Technical White Paper

Deep Learning with Dell Technologies Cloud PowerScale for Google Cloud

Abstract

This document shows how Dell Technologies™ Cloud PowerScale™ for Google Cloud can be used with the Google Cloud Compute Engine and 120 NVIDIA® GPUs to perform deep learning at scale.

June 2020

H18230

Revisions

Date	Description
May 2020	Initial release, PowerScale for Google Cloud, tier 2 test results
June 2020	Update for PowerScale for Google Cloud, tier 1 test results

Acknowledgments

Authors: Claudio Fahey, Damien Mas

Table of contents

Executive summary

Dell Technologies™ Cloud PowerScale™ for Google Cloud is a Google native scalable file service providing high-speed file access over SMB, NFS, and HDFS. It is powered by the Dell EMC™ PowerScale family which includes Dell EMC Isilon™ and PowerScale nodes as well as PowerScale OneFS™ which runs across these systems. PowerScale for Google Cloud provides linear scalability for performance and capacity, storage tiering, and multiprotocol access, making it an ideal storage platform for many deep-learning workloads.

This white paper demonstrates how PowerScale for Google Cloud can be used with Google Cloud Compute Engine and 120 NVIDIA® GPUs to perform deep-learning training at scale. The results of industry-standard image classification training benchmarks using TensorFlow are included.

Audience

This document is intended for organizations that are looking to run deep learning (DL) and other artificial intelligence (AI) workloads in the cloud. Solution architects, system administrators, data scientists, data engineers, and other interested readers within those organizations constitute the target audience.

1 Introduction

Dell Technologies has multiple GPU-accelerated server models for data center environments including the Dell EMC PowerEdge™ C4140 and Dell DSS 8440. We also have complete solutions that combine these servers with networking and Isilon scale-out NAS. For organizations looking to utilize Google Cloud, we now offer PowerScale for Google Cloud. PowerScale for Google Cloud is a native Google Cloud offering within the Google Cloud intranet, powered by Dell EMC Isilon technology, and managed by Dell Enterprise SLA Services. It provides terabit networking and sub-millisecond latency between the PowerScale for Google Cloud nodes and Google Compute Engine services. The service is billed through Google Cloud (as with all other Google services) and can count toward any committed Google Cloud spend. Since PowerScale for Google Cloud is native within Google Cloud, there are no egress charges associated with moving data between this storage platform and other Google Cloud services.

DL is an area of AI which uses artificial neural networks to enable accurate pattern recognition of complex real-world patterns by computers. These new levels of innovation have applicability across nearly every industry vertical. Some of the early adopters include advanced research, precision medicine, high tech manufacturing, advanced driver assistance systems (ADAS), and autonomous driving. Building on these initial successes, AI initiatives are springing up in various business units, such as manufacturing, customer support, life sciences, marketing, and sales. Gartner predicts that AI augmentation will generate $2.9 trillion in business value by 2021 alone. Organizations are faced with a multitude of complex choices that are related to data, analytic skill-sets, software stacks, analytic toolkits, and infrastructure components; each with significant implications on the time to market and the value associated with these initiatives.

2 Solution architecture

2.1 Overview

Each PowerScale for Google Cloud customer is provided with a dedicated PowerScale for Google cluster. Management of the PowerScale cluster is accomplished through the Google Cloud Console for a native cloud experience. Since each PowerScale cluster is dedicated to a single customer, performance is predictable and data tenancy is assured. Customers can leverage native OneFS data services for moving data between their on-premises and PowerScale for Google Cloud instances.

Figure 1 shows the key components that make up this DL solution. Note that in a customer deployment, the number of GCE VM instances and Isilon storage nodes will vary and can be scaled independently to meet the requirements of the specific workload.

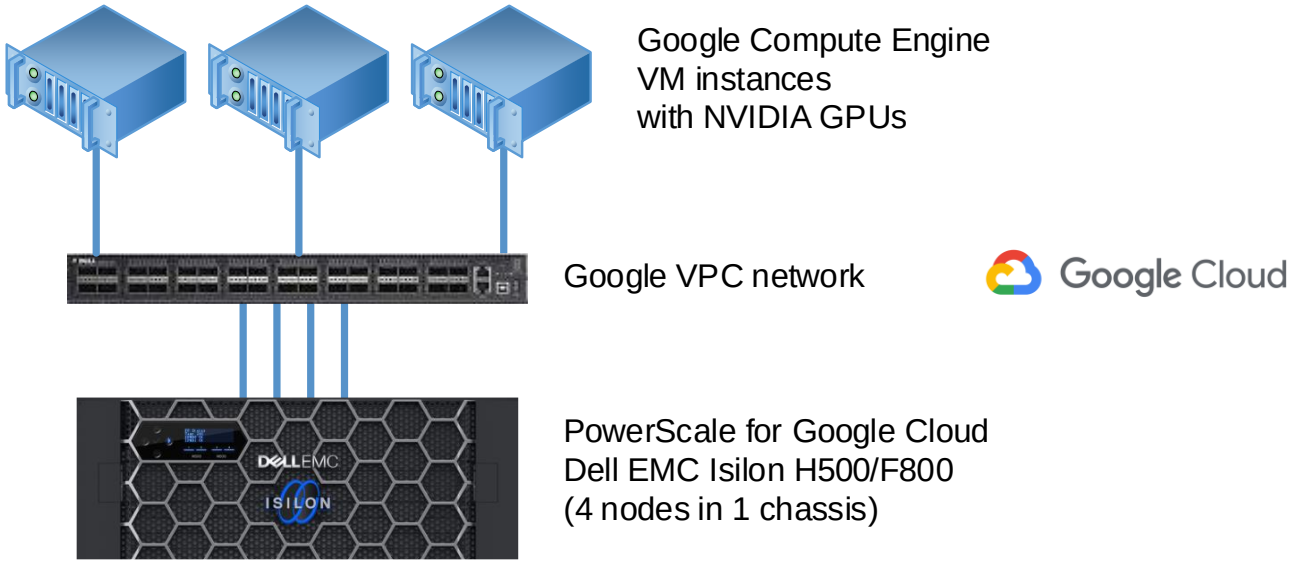

Figure 1 Solution architecture

2.2 High performance scale-out NAS

Many common distributed DL frameworks today, such as TensorFlow, require training files to be accessible on a shared file system. For very large datasets, it is generally best to store a single copy of the dataset on an NFS server, where compute nodes can read the files needed just prior to each training batch. This allows training to begin as soon as the VM instance starts up, without waiting for all training files to be copied.

2.2.1 Dell EMC Isilon H500 hybrid scale-out NAS

Dell EMC Isilon hybrid storage platforms, powered by the PowerScale OneFS operating system, use a highly versatile yet simple scale-out storage architecture to speed access to massive amounts of data, while dramatically reducing cost and complexity. The hybrid storage platforms are highly flexible and strike the balance between large capacity and high-performance storage to provide support for a broad range of enterprise file workloads. The H500 provides a balance of performance, capacity, and value to support a wide range of file workloads. The H500 delivers up to 5 GB/s bandwidth per chassis and provides capacity options ranging from 120 TB to 720 TB per chassis.

