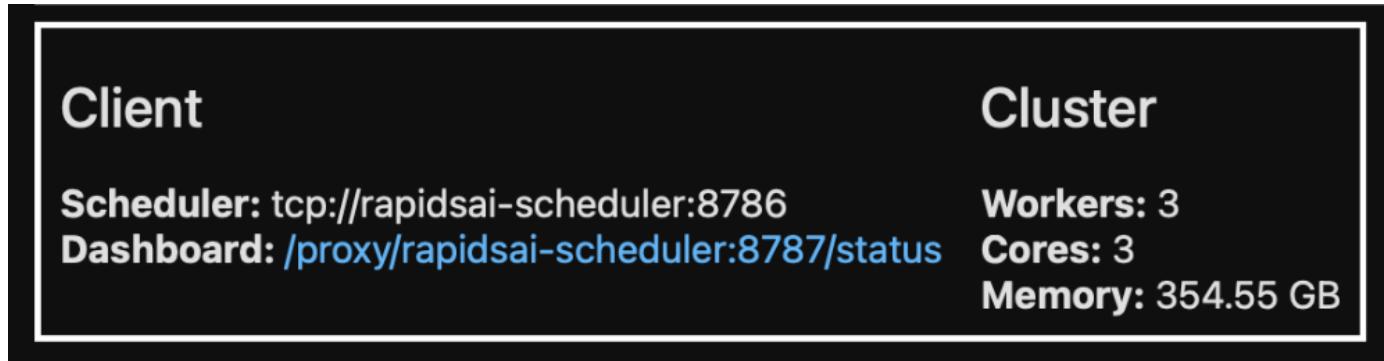
Initiate Dask Client().

```
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:



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.

```

Adsf
# Random Forest building parameters
n_streams = 8 # optimization
max_depth = 10
n_bins = 16
n_trees = 10
cuml_model = cumlDaskRF(max_depth=max_depth, n_estimators=n_trees,
n_bins=n_bins, n_streams=n_streams, verbose=True, client=client)
cuml_model.fit(gdf_sliced_small, Y)
# Model prediction
pred_df = cuml_model.predict(gdf_test)
# calculate accuracy
cu_score = cuml.metrics.accuracy_score( test_y, pred_df )

```

[Next: Monitor Dask using native Task Streams dashboard.](#)

Monitor Dask using native Task Streams dashboard

[Previous: Load Day 15 in Dask and train a Dask cuML random forest model.](#)

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.



[Next: Training time comparison.](#)

Training time comparison

[Previous: Monitor Dask using native Task Streams dashboard.](#)

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 day15 of the dataset). This sample is only using less than 0.25% of all available data. Whereas for RAPIDS-Dask we trained the random forest model on all 20 billion available rows. The two approaches yielded comparable training time.

Approach	Training time
Scikit-learn: Using only 50M rows in day15 as the training data	47 minutes and 21 seconds
RAPIDS-Dask: Using all 20B rows in day15 as the training data	1 hour, 12 minutes, and 11 seconds

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.

Approach	Training time
Scikit-learn: Using all 20B rows in day15 as the training data	13 days, 3 hours, 40 minutes, and 11 seconds
RAPIDS-Dask: Using all 20B rows in day15 as the training data	1 hour, 12 minutes, and 11 seconds

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.

[Next: Monitor Dask and RAPIDS with Prometheus and Grafana.](#)

Monitor Dask and RAPIDS with Prometheus and Grafana

[Previous: Training time comparison.](#)

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](#).

[Next: Dataset and Model Versioning using NetApp DataOps Toolkit.](#)

Dataset and model versioning using NetApp DataOps Toolkit

[Previous: Monitor Dask and RAPIDS with Prometheus and Grafana.](#)

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](#).

[Next: Conclusion.](#)

Jupyter notebooks for reference

[Previous: Dataset and Model Versioning using NetApp DataOps Toolkit.](#)

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.

[Next: Conclusion.](#)

Conclusion

Previous: [Dataset and Model Versioning using NetApp DataOps Toolkit.](#)

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.

Next: [Where to find additional information.](#)

Where to find additional information

Previous: [Conclusion.](#)

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
<https://netapptrident.readthedocs.io/en/stablev20.07/kubernetes/operations/tasks/backends/anf.html>
- 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
<https://docs.dask.org/en/latest/diagnostics-distributed.html>
- ML framework and tools:
 - TensorFlow: An Open-Source Machine Learning Framework for Everyone
<https://www.tensorflow.org/>
 - Docker
<https://docs.docker.com>
 - Kubernetes
<https://kubernetes.io/docs/home/>
 - Kubeflow
<http://www.kubeflow.org/>
 - Jupyter Notebook Server
<http://www.jupyter.org/>

Next: Version history.

Version history

Previous: Where to find additional information.

Version	Date	Document version history
Version 1.0	August 2021	Initial release.

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.

Cloud	Quantity
AKS	Minimum of three system nodes and three GPU worker nodes
Virtual machine (VM) SKU system nodes	Three Standard_DS2_v2
VM SKU GPU worker nodes	Three Standard_NC6s_v3
Azure NetApp Files	4TB standard tier

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.

Software	Version or other information
AKS - Kubernetes version	1.18.14
RUN:AI CLI	v2.2.25
RUN:AI Orchestration Kubernetes Operator version	1.0.109
Horovod	0.21.2
NetApp Trident	20.01.1
Helm	3.0.0

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.

Name	Mode	OS type	Node count	Node size
agentpool	System	Linux	3	Standard_DS2_v2
gpupool	User	Linux	3	Standard_NC6s_v3



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:

```
az account set --subscription xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  
aks get-credentials --resource-group resourcegroup --name aksclustername
```

6. Enter this command in the Azure CLI:

```
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.

```
verronmartina@verron-mac-0 ~ % kubectl get nodes  
NAME                      STATUS   ROLES      AGE     VERSION  
aks-agentpool-34613062-vmss000000  Ready    agent      22m    v1.18.14  
aks-agentpool-34613062-vmss000001  Ready    agent      22m    v1.18.14  
aks-agentpool-34613062-vmss000002  Ready    agent      22m    v1.18.14  
aks-gpupool-34613062-vmss000000  Ready    agent      20m    v1.18.14  
aks-gpupool-34613062-vmss000001  Ready    agent      20m    v1.18.14  
aks-gpupool-34613062-vmss000002  Ready    agent      20m    v1.18.14  
verronmartina@verron-mac-0 ~ %
```

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.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo, a search bar, and user account information. Below the navigation bar, the 'Virtual networks' blade is open. The title 'Virtual networks' is at the top left, with a back arrow and a close button. Underneath, it says 'seanlucelive (Default Directory)'. The main content area displays a table of virtual network resources. The columns are: Name, Resource group, Location, and Subscription. A single row is selected, showing 'aks-vnet-22885919' under 'Name', 'MC_sluce:rg_TridentDemo_eastus2' under 'Resource group', 'East US 2' under 'Location', and 'AzureSub01' under 'Subscription'. The entire row is highlighted with a red box. At the bottom of the table, there are sorting options ('Name ↑↓', 'Resource group ↑↓', 'Location ↑↓', 'Subscription ↑↓') and grouping/filtering options ('No grouping', 'List view').

2. Click Subnets and select +Subnet from the top toolbar.

The screenshot shows the Microsoft Azure portal interface, specifically the 'Subnets' blade for the virtual network 'aks-vnet-22885919'. The top navigation bar includes the Microsoft Azure logo, a search bar, and user account information. The title 'aks-vnet-22885919 | Subnets' is at the top left, with a back arrow and a close button. Below the title, it says 'Virtual network'. The main content area displays a table of subnets. The columns are: Name, IPv4, IPv6 (many available), Delegated to, and Security group. One subnet, 'aks-subnet', is listed with the IP range '10.240.0.0/16 (65530 available)'. The 'Delegated to' and 'Security group' columns show '-' and 'aks-agentpool-2288591...' respectively. At the top of the table, there is a search bar labeled 'Search subnets'. Above the table, the toolbar includes a search bar, a 'Subnet' button (highlighted with a red box), a 'Gateway subnet' button, a 'Refresh' button, a 'Manage users' button, and a 'Delete' button. On the left side, there is a sidebar with various navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Address space, Connected devices, and Subnets (which is also highlighted with a red box). The 'Subnets' link in the sidebar is the active tab.