Each H500 chassis, shown in Figure 2, contains four storage nodes, 60 SATA hard drives and eight 40 GbE network connections. OneFS combines up to 252 nodes in 63 chassis into a single high-performance file system designed to handle the most intense I/O workloads such as DL. As performance and capacity demands increase, both can be scaled-out simply and non-disruptively, allowing applications and users to continue working.

Figure 2 Isilon H500/F800 chassis, containing four storage nodes

2.2.2 Dell EMC Isilon all-flash F800 scale-out NAS

Dell EMC Isilon all-flash storage platforms, powered by the OneFS operating system, provide a powerful yet simple scale-out storage architecture to speed access to massive amounts of unstructured data, while dramatically reducing cost and complexity. With a highly dense design that contains 4 nodes within a single 4U chassis, all-flash platforms deliver extreme performance and efficiency for your most demanding unstructured data applications and workloads. The F800 delivers up to 15 GB/s bandwidth per chassis and provides capacity options ranging from 96 TB to 924 TB per chassis.

Each F800 chassis, shown in Figure 2, contains four storage nodes, 60 SATA hard drives and eight 40 GbE network connections. OneFS combines up to 252 nodes in 63 chassis into a single high-performance file system designed to handle the most intense I/O workloads such as DL. As performance and capacity demands increase, both can be scaled-out simply and non-disruptively, allowing applications and users to continue working.

Two solutions have been tested for this document:

- PowerScale for Google Cloud Tier II: 4 H500 nodes, in one chassis
- PowerScale for Google Cloud Tier 1: 4 F800 nodes, in one chassis

Dell EMC Isilon H500 and F800 systems have the following features.

Isilon H500	Isilon F800
High capacity with the ability to grow as needed: • 120 TB to 720 TB raw hard drive capacity per chassis; up to 45 PB per cluster • Up to 5 GB/s throughput per chassis	High capacity with the ability to grow as needed: • 96 TB to 924 TB raw SSD capacity per chassis; up to 58 PB per cluster • Up to 15 GB/s throughput per chassis
The ability to run AI in-place on data using multiprotocol access: • Multiprotocol support such as SMB, NFS, HTTP, and native HDFS to maximize operational flexibility This eliminates the need to migrate/copy data and results over to a separate AI stack.	

<table>
<tr><th>Isilon H500</th><th>Isilon F800</th></tr>
<tr><td colspan="2">Enterprise grade features out-of-box:

• Enterprise data protection and resiliency
• Robust security options

This enables organizations to manage AI data life cycle with minimal cost and risk, while protecting data and meeting regulatory requirements.</td></tr>
<tr><td>Extreme scale:

• Grow-as-you-go scalability with up to 45 PB hard drive capacity per cluster
• New nodes can be added to a cluster simply by connecting power, back-end Ethernet and front-end Ethernet
• As new nodes are added, storage capacity, throughput, IOPS, cache, and CPU grow
• Up to 63 chassis (252 nodes) may be connected to form a single cluster with a single namespace and a single coherent cache
• Up to 85% storage efficiency to reduce costs
• Optional data deduplication and compression enabling up to a 3:1 data reduction
• Seamlessly tier between All Flash, Hybrid, and Archive nodes using SmartPools</td><td>Extreme scale:

• Grow-as-you-go scalability with up to 58 PB SSD capacity per cluster
• New nodes can be added to a cluster simply by connecting power, back-end Ethernet and front-end Ethernet
• As new nodes are added, storage capacity, throughput, IOPS, cache, and CPU grow
• Up to 63 chassis (252 nodes) may be connected to form a single cluster with a single namespace and a single coherent cache
• Up to 85% storage efficiency to reduce costs
• Optional data deduplication and compression enabling up to a 3:1 data reduction
• Seamlessly tier between All Flash, Hybrid, and Archive nodes using SmartPools</td></tr>
</table>

Organizations can achieve AI at scale in a cost-effective manner, enabling them to handle multi-petabyte datasets with high-resolution content without re-architecture or performance degradation.

There are several key features of Isilon OneFS that make it an excellent storage system for DL workloads that require performance, concurrency, and scale. These features are detailed below.

2.2.3 Storage tiering

Dell EMC Isilon SmartPools software enables multiple levels of performance, protection, and storage density to co-exist within the same file system and unlocks the ability to aggregate and consolidate a wide range of applications within a single extensible, ubiquitous storage resource pool. This helps provide granular performance optimization, workflow isolation, higher utilization, and independent scalability – all with a single point of management.

SmartPools allows you to define the value of the data within your workflows based on policies and automatically aligns data to the appropriate price/performance tier over time. Data movement is seamless, and with file-level granularity and control using automated policies, manual control, or API, you can tune performance and layout, storage tier alignment and protection settings—all with minimal impact to your end users.

Storage tiering has a very convincing value proposition, namely segregating data according to its business value and aligning it with the appropriate class of storage and levels of performance and protection. Information Lifecycle Management techniques have been around for several years, but have typically suffered

from the following inefficiencies: complex to install and manage, involves changes to the file system, requires the use of stub files, and so forth.

Dell EMC Isilon SmartPools is a next generation approach to tiering that facilitates the management of heterogeneous clusters. The SmartPools capability is native to the Isilon OneFS scale-out file system, which allows for unprecedented flexibility, granularity, and ease of management. In order to achieve this, SmartPools leverages many of the components and attributes of OneFS, including data layout and mobility, protection, performance, scheduling, and impact management.

A typical Isilon cluster will store multiple datasets with different performance, protection, and price requirements. Generally, files that have been recently created and accessed should be stored in a hot tier while files that have not been accessed recently should be stored in a cold tier. Because Isilon supports tiering based on a file's access time, this can be performed automatically. For storage administrators that want more control, complex rules can be defined to set the storage tier based on a file's path, size, or other attributes.

All files on Isilon are always immediately accessible (read and write) regardless of their storage tier and even while being moved between tiers. The file system path to a file is not changed by tiering. Storage tiering policies are applied and files are moved by the Isilon SmartPools job, which runs daily at 22:00 by default.

For more details, see the document Storage Tiering with Dell EMC Isilon SmartPools.

2.2.4 OneFS caching

The OneFS caching infrastructure design is predicated on aggregating the cache present on each node in a cluster into one globally accessible pool of memory. This allows all the memory cache in a node to be available to every node in the cluster. Remote memory is accessed over an internal interconnect and has lower latency than accessing hard disk drives and SSDs.

For files marked with an access pattern of concurrent or streaming, OneFS can take advantage of prefetching of data based on heuristics used by the Isilon SmartRead component. This greatly improves sequential-read performance across all protocols and means that reads come directly from RAM within milliseconds. For high-sequential cases, SmartRead can very aggressively prefetch ahead, allowing reads of individual files at very high data rates.

OneFS uses up to three levels of read cache, plus an NVRAM-backed write cache. L1 and L2 read caches use RAM while L3 uses the SSDs that are available on all Isilon hybrid nodes.

For more details, see the document OneFS SmartFlash.

2.2.5 Locks and concurrency

OneFS has a fully distributed lock manager that coordinates locks on data across all nodes in a storage cluster. The lock manager is highly extensible and allows for multiple lock personalities to support both file system locks as well as cluster-coherent protocol-level locks such as SMB share mode locks or NFS advisory-mode locks. Every node in a cluster is a coordinator for locking resources and a coordinator is assigned to lockable resources based upon an advanced hashing algorithm.