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.

Add subnet

X

Name *

ANF.sn



Subnet address range * ⓘ

10.0.0.0/24

10.0.0.0 - 10.0.0.255 (251 + 5 Azure reserved addresses)

Add IPv6 address space ⓘ

NAT gateway ⓘ

None



Network security group

None



Route table

None



SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. [Learn more](#)

Services ⓘ

0 selected



SUBNET DELEGATION

Delegate subnet to a service ⓘ

Microsoft.Netapp/volumes



OK

Cancel

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:

1. [Registered for Azure NetApp Files and NetApp Resource Provider \(through the Azure Cloud Shell\)](#).
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:

Field	Value or description
Peering link name	aks-vnet-name_to_anf
SubscriptionID	Subscription of the Azure NetApp Files VNet to which you're peering
VNet peering partner	Azure NetApp Files VNet



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.

```
wget  
https://github.com/NetApp/trident/releases/download/v21.01.1/trident-  
installer-21.01.1.tar.gz  
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`.

```
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.

```
cd ./lane-detection-SCNN-horovod/trident-config
```

4. Create an Azure Service Principle (the service principle is how Trident communicates with Azure to access your Azure NetApp Files resources).

```
az ad sp create-for-rbac --name
```

The output should look like the following example:

```
{  
    "appId": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
    "displayName": "netapprtrident",  
    "name": "http://netapprtrident",  
    "password": "xxxxxxxxxxxxxx.xxxxxxxxxxxxxx",  
    "tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
}
```

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.

Field	Value
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)

The file should look like the following example:

```
{
  "version": 1,
  "storageDriverName": "azure-netapp-files",
  "subscriptionID": "fakec765-4774-fake-ae98-a721add4fake",
  "tenantID": "fakef836-edc1-fake-bff9-b2d865eefake",
  "clientID": "fake0f63-bf8e-fake-8076-8de91e57fake",
  "clientSecret": "SECRET",
  "location": "westeurope",
  "serviceLevel": "Standard",
  "virtualNetwork": "anf-vnet",
  "subnet": "default",
  "nfsMountOptions": "vers=3,proto=tcp",
  "limitVolumeSize": "500Gi",
  "defaults": {
    "exportRule": "0.0.0.0/0",
    "size": "200Gi"
  }
}
```

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. K8 users provision volumes by using PVCs that specify a storage class by name. Instruct K8s to create a storage class `azurenetaffiles` 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 azurenetaffiles
```

The output should look like the following example:

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	AGE
azurenetaffiles	csi.trident.netapp.io	Delete	Immediate	false	98s

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:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml  
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

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

RUN:AI installation

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.

New Project

Basics

Node Affinity

Time Limit

Project Name

Assigned GPUs

Over-quota for project Allow over-quota

Save Cancel

3. [Install RUN:AI CLI](#).
4. On your terminal, set lane-detection as a default RUN: AI project by using the following command:

```
`runai config project lane-detection`
```

The output should look like the following example:

```
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:

NAME	STATUS	AGE
default	Active	130m
kube-node-lease	Active	130m
kube-public	Active	130m
kube-system	Active	130m
runai	Active	4m44s
runai-lane-detection	Active	13s
trident	Active	102m

6. Create ClusterRole netappssnapshot and ClusterRoleBinding netappssnapshot 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

TAG (such as download-tusimple and 1.0):

```
#!/bin/bash
#
# A simple script to build the Docker image.
#
# $ build_image.sh
set -ex

IMAGE=muneer7589/download-tusimple
TAG=1.0

# Build image
echo "Building image: "$IMAGE
docker build . -f Dockerfile \
--tag "${IMAGE}:${TAG}"
echo "Finished building image: "$IMAGE

# Push image
echo "Pushing image: "$IMAGE
docker push "${IMAGE}:${TAG}"
echo "Finished pushing image: "$IMAGE
```

- 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 azurenetaffiles:100Gi:/mnt
--image muneer7589/download-tusimple:1.0
```

- b. Enter the information from the table below to submit the RUN:AI job:

Field	Value or description
-name	Name of the job
-pvc	PVC of the format [StorageClassName]:Size:ContainerMountPath In the above job submission, you are creating an PVC based on-demand using Trident with storage class azurenetaffiles. Persistent volume capacity here is 100Gi and it's mounted at path /mnt.
-image	Docker image to use when creating the container for this job

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.

```
runai list jobs
```

```
Showing jobs for project lane-detection
NAME          STATUS      AGE     NODE           IMAGE          TYPE    PROJECT      USER        GPUs Allocated (Requested)
PODs Running (Pending) SERVICE URL(S)
download-tusimple-data  ContainerCreating  1m   aks-agentpool-34613062-vmss00000a  muneer7589/download-tusimple:1.0  Train  lane-detection  veronnarmartina  0 (0)
1 (@)
```

- d. Check the submitted job logs.

```
runai logs download-tusimple-data -t 10
```

```
751150K ..... 6% 16.2M 20m37s
751200K ..... 6% 11.1M 20m37s
751250K ..... 6% 12.5M 20m36s
751300K ..... 6% 11.3M 20m36s
751350K ..... 6% 15.2M 20m36s
751400K ..... 6% 10.5M 20m36s
751450K ..... 6% 15.2M 20m36s
751500K ..... 6% 14.1M 20m36s
751550K ..... 6% 24.3M 20m36s
751600K ..... 6% 26.3M 20m36s
```

- 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-bb03b74d-2c17-40c4-a445-79f3de8d16d5    100Gi    RWO    azurenetsappfiles    4m47s
```

- f. Check the job in RUN: AI UI (or app.run.ai).

Job Name	Status	User	Project	Total Run Time	Creation Time	Type	GPU Utilization	Used CPU
download-tusimple-data	Running	vernonma...	lane-detection	00:07:11	03/03/21, 2:51PM	Train	-	0.00
build1	Deleted	root	lane-detection	00:01:56	03/01/21, 10:18...	Interactive	-	-
download-tusimple-data	Deleted	root	lane-detection	-	03/01/21, 9:58AM	Train	-	-
download-tusimple-data	Deleted	root	lane-detection	-	03/01/21, 10:03...	Train	-	-
download-tusimple-data	Deleted	root	lane-detection	00:02:55	03/01/21, 10:24...	Train	-	-
download-tusimple-data	Deleted	root	lane-detection	-	03/01/21, 10:30...	Train	-	-
download-tusimple-data	Deleted	root	lane-detection	00:13:17	03/01/21, 11:41...	Train	-	-
download-tusimple-data-1	Deleted	vernonma...	lane-detection	-	02/26/21, 5:30PM	Train	-	-

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).

```

#!/bin/bash
#
# A simple script to build the distributed Docker image.
#
# $ build_image.sh
set -ex

IMAGE=muneer7589/dist-lane-detection
TAG=3.0

# Build image
echo "Building image: "$IMAGE
docker build . -f Dockerfile \
--tag "${IMAGE}:${TAG}"
echo "Finished building image: "$IMAGE

# Push image
echo "Pushing image: "$IMAGE
docker push "${IMAGE}:${TAG}"
echo "Finished pushing image: "$IMAGE

```

- d. Run the script to build the docker image and push to the docker repository.

```

chmod +x build_image.sh
./build_image.sh

```

2. Submit the RUN: AI job for carrying out distributed training (MPI):

- 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.
- First, get the volume name of the PVC by running the following command:

```

kubectl get pvc | grep download-tusimple-data

```

```

root@ai-w-gpu-2:/mnt/ai_data/anf_runai/lane-detection-SCNN-horovod# kubectl get pvc | grep download-tusimple-data
pvc-download-tusimple-data-0 Bound pvc-bb03b74d-2c17-40c4-a445-79f3de8d16d5 100Gi RWX azurenetaffiles 2d4h

```

- Patch the volume and update access mode to ReadWriteMany (replace volume name with yours in the following command):

```

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"
```