Efficient locking is critical to support the parallel I/O profile demanded by many iterative AI and DL workloads enabling concurrent file read access up into the millions.

For more details, see the document OneFS Technical Overview.

2.3 Key components

Table 1 shows the key components as tested for this document.

Table 1 Key components

Component	Purpose	Quantity
PowerScale for Google Cloud – Tier II Dell EMC Isilon H500 120 TB hard drive 12.8 TB SSD 512 GB RAM Four 1 GbE, eight 40 GbE interfaces	NAS	1 4U chassis (4 nodes)
PowerScale for Google Cloud - Tier I Dell EMC Isilon F800 342 TB SSD 1024 GB RAM Four 1 GbE, eight 40 GbE interfaces	NAS	1 4U chassis (4 nodes)
Google Cloud Platform Filestore Premium Tier, 2.5 TB	NAS	1
Google Compute Engine VM instance Machine type: n1-standard-16 16 vCPUs 60 GB memory CPU platform: Intel Skylake GPUs: 4 x NVIDIA Tesla P4 (8 GB each) Region: us-east4-a	Compute node	30 (120 GPUs)

2.4 Software versions

Table 2 shows the software versions that were tested for this document.

Table 2 Software versions

Component	Version
AI Benchmark Util	https://github.com/claudiofahey/ai-benchmark-util/commit/ca7f5d2
Dell EMC Isilon – OneFS	8.1.2.0 for H500 8.2.1.0 for F800
GCP VM instance image	c0-common-gce-gpu-image-20200128 Google, GPU Optimized Debian, m32 (with CUDA 10.0), A Debian 9 based image with CUDA/CuDNN/NCCL preinstalled
NVIDIA GPU Cloud TensorFlow Image	nvcr.io/nvidia/tensorflow:19.09-py3
TensorFlow	1.14.0
TensorFlow Benchmarks	https://github.com/claudiofahey/benchmarks/commit/31ea13f

3 Deep learning training performance and analysis

3.1 Benchmark methodology

In order to measure the performance of the solution, various benchmarks from the TensorFlow Benchmarks repository were run. This suite of benchmarks performs training of an image classification convolutional neural network (CNN) on labeled images. Essentially, the system learns whether an image contains a cat, dog, car, or train. The well-known ILSVRC2012 image dataset (often referred to as ImageNet) was used. This dataset contains 1,281,167 training images in 144.8 GB[1]. All images are grouped into 1000 categories or classes. This dataset is commonly used by DL researchers for benchmarking and comparison studies.

The individual JPG images in the ImageNet dataset were converted to 1024 TFRecord files. The TFRecord file format is a Protocol Buffers binary format that combines multiple JPG image files together with their metadata (bounding box for cropping and label) into one binary file. It maintains the image compression offered by the JPG format and the total size of the dataset remained roughly the same (148 GB). The average image size was 115 KB.

As many datasets are often significantly larger than ImageNet, we wanted to determine the performance with datasets that are larger than the 512 GB of coherent shared cache available across the four-node Isilon H500 cluster. To accomplish this, we simply made seven exact copies of each TFRecord file, creating a 1.0 TB dataset. Having seven copies of the exact same images does not improve training accuracy or speed but it does produce the same I/O pattern for the storage, network, and GPUs. Having identical files did not provide an unfair advantage as Isilon deduplication was not enabled and all images are reordered randomly (shuffled) in the input pipeline. We applied the same methodology for the Isilon F800 and created a dataset larger than 1024 GB. To accomplish that we simply made 13 copies of Each TFRecord file, creating a 1.4 TB dataset.

One of the critical questions one has when trying to size a system is how fast the storage must be so that it is not a bottleneck. To answer this question, we created a subset of the ImageNet dataset by using only 100 of the 1024 TFRecord files. This produced a 14.5 GB dataset that could be cached by each compute node with 60 GB of RAM. After performing several warm-up runs, the benchmark was run, and it was confirmed that there was virtually zero NFS network I/O to the storage system. The image rate (images/sec) measured in this way accounts for the significant preprocessing pipeline as well as the GPU computation. To determine the throughput (bytes/sec) demanded by this workload, we simply multiply the images/sec by the average image size (118 KB). In the next section, results using this method are labeled Linux Cache.

Prior to each execution of the benchmark, the L1, L2, and L3 caches on Isilon were flushed with the command **isi_for_array isi_flush -123k**. In addition, the Linux buffer cache was flushed on all compute systems by running **sync; echo 3 > /proc/sys/vm/drop_caches**. However, note that the training process will read the same files repeatedly and after just several minutes, much of the data will be served from one of these caches.

Finally, we ran the same benchmark with the TFRecord files stored on the premium tier of Google Cloud Filestore.

[1] All unit prefixes in this document use the SI standard (base 10) where 1 GB is 1 billion bytes.

3.2 Benchmark results

There are a few conclusions that we can make from the benchmarks represented in Figure 3.

- When all data comes from the Linux cache, we see ideal linear scaling 4–32 GPUs. This means that the image throughput with 32 GPUs equals eight times the image throughput with 4 GPUs.
- At 64 GPUs and beyond, the Linux cache tests show sublinear performance, indicating that there are inefficiencies in the training algorithm or communication bottlenecks. As our goal was to investigate storage performance, the cause of this was not investigated. However, this is the upper limit for what we can possibly achieve with any storage technology.
- Using the Isilon H500, image throughput and therefore storage throughput match the Linux cache tests 4–64 GPUs and reach up to 800 MB/sec. At 96 GPUs and beyond, we can see that the throughput is significantly less than the Linux cache tests and we can conclude that storage is the bottleneck.
- Using the Isilon F800, image throughput and therefore storage throughput match the Linux cache tests from 4 to 96 GPUs and reach up to 1200 MB/sec, at 120 GPUs and beyond, we can see that the throughput is slightly less than the Linux cache tests, and we can conclude that storage is not the bottleneck.
- Google Cloud Filestore matches the Linux cache tests 4–32 GPUs.
- Isilon performance was better than or equal to Google Cloud Filestore in all cases.
- During Isilon tests, GPU utilization was >97% for 4–32 GPUs. At 64 GPUs, GPU utilization dropped to 90%. At 120 GPUs, GPU utilization dropped to 50%.
- Average CPU utilization was 40-60% for all tests.

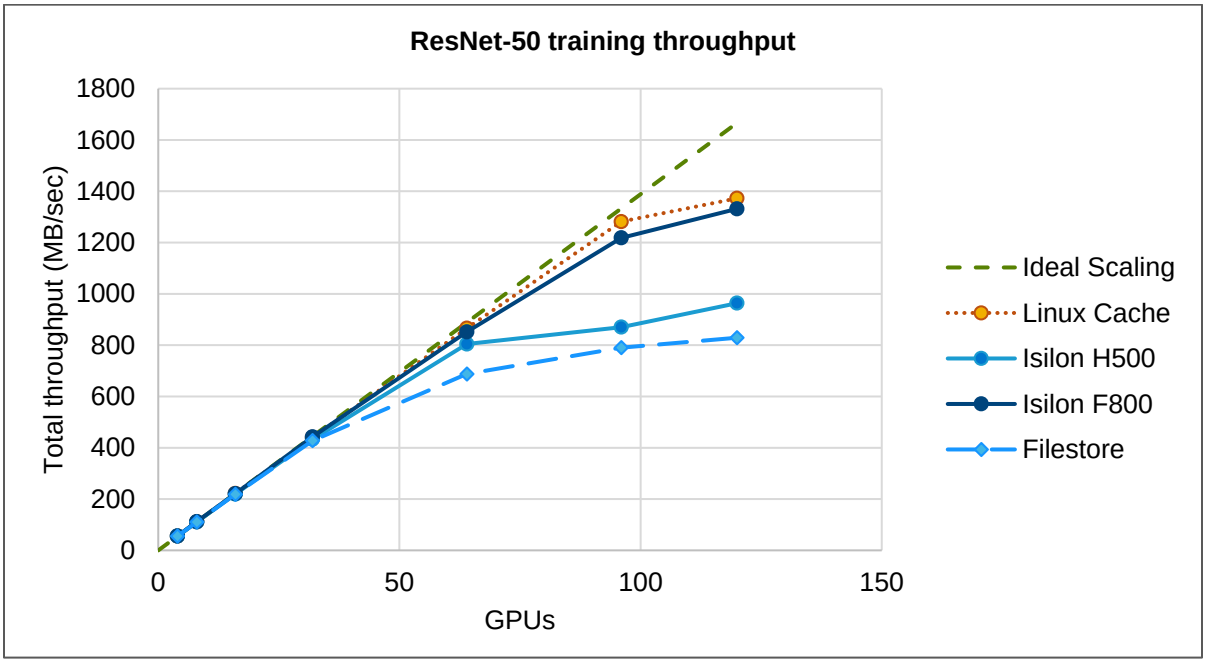

Figure 3 Model development: Training benchmark results

4 Storage-only performance

4.1 Storage network performance using iPerf

When investigating the performance of a NAS, it is often useful to measure the network performance in isolation using a network benchmark tool such as iPerf. iPerf comes installed on all Isilon nodes and can be started from the Isilon CLI.

In the first experiment, shown in Figure 4, we used one GCE worker and one Isilon H500 node. The number of TCP connections (*parallel* parameter) was varied between 1 and 32. The write direction (GCE worker sending data to Isilon) topped out at 2,275 MB/sec while the read direction reached 3,788 MB/sec

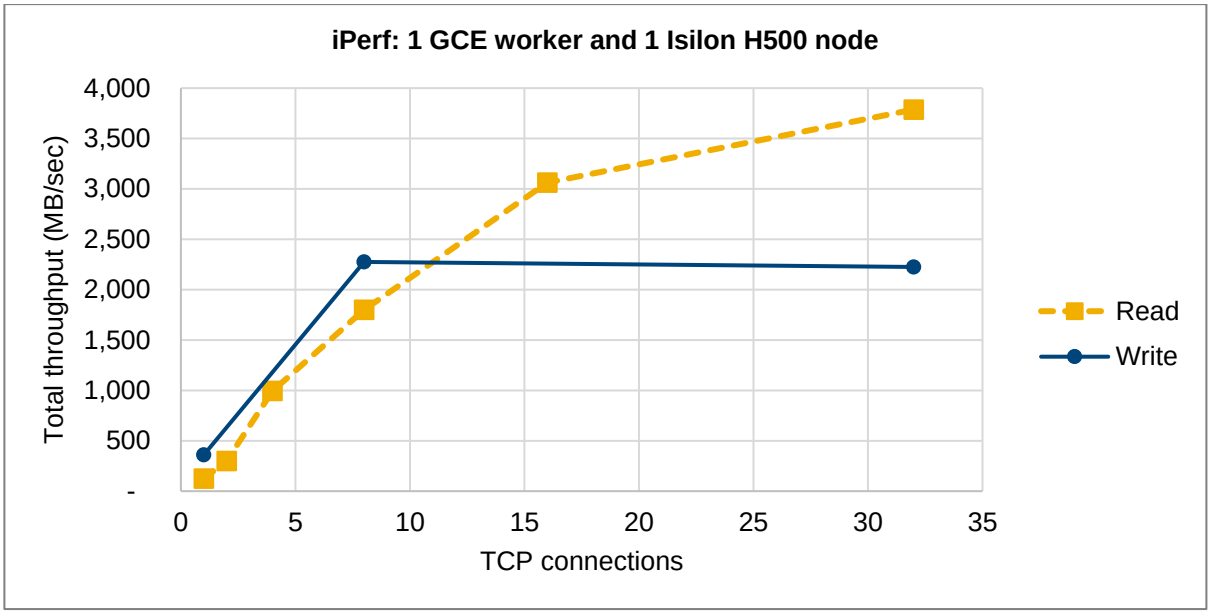

Figure 4 iPerf throughput from one GCE worker to one H500 Isilon node; a read means that Isilon storage sends data to the GCE worker

In the second experiment, shown in Figure 5, we used one GCE worker and 1 Isilon F800 node. The number of TCP connections (*parallel* parameter) was varied between 1 and 32. The write direction (GCE worker sending data to Isilon) topped out at 3950 MB/sec while the read direction reached 4737 MB/sec.

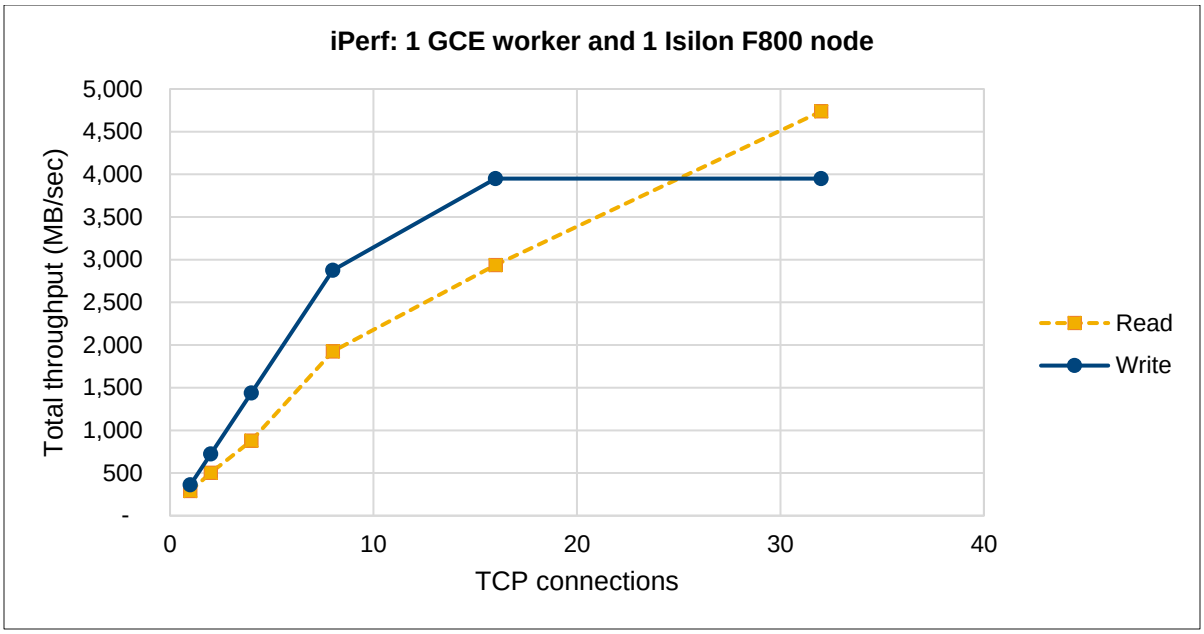

Figure 5 iPerf throughput from 1 GCE worker to 1 Isilon node; Read means Isilon sends data to the GCE worker

In the next experiment, shown in Figure 6, we increased the number of GCE workers to 2, 4, and 8 while using 2, 4, and 4 Isilon H500 nodes, respectively. We measured a maximum write throughput of 7206 MB/sec and a read throughput of 18,538 MB/sec.