Field	Value or description
name	Name of the distributed training job
large shm	Mount a large /dev/shm device It is a shared file system mounted on RAM and provides large enough shared memory for multiple CPU workers to process and load batches into CPU RAM.
processes	Number of distributed training processes
gpu	Number of GPUs/processes to allocate for the job In this job, there are three GPU worker processes (--processes=3), each allocated with a single GPU (--gpu 1)
pvc	Use existing persistent volume (pvc-download-tusimple-data-0) created by previous job (download-tusimple-data) and it is mounted at path /mnt
image	Docker image to use when creating the container for this job
Define environment variables to be set in the container	
USE_WORKERS	Setting the argument to true turns on multi-process data loading
NUM_WORKERS	Number of data loader worker processes
BATCH_SIZE	Training batch size

Field	Value or description
USE_VAL	Setting the argument to true allows validation
VAL_BATCH_SIZE	Validation batch size
ENABLE_SNAPSHOT	Setting the argument to true enables taking data and trained model snapshots for ML versioning purposes
PVC_NAME	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

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.

```
runai list jobs
```

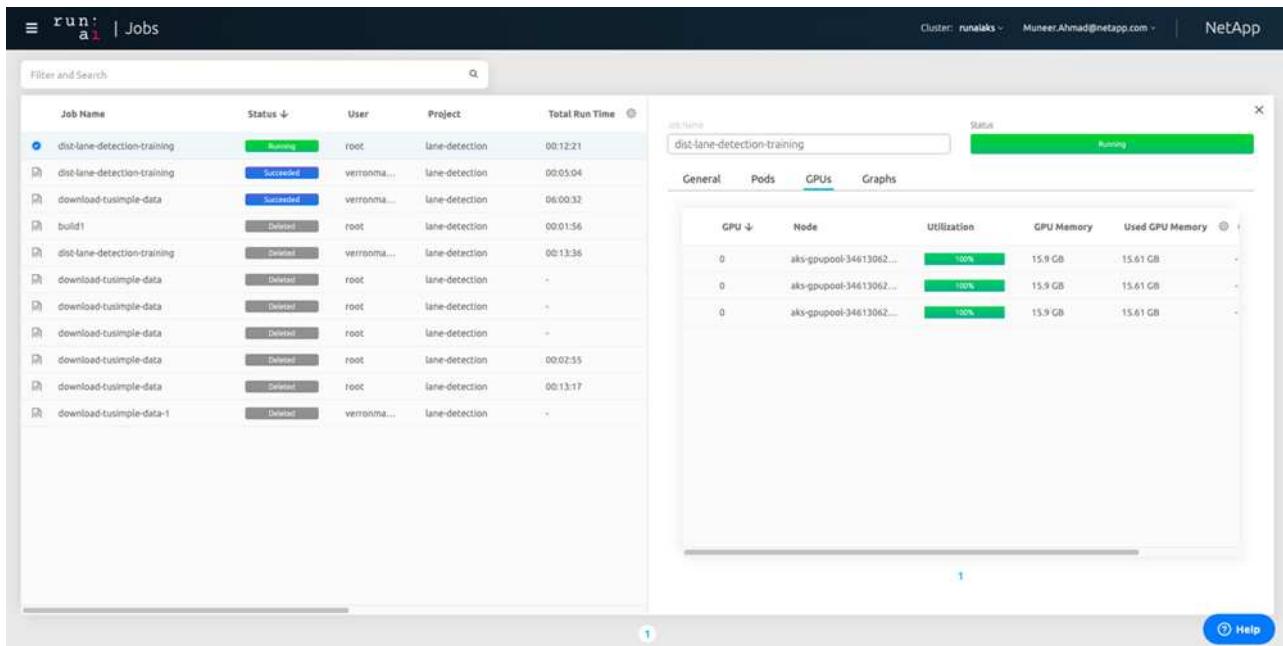
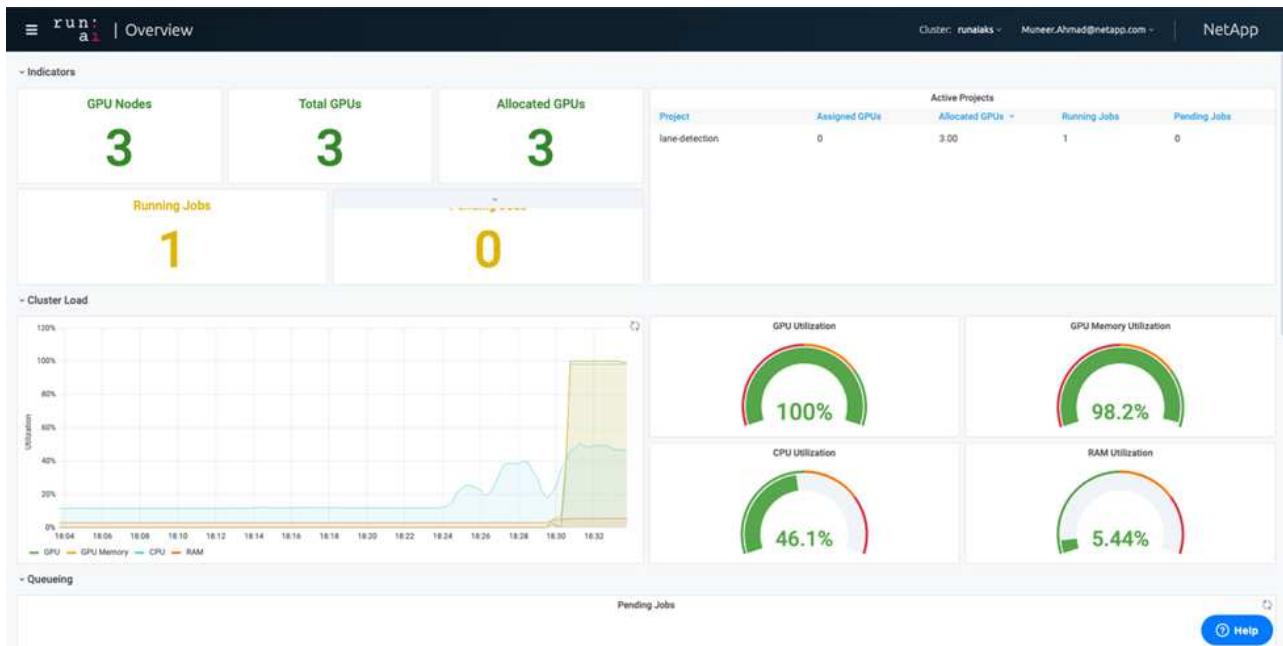
NAME	STATUS	AGE	NODE	IMAGE	TYPE	PROJECT	USER	GPUs Allocated (Requested)	PODs
download-tusimple-data	Succeeded	1d		muneeer7589/download-tusimple:1.0	Train	lane-detection	vernonmartina	- (0)	0 (0)
dist-lane-detection-training	Init:0/1	2m	<multiple>	muneeer7589/dist-lane-detection:3.1	Train	lane-detection	root	3 (3)	4 (0)

- f. Submitted job logs:

```
runai logs dist-lane-detection-training
```

```
root@ai-w-gpu-2:~/runai# runai logs dist-lane-detection-training
Running with 3 workers
2021-03-04 17:29:23.158449: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.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
```

- 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:



- 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
```

```
[1,0]<stdout>:Snapshot snap-pvc-download-tusimple-data-0-dist-lane-detection-training-launcher-2021-03-05-16-23-42 created in namespace runai-lane-detection
```

```
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.

```
cd ~
```

2. Go to the project directory lane-detection-SCNN-horovod.

```
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`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restored-tusimple
spec:
  storageClassName: azurenetappfiles
  dataSource:
    name: snap-pvc-download-tusimple-data-0-dist-lane-detection-training-launcher-2021-03-05-16-23-42
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
```

4. Create a new PVC by using `restore-snapshot-pvc.yaml`.

```
kubectl create -f restore-snapshot-pvc.yaml
```

The output should look like the following example:

```
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.

NFSv3	Address	Service Level	pool0	...
NFSv4.1	10.28.254.4:/norootfor...	Standard		
NFSv4.1	NAS-735a.docs.lab:/for...	Premium		
NFSv3	NAS-735a.docs.lab:/krt...	Premium		
NFSv3	10.28.254.4:/moveme0	Premium		
NFSv3	10.28.254.4:/placeholder	Premium		

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](#).