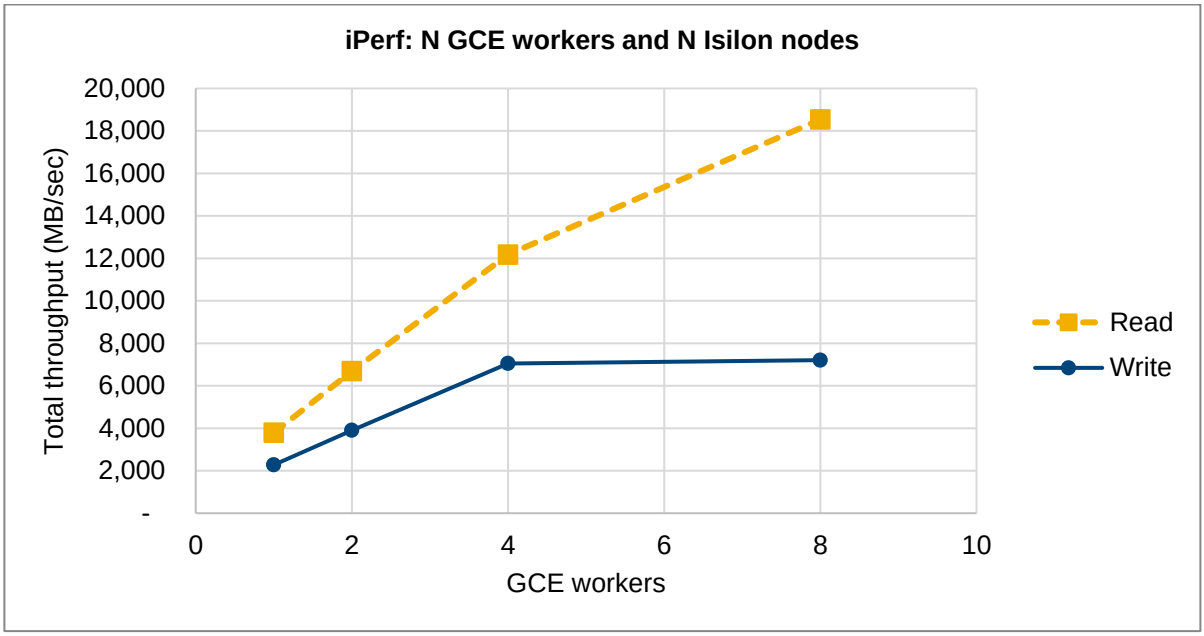

Figure 6 iPerf throughput using multiple GCE workers and multiple Isilon H500 nodes

With four or more GCE workers, the measured network limits are beyond the best-case performance limits of the 4-node Isilon H500. However, for a workload that requires extreme performance with fewer than four GCE

workers, or a faster Isilon cluster, the network limitations of Google Cloud and PowerScale for Google Cloud must be carefully considered.

In the next experiment, shown in Figure 7, we increased the number of GCE workers to 2, 4, and 8 while using 2, 4, and 4 Isilon F800 nodes, respectively. We measured a maximum write throughput of 7206 MB/sec and a read throughput of 18,538 MB/sec.

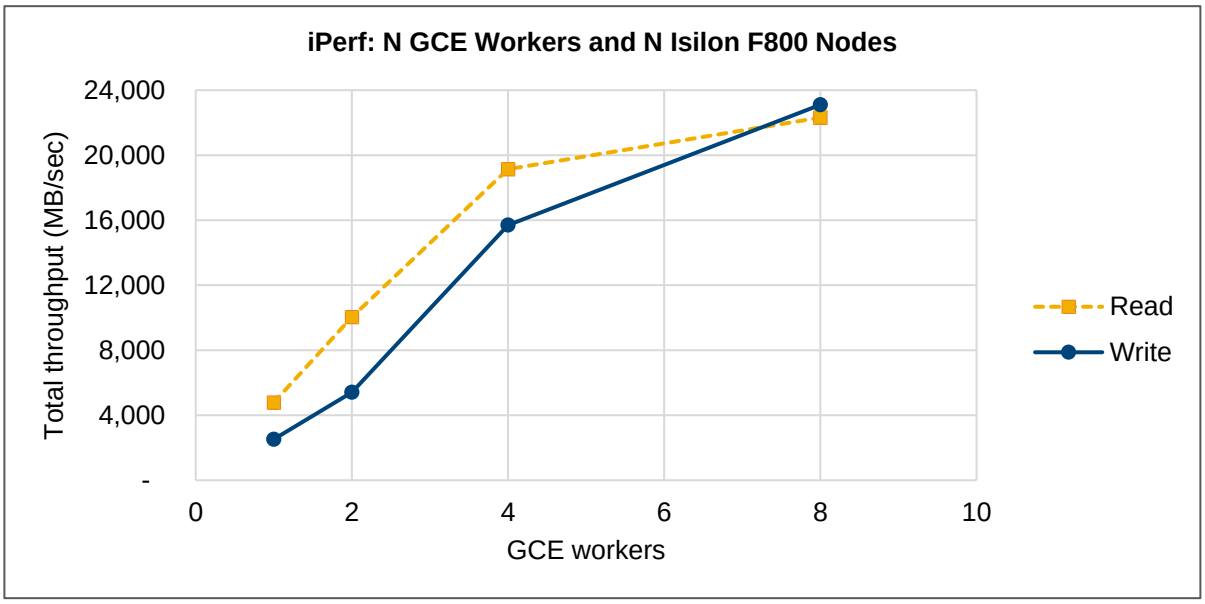

Figure 7 iPerf throughput using multiple GCE workers and multiple Isilon F800 nodes.

4.2 Storage-only performance using FIO

To understand the limits of Isilon storage I/O in this environment, we used the common storage benchmark tool FIO to perform concurrent sequential reads from Isilon. As shown in Figure 8, FIO was able to achieve a maximum throughput of 3762 MB/sec for reads and 2874 MB/sec for writes. These rates were achieved when using 18 GCE worker nodes and 36 total concurrent files.

When using only six GCE worker nodes but the same total number of concurrent files, the read throughput dropped to 3023 MB/sec, indicating a likely per-VM limitation of approximately 500 MB/sec.

See the FIO section in the appendix for details about how FIO was run.

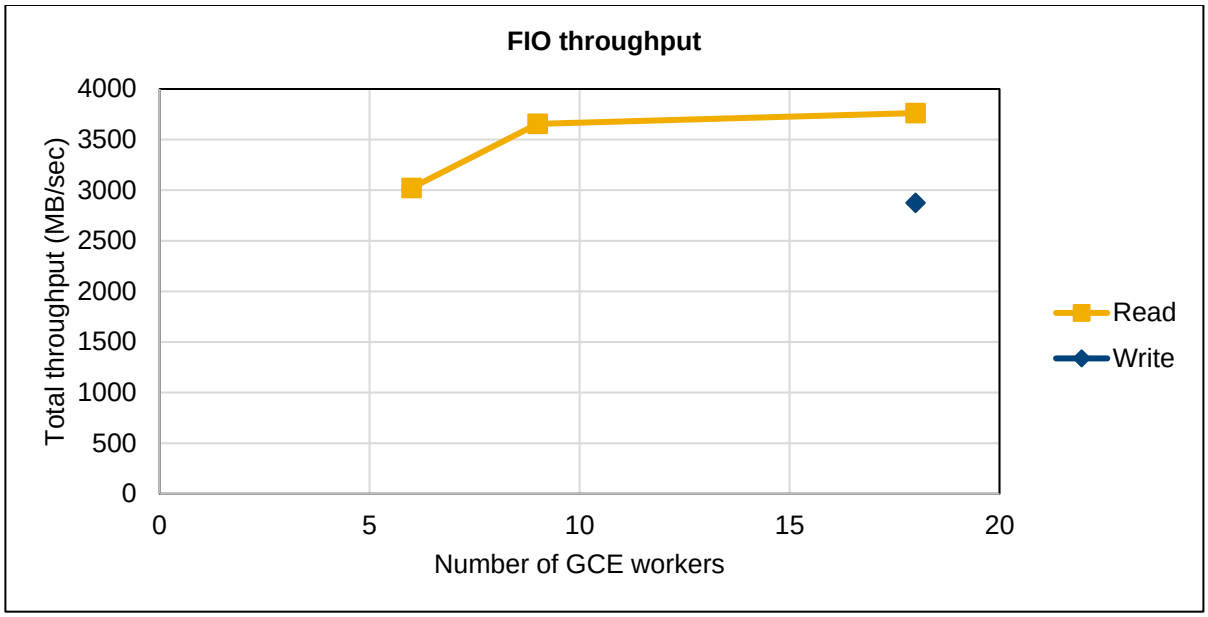

Figure 8 FIO throughput with Isilon, sequential I/O, 4 Isilon H500 nodes, 36 concurrent files

5 Conclusion

This document presented a solution using PowerScale for Google Cloud and Google Compute Engine that can be used to perform DL training at scale. The Isilon H500 nodes used in the PowerScale for Google Cloud Tier II tests provided good performance for up to 64 NVIDIA Tesla P4 GPUs. The Isilon F800 used in the PowerScale for Google Cloud Tier I tests provided good performance for up to 120 NVIDIA Tesla P4 GPUs. When using more GPUs or faster GPUs available in other Google Cloud regions, additional nodes can be added to any performance tier to increase performance and scalability.

It is important to point out that AI algorithms have a diverse set of requirements with various compute, memory, I/O, and disk capacity profiles. That said, the architecture and the performance data points presented in this white paper can be used as the starting point for building AI solutions tailored to varied sets of resource requirements. More importantly, all the components of this architecture are linearly scalable and can be independently expanded to provide AI solutions that can manage tens of PBs of data.

While the benchmarks presented here provide several performance data points, there are several other operational benefits of persisting data for AI on Isilon:

- The ability to run AI in-place on data using multiprotocol access
- Enterprise grade features out-of-box
- Seamlessly tier to more cost-effective nodes
- Scale up to 58 PB per cluster (some node models will have lower limits)

In summary, PowerScale for Google Cloud delivers the capacity, performance and high concurrency to eliminate the I/O storage bottlenecks for AI. This provides a rock-solid foundation for large scale, enterprise-grade DL solutions with a future proof scale-out architecture that meets your AI needs of today and scales for the future.

A System configuration

A.1 Google Compute Engine

GCE VM instances were provisioned using the following command.

```
gcloud beta compute \
--project=isilon-project \
instances \
create \
dl-worker-001 \
--machine-type=n1-standard-16 \
--subnet=isilon-vpc \
--metadata=VmDnsSetting=GlobalOnly \
--maintenance-policy=TERMINATE \
--image=c0-common-gce-gpu-image-20200128 \
--image-project=ml-images \
--boot-disk-size=50GB \
--boot-disk-type=pd-standard \
--reservation-affinity=any" \
--accelerator=type=nvidia-tesla-p4,count=4 \
--min-cpu-platform=Intel\ Skylake \
--zone=us-east4-a \
--boot-disk-device-name=dl-worker-001
```

A.2 Install AI Benchmark Utilities

Throughout this document, we use several Bash and Python scripts that are available at AI Benchmark Util. These should be installed on one GCE VM instance.

```
cd /mnt/isilon/data
git clone https://github.com/claudiofahey/ai-benchmark-util
cd ai-benchmark-util
git checkout gcp
sudo apt install python3-pip
pip3 install setuptools
pip3 install --requirement requirements.txt
```

A.3 Isilon volume mounting

Isilon is used for two types of file storage. First, it is used for scripts, binaries and logs. This requires low bandwidth and must support NFS locks for consistency and proper visibility of changes. This generally uses the default mount options and is mounted with:

```
mount -t nfs 10.200.10.151:/ifs /mnt/isilon
```

Next, there is the data that will be read or written at high speed. To ensure an even balance of traffic across the Isilon nodes, we will want to carefully create several mounts explicitly to the IP addresses of several Isilon nodes.

As an example, the commands below can be run on each GCE VM instance to mount to each of the four Isilon interfaces.

```
mount -t nfs 10.200.10.151:/ifs \
rsize=524288,wsize=524288,nolock /mnt/isilon1
mount -t nfs 10.200.10.152:/ifs \
rsize=524288,wsize=524288,nolock /mnt/isilon2
...
mount -t nfs 10.200.10.154:/ifs \
rsize=524288,wsize=524288,nolock /mnt/isilon4
```

Note that since training is purely a read workload, it is safe to add the **nolock** parameter to the mounts that contain the input data. In some cases, this can improve performance.

If you are using the recommended dynamic IP allocation policy in the Isilon IP address pool, all IP addresses will remain accessible, even in the event of an Isilon node failure.

If you are expecting to perform more than 1000 MB/sec of I/O from a single GCE VM instance, you should ensure that the application uses multiple mount points to multiple Isilon nodes. Ideally, each VM instance system should use all four mount points evenly. This can be easily accomplished with a script like **round_robin_mpi.py** which uses the rank of the MPI process to select a mount point.

As a simple but less effective alternative, you may mount **/mnt/isilon1** to a different Isilon interface on each VM instance, but this may result in a suboptimal balance of traffic. Also, it has been observed that a single NFS mount can read about 2500 MB/sec which is only half of the 40 Gbps front-end Ethernet links on the Isilon nodes. Therefore, it is best to have at least two mounts per Isilon interface.