```
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](#).

```

Set-AzNetAppFilesVolumePool
-ResourceGroupName "MyRG"
-AccountName "MyAnfAccount"
-PoolName "MyAnfPool"
-Name "MyAnfVolume"
-NewPoolResourceId 7d6e4069-6c78-6c61-7bf6-c60968e45fbf

```

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

<https://docs.run.ai/Administrator/Cluster-Setup/cluster-install/#step-3-install-runai>

<https://docs.run.ai/Administrator/Researcher-Setup/cli-install/#runai-cli-installation>

- Submitting jobs in RUN: AI CLI

<https://docs.run.ai/Researcher/cli-reference/runai-submit/>

<https://docs.run.ai/Researcher/cli-reference/runai-submit-mpi/>

- Azure Cloud resources: Azure NetApp Files

<https://docs.microsoft.com/azure/azure-netapp-files/>

- Azure Kubernetes Service

<https://azure.microsoft.com/services/kubernetes-service/-features>

- Azure VM SKUs

<https://azure.microsoft.com/services/virtual-machines/>

- Azure VM with GPU SKUs

<https://docs.microsoft.com/azure/virtual-machines/sizes-gpu>

- NetApp Trident

<https://github.com/NetApp/trident/releases>

- Data Fabric powered by NetApp

<https://www.netapp.com/data-fabric/what-is-data-fabric/>

- NetApp Product Documentation

<https://www.netapp.com/support-and-training/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 straight forward 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.

[Next: Use Case Overview and Problem Statement](#)

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.

[Next: Solution Overview](#)

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
- Monitoringcnvrg.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.

ONTAP		Datasets		6 Datasets	290.2 MB	3 NFS Connected	1.07/360.77 GB Used by cached commits
	@Padmas						
		fraud_jun	fraud_may	consumer_img1	public_set12		
		M V D J	S V	J	M V D J		
		2 Cached Commits	0 Cached Commits	0 Cached Commits	1 Cached Commits		
		32.4 MB	57.6 MB	27.3 MB	102.7 MB		
		4 commits	2 commits	4 commits	7 commits		
		Active 8 hours ago	Active 3 hours ago	Active 4 days ago	Active 4 hours ago		
		fraud_sim_base	misc_base1				
		J V	M D J				
		1 Cached Commits	0 Cached Commits				
		45.1 MB	25.1 MB				
		2 commits	1 commits				
		Active 24 days ago	Active 2 days ago				
cnvrg.io							

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.



[Next: Concepts and Components](#)

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 therefore 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 is 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.

[Next: Hardware and Software Requirements](#)

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.

Component	Version
Ubuntu	18.04.4 LTS
NVIDIA DGX OS	4.4.0
NVIDIA DeepOps	20.02.1
Kubernetes	1.15
Helm	3.1.0
cnvrg.io	3.0.0
NetApp ONTAP	9.6P4

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.

[Next: Solution Deployment and Validation Details](#)

Solution Deployment and Validation Details

The following sections discuss the details of solution deployment and validation.

[Next: ONTAP AI Deployment](#)

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.

[Next: Kubernetes Deployment](#)

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.

[Next: Cnvrq.io Deployment](#)

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:

NAME	READY	STATUS	RESTARTS	AGE
cnvrg-app-69fbb9df98-6xrgf	1/1	Running	0	2m
cnvrg-sidekiq-b9d54d889-5x4fc	1/1	Running	0	2m
controller-65895b47d4-s96v6	1/1	Running	0	2m
init-app-vs-config-wv9c4	0/1	Completed	0	9m
init-gateway-vs-config-2zbpp	0/1	Completed	0	9m
init-minio-vs-config-cd2rg	0/1	Completed	0	9m
minio-0	1/1	Running	0	2m
postgres-0	1/1	Running	0	2m
redis-695c49c986-kcbt9	1/1	Running	0	2m
seeder-wh655	0/1	Completed	0	2m
speaker-5sghr	1/1	Running	0	2m

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.

X-ray train (ResNet50)
by yochz

Status: Success Duration: 33m 54s

Input: python3 resnet50.py --data /data/x-ray-sample-splitted --data_test None --output_model model.h5 --val_size 0.2

Start Time: 22-Mar-2020, 3:55:37 PM **End Time:** 22-Mar-2020, 4:29:22 PM **Duration:** 33m 45s **Compute:** gpu-real **Image:** tensorflow:20.01-tf2-py3

Start Commit: e0854e73 **End Commit:** a980dd8e

CPU **Memory** **Block IO** **GPU** **GPU Memory**

Classes list: ["No Finding", "Hemato", "Fibrosis", "Pleural_Thickening", "Mass", "Infiltration", "Effusion", "Cardiomegaly", "Atelectasis", "Edema", "Consolidation", "Touch Bar Shot 2020-03-12 at 7.53.13 PM.png", "Pneumonia", "Pneumothorax", "Nodule", "Emphysema"]

Model: resnet50 **GPU Found:** 1 **tensorflow local version:** 2.0.0

GridSearch_ID: 2451r **output_layer_activation:** softmax **hidden_layer_activation:** relu **pooling_height:** 2
pooling_width: 2 **conv_height:** 3 **conv_width:** 3 **image_height:** 224
image_width: 224 **optimizer:** adam **dropout:** 0.3 **image_color:** rgb
batch_size: 1024 **steps_per_epoch:** 10 **epoches:** 10 **val_size:** 0.2
output_model: model.h5 **data_test:** None

loss

Epoch	Experiment 50 Loss
0	2.35
1	1.85
2	1.75
3	1.70
4	1.68
5	1.67
6	1.66
7	1.65
8	1.64
9	1.63
10	1.62
11	1.61



Next: Conclusion

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.

Next: [Acknowledgments](#)

Acknowledgments

- Mike Oglesby, Technical Marketing Engineer, NetApp
- Santosh Rao, Senior Technical Director, NetApp

Next: [Where to Find Additional Information](#)

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>):
 - Cnvrg CORE (free ML platform)
<https://cnvrg.io/platform/core>
 - Cnvrg docs
<https://app.cnvrg.io/docs>
- NVIDIA DGX-1 servers:
 - NVIDIA DGX-1 servers
<https://www.nvidia.com/en-us/data-center/dgx-1/>
 - NVIDIA Tesla V100 Tensor Core GPU
<https://www.nvidia.com/en-us/data-center/tesla-v100/>
 - NVIDIA GPU Cloud (NGC)
<https://www.nvidia.com/en-us/gpu-cloud/>
- NetApp AFF systems:
 - AFF datasheet
<https://www.netapp.com/us/media/d-3582.pdf>
 - NetApp FlashAdvantage for AFF
<https://www.netapp.com/us/media/ds-3733.pdf>
 - ONTAP 9.x documentation

<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>

- NetApp FlexGroup technical report

<https://www.netapp.com/us/media/tr-4557.pdf>

- 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

<https://www.cisco.com/c/en/us/products/switches/nexus-3232c-switch/index.html>

- Mellanox Spectrum 2000 series switches

http://www.mellanox.com/page/products_dyn?product_family=251&mtag=sn2000

- ML framework and tools:

- DALI

<https://github.com/NVIDIA/DALI>

- TensorFlow: An Open-Source Machine Learning Framework for Everyone

<https://www.tensorflow.org/>

- Horovod: Uber's Open-Source Distributed Deep Learning Framework for TensorFlow

<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
<http://www.kubeflow.org/>
- Jupyter Notebook Server
<http://www.jupyter.org/>
- Dataset and benchmarks:
 - NIH chest X-ray dataset
<https://nihcc.app.box.com/v/ChestXray-NIHCC>
 - Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadadi Bagheri, Ronald Summers, ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases, IEEE CVPR, pp. 3462-3471, 2017TR-4841-0620

AI Inferencing at the Edge - NetApp with Lenovo ThinkSystem - Solution Design

TR-4886: AI Inferencing at the Edge - NetApp with Lenovo ThinkSystem - Solution Design

Sathish Thyagarajan, NetApp
 Miroslav Hodak, Lenovo

Summary

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.





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.



red	blue
Lenovo compute system	NetApp AFF storage system
Edge devices performing inference on inputs from cameras, sensors, and so on.	Shared storage holding inference models and data from edge devices for later analysis.

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.

[Next: Technology overview.](#)

Technology overview

[Previous: Introduction.](#)

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 Cloud Sync

[Cloud 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, Cloud 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. Cloud Sync continuously synchronizes the data, based on your predefined schedule, moving only the deltas, so time and money spent on data replication is minimized. Cloud Sync is a software as a service (SaaS) tool that is extremely simple to set up and use. Data transfers that are triggered by Cloud Sync are carried out by data brokers. You can deploy Cloud 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:

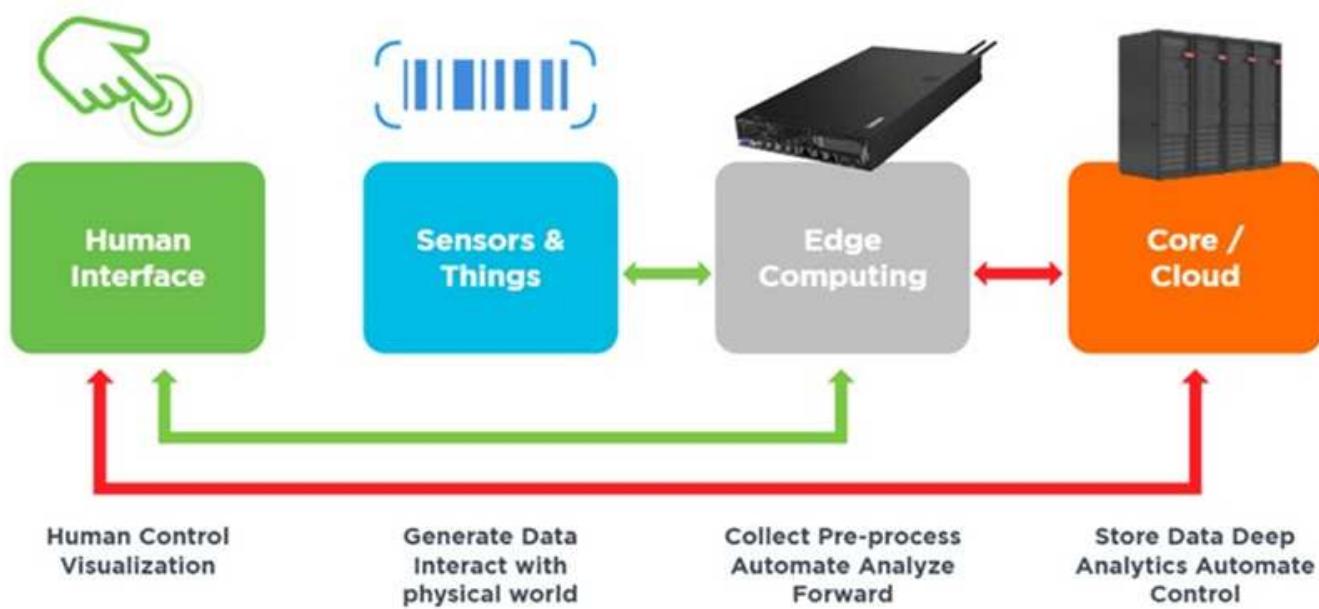
- Highly scalable, modular designs to grow with your business
- Industry-leading resilience to save hours of costly unscheduled downtime
- Fast flash technologies for lower latencies, quicker response times, and smarter data management in real time

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/> in the “Edge, Closed Division” section in entry #0.7-145.

[Next: Test plan.](#)

Test plan

[Previous: Technology overview.](#)

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.

Area	Task	Model	Dataset	QSL size	Quality	Multistream latency constraint
Vision	Image classification	Resnet50v1.5	ImageNet (224x224)	1024	99% of FP32	50ms
Vision	Object detection (large)	SSD-ResNet34	COCO (1200x1200)	64	99% of FP32	66ms
Vision	Object detection (small)	SSD-MobileNetsv1	COCO (300x300)	256	99% of FP32	50ms
Vision	Medical image segmentation	3D UNET	BraTS 2019 (224x224x160)	16	99% and 99.9% of FP32	n/a
Speech	Speech-to-text	RNN	Librispeech dev-clean	2513	99% of FP32	n/a
Language	Language processing	BERT	SQuAD v1.1	10833	99% of FP32	n/a

The following table presents Edge benchmark scenarios.

Area	Task	Scenarios
Vision	Image classification	Single stream, offline, multistream

Area	Task	Scenarios
Vision	Object detection (large)	Single stream, offline, multistream
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.

[Next: Test configuration.](#)

Test configuration

[Previous: Test plan.](#)

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.

Solution components	Details
Lenovo ThinkSystem servers	<ul style="list-style-type: none"> • 2x SE350 servers each with one NVIDIA T4 GPU card
	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • 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.

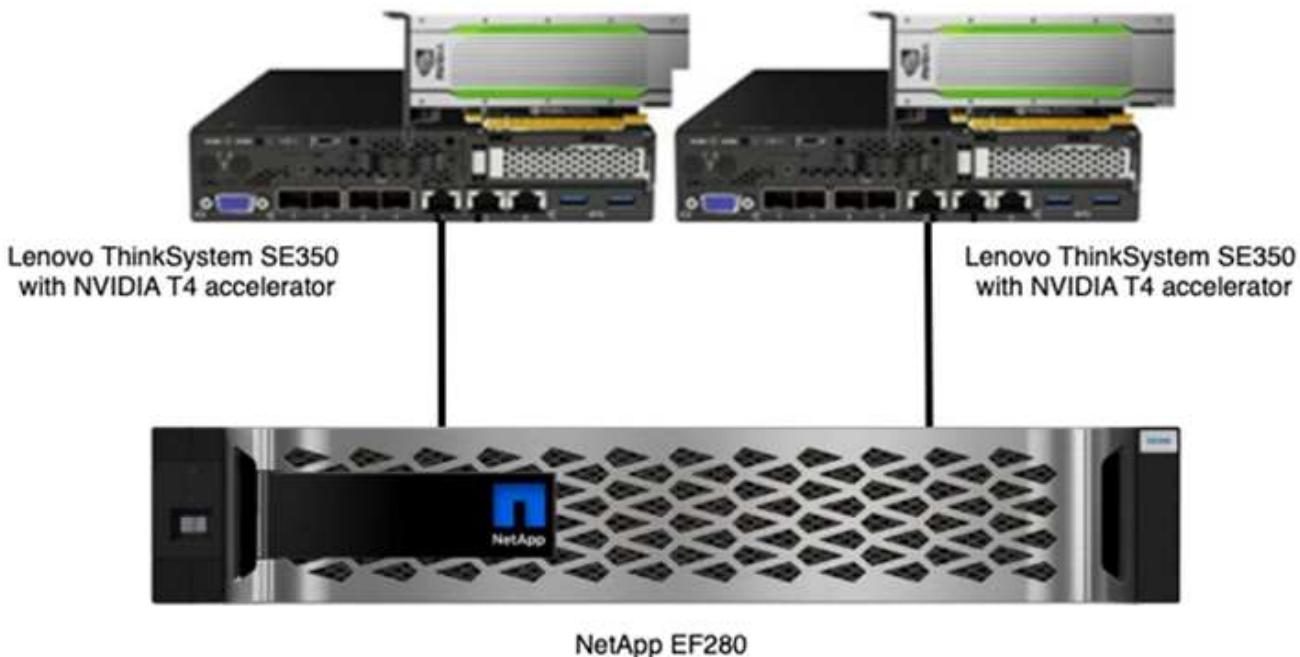
Controller	Aggregate	FlexGroup volume	Aggregatesize	Volumesize	Operating systemmount point
Controller1	Aggr1	/netapplenovo_AI_fg	8.42TiB	15TB	/netapp_lenovo_fg
Controller2	Aggr2		8.42TiB		

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.

Controller	Volume Group	Volume	Volumesize	DDPsize	Connection method
Controller1	DDP1	Volume 1	8.42TiB	16TB	SE350-1 to iSCSI LUN 0
Controller2		Volume 2	8.42TiB		SE350-2 to iSCSI LUN 1



[Next: Test procedure.](#)

Test procedure

[Previous: Test configuration.](#)

We used the following test procedure in this validation.

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

[Next: Test results.](#)

Test results

[Previous: Test procedure.](#)

Test results for AFF

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.