Note that different mounts do not share the Linux buffer cache. If your dataset is small enough to fit in the VM instance RAM, consider using fewer mounts to allow your entire dataset to be cached.

For all benchmarks presented in this document, the script **mount_isilon.py** in *AI Benchmark Util* was used to automate the drive mounts.

B Benchmark setup

B.1 Creating the ImageNet TFRecord datasets

To run the TensorFlow Benchmarks suite, the standard 148 GB ImageNet TFRecord dataset was created based on the documentation at https://github.com/tensorflow/models/tree/master/research/inception#getting-started.

To create the 1.0 TB dataset, consisting of seven copies of the above, the script **expand_tfrecords.sh** was used.

B.2 Obtain the TensorFlow benchmarks

The TensorFlow Benchmark suite can be obtained from the following Git repository.

```
cd /mnt/isilon/data
git clone https://github.com/claudiofahey/benchmarks tensorflow-benchmarks
cd tensorflow-benchmarks
git checkout 31ea13f
```

Note that the commit above differs from the official repository in only one significant way. It removes an unnecessary file name wildcard globbing step which has a huge performance impact when the number of TFRecord files exceeds 10,000. See https://github.com/claudiofahey/benchmarks/commit/31ea13f.

B.3 Start TensorFlow containers

In a basic bare-metal deployment of TensorFlow and MPI, all software must be installed on each node. MPI then uses SSH to connect to each node to start the TensorFlow application processes.

In the world of Docker containers, this becomes a bit more complex but significantly easier to manage dependencies with. On each GCE VM Instance, a single Docker container is launched which has an SSH daemon that listens on the custom port 2222. This Docker container also has TensorFlow, OpenMPI and NVIDIA libraries and tools. We can then run the **docker exec** and the **mpirun** command on one of these containers and MPI will connect to the Docker containers on all other VM instances by SSH on port 2222.

First, a custom Docker image is created using the following **Dockerfile**.

```
FROM nvcr.io/nvidia/tensorflow:19.09-py3

MAINTAINER Firstname Lastname <Firstname.Lastname@email.com>

# Install SSH and various utilities.
RUN apt-get update && apt-get install -y --no-install-recommends \
        openssh-client \
        openssh-server \
        lsof \
    && \
    rm -rf /var/lib/apt/lists/*
```

```
# Configure SSHD for MPI.
RUN mkdir -p /var/run/sshd && \
    mkdir -p /root/.ssh && \
    echo "StrictHostKeyChecking no" >> /etc/ssh/ssh_config && \
    echo "UserKnownHostsFile /dev/null" >> /etc/ssh/ssh_config && \
    sed -i 's/^#*Port 22/Port 2222/' /etc/ssh/sshd_config && \
    echo "HOST *" >> /root/.ssh/config && \
    echo "PORT 2222" >> /root/.ssh/config && \
    mkdir -p /root/.ssh && \
    ssh-keygen -t rsa -b 4096 -f /root/.ssh/id_rsa -N "" && \
    cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys && \
    chmod 700 /root/.ssh && \
    chmod 600 /root/.ssh/*

# Install Python libraries.
COPY requirements.txt /tmp/requirements.txt
RUN pip install --requirement /tmp/requirements.txt

WORKDIR /scripts

EXPOSE 2222
```

As you can see, this **Dockerfile** is based on the NVIDIA GPU Cloud (NGC) TensorFlow image.

Run the following command to build the Docker image. Replace user with your NGC ID, Docker ID, or **host:port** if you are using an on-premises container registry.

```
docker build -t user/tensorflow:19.09-py3-custom .
```

Note that during the build process, a new RSA key pair is randomly generated and stored in the image. This key pair allows containers running this image to SSH into each other. Although this is convenient for a lab environment, a production environment should never store private keys in an image.

Next, you must push this image to a Docker container registry so that it can be pulled from all other VM instances. Once logged in to your container registry, run the following command to upload the container.

```
docker push user/tensorflow:19.09-py3-custom
```

You are now ready to start the containers on all VM instances. Repeat this command for each VM instance, replacing host with the server name.

```
ssh host \
docker \
run \
--rm \
--detach \
--privileged \
--gpus all \
-v /mnt:/mnt \
--network=host \
--shm-size=1g \
```

```
--ulimit memlock=-1 \
--ulimit stack=67108864 \
--name tf \
user/tensorflow:19.09-py3-custom \
bash -c \
"/usr/sbin/sshd ; sleep infinity"
```

The final line starts the SSH daemon and waits forever. At this point, the container can be accessed by MPI by the SSH daemon listening on port 2222.

Choose any one of the VM instances as the master and enter the container by running the following command. This will give you a bash prompt within the container.

```
docker exec -it tf bash
```

Confirm that this container can connect to all other containers by password-less SSH on port 2222.

```
ssh dl-worker-002 hostname
ssh dl-worker-002 hostname
```

Next, test that MPI can launch processes across all VM instances.

```
mpirun --allow-run-as-root -np 2 -H dl-worker-001 -H dl-worker-002 hostname
```

To stop the containers and all processes within them, run the following command on each VM instance system

```
docker stop tf
```

The script **start_containers.sh** automates some of these steps.

C Benchmark details

The command below was used to perform the ResNet-50 training with 120 GPUs.

```
mpirun \
--n 120 \
-allow-run-as-root \
--host dl-worker-001:4,dl-worker-002:4,...,dl-worker-030:4 \
--report-bindings \
-bind-to none \
-map-by slot \
-x LD_LIBRARY_PATH \
-x PATH \
-mca plm_rsh_agent ssh \
-mca plm_rsh_args "-p 2222" \
-mca pml ob1 \
-mca btl_tcp_if_include eth0 \
-x NCCL_DEBUG=INFO \
-x NCCL_IB_DISABLE=1 \
./round_robin_mpi.py \
python \
-u \
/mnt/isilon/data/tensorflow-benchmarks/scripts/tf_cnn_benchmarks/\
tf_cnn_benchmarks.py \
--model=resnet50 \
--batch_size=64 \
--batch_group_size=10 \
--num_batches=500 \
--nodistortions \
--num_gpus=1 \
--device=gpu \
--force_gpu_compatible=True \
--data_format=NCHW \
--use_fp16=True \
--use_tf_layers=True \
--data_name=imagenet \
--use_datasets=True \
--num_intra_threads=1 \
--num_inter_threads=40 \
--datasets_prefetch_buffer_size=40 \
--datasets_num_private_threads=4 \
--train_dir=/mnt/isilon/data/train_dir/2019-10-24-14-53-59-resnet50 \
--sync_on_finish=True \
--summary_verbosity=1 \
--save_summaries_steps=100 \
--save_model_secs=600 \
--variable_update=horovod \
--horovod_device=cpu \
--data_dir=/mnt/isilon1/data/imagenet-scratch/tfrecords-7x \
--data_dir=/mnt/isilon2/data/imagenet-scratch/tfrecords-7x \
--data_dir=/mnt/isilon3/data/imagenet-scratch/tfrecords-7x \
--data_dir=/mnt/isilon4/data/imagenet-scratch/tfrecords-7x
```

For the other models, only the **--model** parameter was changed. Each result is the average of three executions.