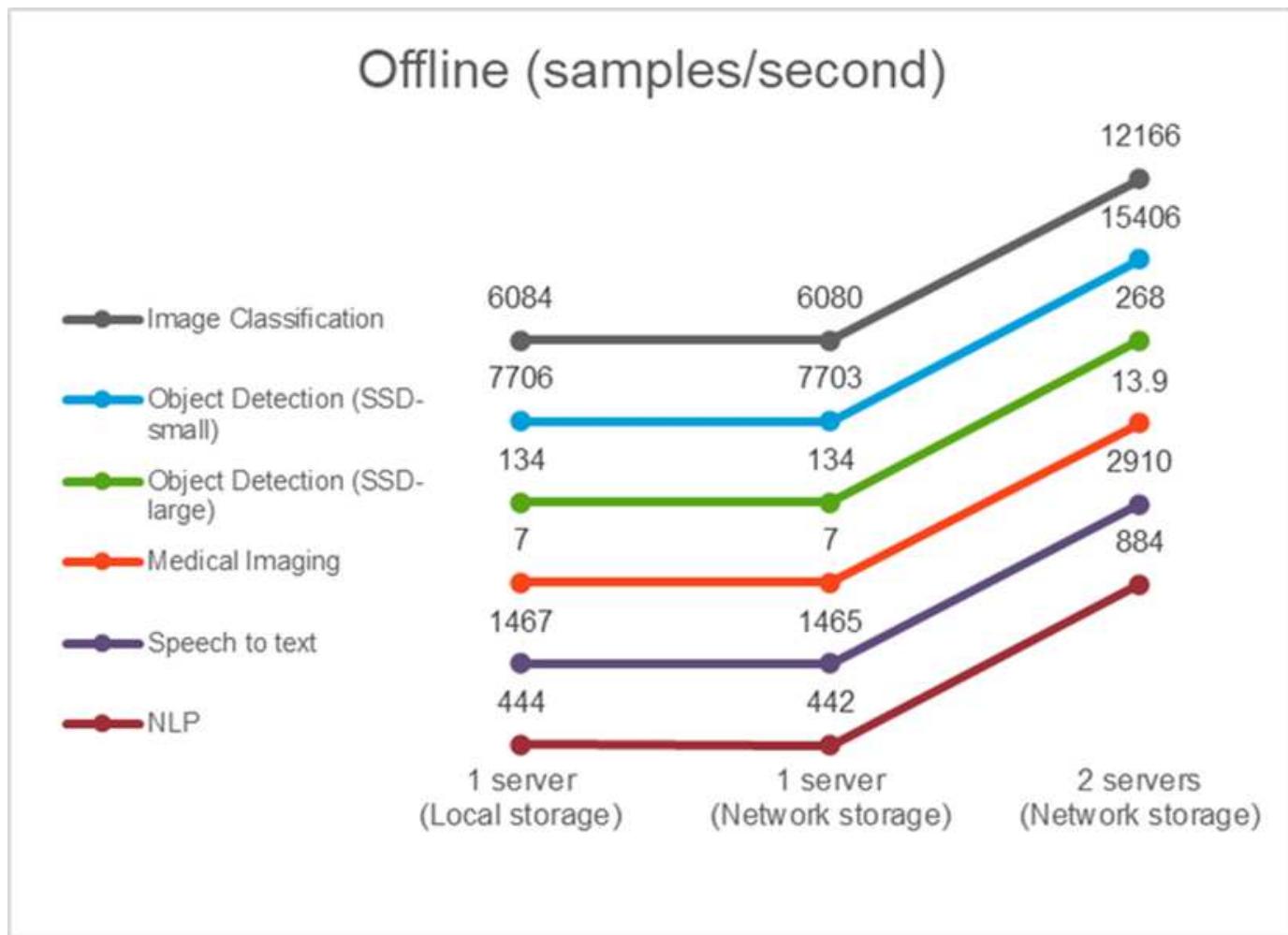
This gives 15 possible workloads, which were all tested under three different setups:

- 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.

[Next: Architecture sizing options.](#)

Architecture sizing options

[Previous: Test results.](#)

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.

[Next: Conclusion.](#)

Conclusion

[Previous: Architecture sizing options.](#)

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
<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>
- TR-4727: NetApp EF-Series Introduction

<https://www.netapp.com/pdf.html?item=/media/17179-tr4727pdf.pdf>

- NetApp E-Series SANtricity Software Datasheet

<https://www.netapp.com/pdf.html?item=/media/19775-ds-3171-66862.pdf>

- NetApp Persistent Storage for Containers—NetApp Trident

<https://netapp.io/persistent-storage-provisioner-for-kubernetes/>

- MLPerf

- <https://mlcommons.org/en/>
- <http://www.image-net.org/>
- <https://mlcommons.org/en/news/mlperf-inference-v11/>

- NetApp Cloud Sync

https://docs.netapp.com/us-en/occm/concept_cloud_sync.html#how-cloud-sync-works

- TensorFlow benchmark

<https://github.com/tensorflow/benchmarks>

- Lenovo ThinkSystem SE350 Edge Server

<https://lenovopress.com/lp1168>

- Lenovo ThinkSystem DM5100F Unified Flash Storage Array

<https://lenovopress.com/lp1365-thinksystem-dm5100f-unified-flash-storage-array> [<https://lenovopress.com/lp1365-thinksystem-dm5100f-unified-flash-storage-array>]

Version history

Version	Date	Document version history
Version 1.0	March 2021	Initial release
Version 2.0	October 2021	Updated with EF and MLPerf Inference v1.1

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

[Next: Solution Overview](#)

Solution Overview

NetApp ONTAP AI and Cloud 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 pointsNetApp 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 Cloud 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, Cloud 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 Cloud 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](#).

Next: Solution Technology

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 (AlaaS) 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 AlaaS 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.

Hardware	Quantity
T4 or V100 GPU	1
NVIDIA DGX Station	1

Software Requirements

The following table lists the software components that are required to implement the solution as tested.

Software	Version or Other Information
NetApp ONTAP data management software	9.6
Cisco NX-OS switch firmware	7.0(3)I6(1)
NVIDIA DGX OS	4.0.4 - Ubuntu 18.04 LTS
NVIDIA Jarvis Framework	EA v0.2
NVIDIA NeMo	nvcr.io/nvidia/nemo:v0.10
Docker container platform	18.06.1-ce [e68fc7a]

[Next: Build a Virtual Assistant Using Jarvis, Cloud Sync, and NeMo Overview](#)

Overview

This section provides detail on the implementation of the virtual retail assistant.

[Next: Jarvis Deployment](#)

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`.

[Next: Customize States and Flows for Retail Use Case](#)

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.

See 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`

```
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
end_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:
      text: '{{retail_status}}'
    transitions:
      next_state: input_intent
  ask_retail_location:
    type: message_text
    properties:
      text: "For which location? I can find the closest store near you."
    transitions:
      next_state: input_retail_location
  input_retail_location:
    type: input_user
    properties:
      nlp_type: jarvis
      entities:
        slot: location
        require_match: true
    transitions:
      match: retail_state
      notmatch: check_retail_jarvis_error
  output_retail_acknowledge:
    type: message_text_random
    properties:
      responses:
        - 'ok in {{location}}'
        - 'the store in {{location}}'
        - 'I always wanted to shop in {{location}}'
    delay: 0
```

```

transitions:
  next_state: retail_state
output_retail_notlocation:
  type: message_text
  properties:
    text: "I did not understand the location. Can you please repeat?"
  transitions:
    next_state: input_intent
check_rerail_jarvis_error:
  type: conditional_exists
  properties:
    key: '{{jarvis_error}}'
  transitions:
    exists: show_retail_jarvis_api_error
    notexists: output_retail_notlocation
show_retail_jarvis_api_error:
  type: message_text
  properties:
    text: "I am having trouble understanding right now. Come again on
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

```

```
name: error_flow
states:
  error_state:
    type: message_text_random
    properties:
      responses:
        - "Sorry I didn't get that!"
        - "Are we talking about retail or weather? What would you like to know?"
        - "Sorry I know only about retail information or the weather"
        - "You can ask me about retail information or the weather, the rainfall, the temperature. I don't know much more"
        - "Let's talk about retail or the weather!"
    delay: 0
    transitions:
      next_state: input_intent
```

[Next: Connect to Third-Party APIs as Fulfillment Engine](#)

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.

[Next: NetApp Retail Assistant Demonstration](#)

NetApp Retail Assistant Demonstration

We recorded a demonstration video of NetApp Retail Assistant (NARA). Click [this link](#) to open the following figure and play the video demonstration.

NetApp NARA



Hi, welcome to NARA retail and weather service. How can I help you?

Write your message...

Submit

System replied. Waiting for user input.

Unmute System Speech

Next: Use NetApp Cloud Sync to Archive Conversation History

Use NetApp Cloud Sync to Archive Conversation History

By dumping conversation history into a CSV file once a day, we can then leverage Cloud 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 Cloud Sync to send conversation history for NeMo training. Details of NeMo training can be found in the section [Expand Intent Models Using NeMo Training](#).



Next: Expand Intent Models Using NeMo Training

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  
B-location  
B-cost.high
```

```
B-cost.average  
B-cost.low  
B-time.period_of_time  
B-rating.high  
B-rating.average  
B-rating.low  
B-interrogative.location  
B-interrogative.manner  
B-interrogative.time  
B-interrogative.personal  
B-interrogative  
B-verb  
B-article  
I-store.type  
I-store.name  
I-store.status  
I-store.hour.start  
I-store.hour.end  
I-store.hour.day  
I-item.type  
I-item.name  
I-item.color  
I-item.size  
I-item.quantity  
I-location  
I-cost.high  
I-cost.average  
I-cost.low  
I-time.period_of_time  
I-rating.high  
I-rating.average  
I-rating.low  
I-interrogative.location  
I-interrogative.manner  
I-interrogative.time  
I-interrogative.personal  
I-interrogative  
I-verb  
I-article  
O
```

train.tsv

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:v0.1.0
```

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:v0.1.0
```

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.

```
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
```

[Next: Conclusion](#)

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 Cloud Sync and training NeMo models on new data.

[Next: Acknowledgments](#)

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.

Next: [Where to Find Additional Information](#)

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
<https://www.nvidia.com/en-us/data-center/dgx-station/>
 - NVIDIA V100 Tensor Core GPU
<https://www.nvidia.com/en-us/data-center/tesla-v100/>
 - NVIDIA NGC
<https://www.nvidia.com/en-us/gpu-cloud/>
- 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
<https://www.netapp.com/us/media/ds-3582.pdf>
 - NetApp Flash Advantage for All Flash FAS
<https://www.netapp.com/us/media/ds-3733.pdf>
 - ONTAP 9 Information Library
<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>
 - NetApp ONTAP FlexGroup Volumes technical report
<https://www.netapp.com/us/media/tr-4557.pdf>

- NetApp ONTAP AI
 - ONTAP AI with DGX-1 and Cisco Networking Design Guide
<https://www.netapp.com/us/media/nva-1121-design.pdf>
 - ONTAP AI with DGX-1 and Cisco Networking Deployment Guide
<https://www.netapp.com/us/media/nva-1121-deploy.pdf>
 - ONTAP AI with DGX-1 and Mellanox Networking Design Guide
<http://www.netapp.com/us/media/nva-1138-design.pdf>
 - ONTAP AI with DGX-2 Design Guide
<https://www.netapp.com/us/media/nva-1135-design.pdf>

TR-4858: NetApp Orchestration Solution with Run:AI

Rick Huang, David Arnette, Sung-Han Lin, NetApp
 Yaron Goldberg, Run:AI

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

[Next: Solution Overview](#)

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.

[Next: Solution Technology](#)

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.



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.

Hardware	Quantity
DGX-1 systems	2
AFF A800	1
Nexus 3232C switches	2

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	Version or Other Information
NetApp ONTAP data management software	9.6p4
Cisco NX-OS switch firmware	7.0(3)I6(1)
NVIDIA DGX OS	4.0.4 - Ubuntu 18.04 LTS

Software	Version or Other Information
Kubernetes version	1.17
Trident version	20.04.0
Run:AI CLI	v2.1.13
Run:AI Orchestration Kubernetes Operator version	1.0.39
Docker container platform	18.06.1-ce [e68fc7a]

Additional software requirements for Run:AI can be found at [Run:AI GPU cluster prerequisites](#).

[Next: Optimal Cluster and GPU Utilization with Run AI](#)

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.

[Next: Run AI Installation.](#)

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 on an on-premise Kubernetes cluster](#) and [Installing the Run:AI CLI](#).

Next: Run AI Dashboards and Views

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.

Next: Creating Projects for Data Science Teams and Allocating GPUs

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 build 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.

Project Name	Assigned GPUs	Created	Training Node Affinity	Interactive Node Affinity
team-a	2	07/27/20, 9:28AM	none	none
team-b	4	07/28/20, 7:50AM	none	none
team-c	2	07/28/20, 7:50AM	none	none
team-d	8	07/28/20, 7:51AM	none	none

[Next: Submitting Jobs in Run AI CLI](#)

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:

```
$ 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:

```
$ 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.

```
You can also run get hyper1 -p team-a to check the job status
~> runai list
Showing jobs for project team-a
NAME      STATUS    AGE     NODE          IMAGE
hyper1    Running   11s   gke-dev-yaron1-gpu-4-pool-154f511d-5nk5  gcr.io/run-ai-demo/quickstart
TYPE      PROJECT  USER    GPUs
Train     team-a   yaron   1
```

To get an additional status on your job, run the following command:

```
$ runai get hyper1
```

To view the logs of the job, run the `runai logs <job-name>` command:

```
$ 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/>. 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](#).

Next: Achieving High Cluster Utilization

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.



[Next: Fractional GPU Allocation for Less Demanding or Interactive Workloads](#)

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.

```

root@run-deploy:~# runai bash frac05 -p team-d
root@frac05-0:/workload# nvidia-smi
Tue Jul 28 15:17:03 2020
+-----+
| NVIDIA-SMI 450.51.05    Driver Version: 450.51.05    CUDA Version: 11.0    |
|-----+-----+-----+
| GPU  Name      Persistence-MI Bus-Id      Disp.A  Volatile Uncorr. ECC  |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M.  |
| |          |          |          |          |          |          MIG M.  |
|-----+-----+-----+-----+-----+-----+-----+
| 0  Tesla V100-SXM2... On  00000000:07:00.0 Off   0          Default  |
| N/A  57C   P0  240W / 300W | 15525MiB / 16255MiB | 100%      N/A      |
|                         |          |          |          |
+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Processes:
| GPU  GI  CI      PID  Type  Process name          GPU Memory  |
| ID   ID           ID
|-----+-----+-----+-----+-----+-----+-----+
| 0  N/A  N/A     156    C  python3          15525MiB  |
+-----+

```

[Next: Achieving High Cluster Utilization with Over-Quota GPU Allocation](#)

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:

Project	Quota
team-a	4
team-b	2
team-c	2
team-d	8

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.

NAME	STATUS	AGE	NODE	IMAGE	TYPE	PROJECT	USER	GPUs	CREATED BY CLI	SERVICE URL(S)
b-4-gg	Running	2m	dgx1-2	gcr.io/run-ai-demo/quickstart	Train	team-b	root	2	true	
c-5-g	Running	2m	dgx1-2	gcr.io/run-ai-demo/quickstart	Train	team-c	root	1	true	
c-4-gg	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-c	root	2	true	
b-3-g	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-b	root	1	true	
c-3-g02	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Interactive	team-c	root	0.2	true	
d-1-gggg	Running	2m	dgx1-2	gcr.io/run-ai-demo/quickstart	Train	team-d	root	4	true	
c-2-g03	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Interactive	team-c	root	0.3	true	
c-1-g05	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Interactive	team-c	root	0.5	true	
a-2-gg	Running	3m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-a	root	2	true	
b-2-g04	Running	3m	dgx1-2	gcr.io/run-ai-demo/quickstart	Interactive	team-b	root	0.4	true	
a-1-g	Running	3m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-a	root	1	true	
b-1-g06	Running	3m	dgx1-2	gcr.io/run-ai-demo/quickstart	Interactive	team-b	root	0.6	true	
a-1-1-jupyter	Running	3m	dgx1-1	jupyter/base-notebook	Interactive	team-a	root	1	true	http://10.61.218.134/a-1-1-jupyter , https://10.61.218.134/a-1-1-jupyter

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.

[Next: Basic Resource Allocation Fairness](#)

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.

Next: Over-Quota Fairness

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:

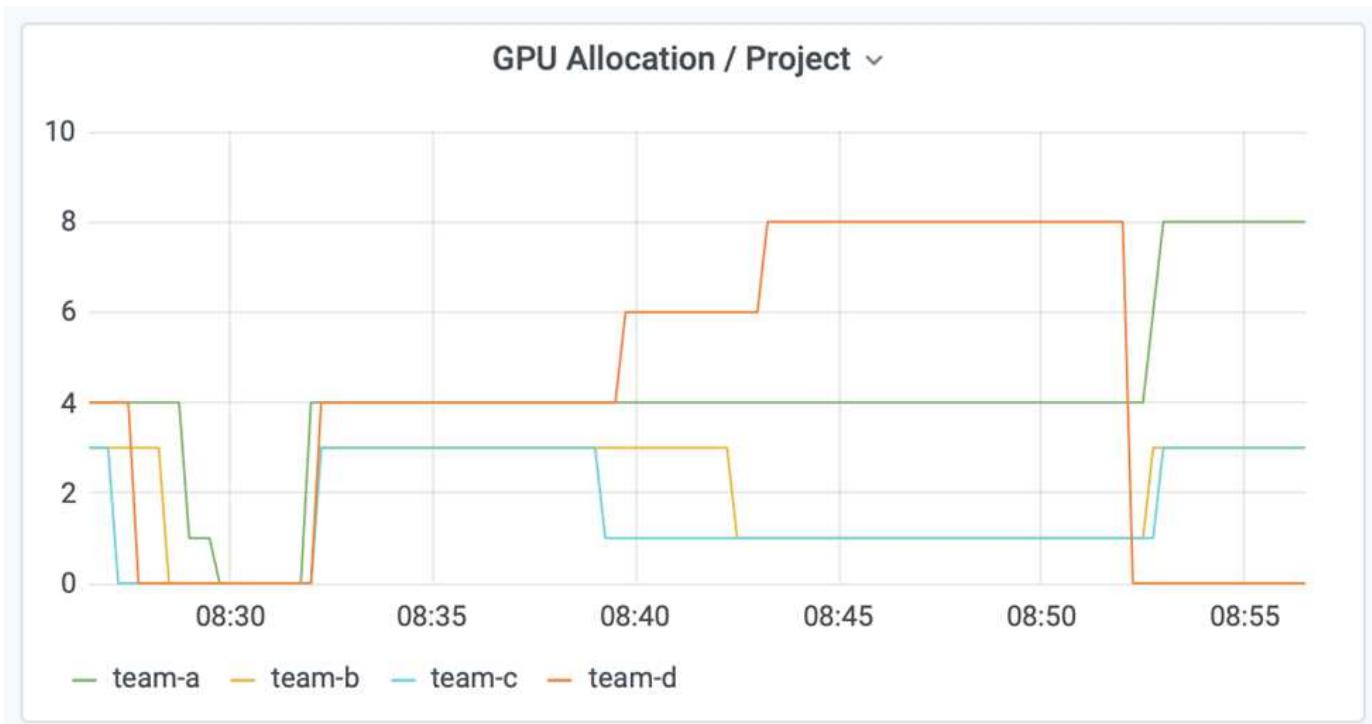
- The system starts to de-queue the workloads of other teams.
- The order of the dequeuing is decided according to fairness algorithms, such that team-b and team-c get the same amount of over-quota GPUs (since they have a similar quota), and team-a gets a double amount of GPUs since their quota is two 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:

Project	GPUs allocated	Comment
team-a	8/4	Four GPUs over the quota. Empty queue.

Project	GPUs allocated	Comment
team-b	4/2	Two GPUs over the quota. One workload queued.
team-c	4/2	Two GPUs over the quota. One workload queued.
team-d	0/8	Not using GPUs at all, no queued workloads.

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.



Next: [Saving Data to a Trident-Provisioned PersistentVolume](#)

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:

```
$ runai submit pvc-test -p team-a --pvc test:/tmp/pvc1mount -i gcr.io/run-ai-demo/quickstart -g 1
```

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.

[Next: Conclusion](#)

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.

[Next: Testing Details for Section 4.8](#)

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:

Project	Image	# GPUs	Total	Comment
team-a	Jupyter	1	1/4	—
team-a	NetApp	1	2/4	—
team-a	Run:AI	2	4/4	Using all their quota
team-b	Run:AI	0.6	0.6/2	Fractional GPU

Project	Image	# GPUs	Total	Comment
team-b	Run:AI	0.4	1/2	Fractional GPU
team-b	NetApp	1	2/2	-
team-b	NetApp	2	4/2	Two over quota
team-c	Run:AI	0.5	0.5/2	Fractional GPU
team-c	Run:AI	0.3	0.8/2	Fractional GPU
team-c	Run:AI	0.2	1/2	Fractional GPU
team-c	NetApp	2	3/2	One over quota
team-c	NetApp	1	4/2	Two over quota
team-d	NetApp	4	4/8	Using half of their quota

Command structure:

```
$ runai submit <job-name> -p <project-name> -g <#GPUs> -i <image-name>
```

Actual command sequence used in testing:

```
$ runai submit a-1-1-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-1-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:

Project	GPUs Allocated	Workloads Queued
team-a	4/4 (soft quota/actual allocation)	None
team-b	4/2	None
team-c	4/2	None
team-d	4/8	None

See the section [Achieving High Cluster Utilization with Over-quota GPU Allocation](#) for discussions on the proceeding testing scenario.

[Next: Testing Details for Section 4.9](#)

Testing Details for Section 4.9

This section contains testing details for the section [Basic Resource Allocation Fairness](#).

Submit jobs in the following order:

Project	# GPUs	Total	Comment
team-d	2	6/8	Team-b/c workload pauses and moves to pending.
team-d	2	8/8	Other team (b/c) workloads pause and move to pending.

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:

Project	GPUs Allocated	Workloads Queued
team-a	4/4	None
team-b	2/2	None
team-c	2/2	None
team-d	8/8	None

See the section [Basic Resource Allocation Fairness](#) for a discussion on the proceeding testing scenario.

[Next: Testing Details for Section 4.10](#)

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:

Project	# GPUs	Total	Comment
team-a	2	4/4	1 workload queued
team-a	2	4/4	2 workloads queued
team-b	2	2/2	2 workloads queued
team-c	2	2/2	2 workloads queued

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:

Project	GPUs Allocated	Workloads Queued
team-a	4/4	Two workloads asking for GPUs two each
team-b	2/2	Two workloads asking for two GPUs each
team-c	2/2	Two workloads asking for two GPUs each
team-d	8/8	None

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.

Next: [Where to Find Additional Information](#)

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
<https://www.nvidia.com/en-us/data-center/dgx-1/>
 - NVIDIA V100 Tensor Core GPU
<https://www.nvidia.com/en-us/data-center/tesla-v100/>

- NVIDIA NGC
<https://www.nvidia.com/en-us/gpu-cloud/>
- 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/>
<https://docs.run.ai/Administrator/Researcher-Setup/Installing-the-Run-AI-Command-Line-Interface/>
 - 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
<https://www.netapp.com/us/media/tr-4798.pdf>
 - 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
<https://www.netapp.com/us/media/ds-3582.pdf>
 - NetApp Flash Advantage for All Flash FAS
<https://www.netapp.com/us/media/ds-3733.pdf>
 - ONTAP 9 Information Library
<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>
 - NetApp ONTAP FlexGroup Volumes technical report
<https://www.netapp.com/us/media/tr-4557.pdf>
- NetApp ONTAP AI
 - ONTAP AI with DGX-1 and Cisco Networking Design Guide
<https://www.netapp.com/us/media/nva-1121-design.pdf>
 - ONTAP AI with DGX-1 and Cisco Networking Deployment Guide
<https://www.netapp.com/us/media/nva-1121-deploy.pdf>
 - ONTAP AI with DGX-1 and Mellanox Networking Design Guide
<http://www.netapp.com/us/media/nva-1138-design.pdf>
 - ONTAP AI with DGX-2 Design Guide
<https://www.netapp.com/us/media/nva-1135-design.pdf>

Copyright Information

Copyright © 2022 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.