D Isilon performance testing with iPerf and FIO

iPerf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It can be used to validate the throughput of the IP network path from an Isilon node to a compute node NIC. It can easily be scripted to run concurrently to allow all nodes to send or receive traffic.

FIO is a disk benchmark tool for Linux. It can be used to easily and quickly produce a storage workload that is similar to the TensorFlow benchmark used in this document.

This section shows how to use iPerf and FIO.

To begin, on any GCE VM instance, create a file named **hosts** containing one host name or IP address per GCE VM instance. For example:

```
dl-worker-001
dl-worker-002
dl-worker-003
dl-worker-004
```

Note: In the commands in this section, you should replace host names with IP addresses to avoid name resolution problems.

D.1 iPerf

D.1.1 Using iPerf to test Isilon to GCE VM instance performance (read)

1. Install iPerf on all GCE VM instances. Note that iPerf is already installed on all Isilon nodes. All versions should match.

   ```
   cat hosts | xargs -i -P 0 ssh {} sudo apt-get -y install iperf
   ```

2. Start iPerf server on each GCE VM instance.

   ```
   cat hosts | xargs -i ssh root@{} pkill -9 iperf
   cat hosts | xargs -i ssh root@{} "iperf --server --daemon \
   /dev/null 2>&1 &"
   ```

3. Start iPerf client on each Isilon node.

   ```
   client_opts="-t 60 --len 65536 --parallel 32"
   ssh -J root@isilon-ssh root@isilon-1 iperf -c dl-worker-001 ${client_opts}
   &
   ssh -J root@isilon-ssh root@isilon-2 iperf -c dl-worker-002 ${client_opts}
   &
   ssh -J root@isilon-ssh root@isilon-3 iperf -c dl-worker-003 ${client_opts}
   &
   ssh -J root@isilon-ssh root@isilon-4 iperf -c dl-worker-004 ${client_opts}
   &
   wait
   ```

4. Record the sum of the throughputs measured by each iPerf client.

D.1.2 Using iPerf to test GCE VM instance to Isilon performance (write)

1. Start iPerf server on each Isilon node.

```
cat hosts-isilon | xargs -i ssh -J root@isilon-ssh root@{} pkill -9 iperf
cat hosts-isilon | xargs -i ssh -J root@isilon-ssh root@{} "iperf \
--server --daemon /dev/null 2>&1 &"
```

2. Start iPerf client on each Isilon node.

```
client_opts="-t 60 --len 65536 --parallel 16"
ssh dl-worker-001 iperf -c isilon-1 ${client_opts} &
ssh dl-worker-002 iperf -c isilon-2 ${client_opts} &
ssh dl-worker-003 iperf -c isilon-3 ${client_opts} &
ssh dl-worker-004 iperf -c isilon-4 ${client_opts} &
wait
```

3. Record the sum of the throughputs measured by each iPerf client.

D.2 FIO

The procedure below shows how to use FIO to benchmark NFS I/O from multiple clients concurrently.

1. Install FIO servers.

```
cat hosts | xargs -i -P 0 ssh {} sudo apt-get install fio
```

2. Start FIO servers.

```
cat hosts | xargs -i ssh {} pkill fio
cat hosts | xargs -i ssh {} fio --server --daemonize=/tmp/fio.pid
```

3. Create a FIO job file shown below, named **fio1.job**. The job file below was used for reading from 18 worker nodes. Each worker node (FIO client) created two files and read them sequentially concurrently.

```
[global]
create_serialize=0
directory=/mnt/isilon1/tmp/fio:/mnt/isilon2/tmp/fio
ioengine=sync
kb_base=1000
name=job1
numjobs=2
ramp_time=30
time_based=0

[job1]
bs=1024KiB
direct=0
end_fsync=1
```

```
fallocate=none
fsync_on_close=1
loops=100
nrfiles=1
randrepeat=0
rw=read
size=250GB
sync=0
```

4. Run the FIO job.

```
mkdir -p /mnt/isilon/tmp/fio
fio --client=hosts fio1.job
```

E Monitoring Isilon performance

E.1 Isilon statistics CLI

For a quick way to investigate the performance of an Isilon cluster when InsightIQ is not available, there is a wealth of statistics that are available through the Isilon CLI, which can be accessed using SSH to any Isilon node.

This first command shows the highest level of statistics. Note that units are in bytes/sec so in the example below Node 1 is sending (NetOut) 2.9 GB/sec to clients.

```
isilon-1# isi statistics system --nodes all --format top
Node   CPU    SMB FTP HTTP    NFS HDFS  Total   NetIn NetOut DiskIn DiskOut
 All 72.7%   0.0 0.0  0.0  10.0G  0.0  10.0G  304.4M  10.4G 315.4M   10.8G
   1 79.2%   0.0 0.0  0.0   2.9G  0.0   2.9G  295.0M   2.9G  70.5M    2.3G
   2 80.6%   0.0 0.0  0.0   2.7G  0.0   2.7G    3.2M   2.7G  95.5M    2.8G
   3 71.9%   0.0 0.0  0.0   2.4G  0.0   2.4G    3.4M   2.6G  75.5M    2.8G
   4 59.2%   0.0 0.0  0.0   2.0G  0.0   2.0G    2.9M   2.1G  73.9M    2.9G
```

The following command shows more details related to NFS. All statistics are aggregated over all nodes.

```
isilon-1 # isi statistics pstat --format top
_____________________________NFS3 Operations Per Second_____________________________
access              2.33/s  commit              0.00/s  create              0.75/s
fsinfo              0.00/s  getattr             2.09/s  link                0.00/s
lookup              0.99/s  mkdir               0.00/s  mknod               0.00/s
noop                0.00/s  null                0.00/s  pathconf            0.00/s
read            18865.24/s  readdir             0.00/s  readdirplus         0.00/s
readlink            0.00/s  remove              0.00/s  rename              0.75/s
rmdir               0.00/s  setattr             0.00/s  statfs              0.00/s
symlink             0.00/s  write               0.75/s
Total           18872.91/s

___CPU Utilization___                                          _____OneFS Stats_____
user               1.4%                                        In          73.81 kB/s
system            72.5%                                        Out          8.96 GB/s
idle              26.1%                                        Total        8.96 GB/s

____Network Input____          ___Network Output____          ______Disk I/O_______
MB/s              12.64        MB/s            9868.11        Disk    272334.03 iops
Pkt/s         150368.20        Pkt/s       6787182.27         Read        11.73 GB/s
Errors/s          0.00         Errors/s          0.00         Write       99.63 MB/s
```

F Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical documents and videos provide expertise that helps to ensure customer success on Dell EMC storage platforms.

F.1 Additional resources

Name	Link
AI Benchmark Utilities	https://github.com/claudiofahey/ai-benchmark-util/tree/gcp
Deep Learning with Dell EMC Isilon	https://www.dellemc.com/resources/en-us/asset/white-papers/products/storage/h17361_wp_deep_learning_and_dell_emc_isilon.pdf
Dell EMC Isilon and Dell EMC DSS 8440 Servers for Deep Learning	https://www.dellemc.com/resources/en-us/asset/white-papers/products/storage/h17843_wp_dell_emc_isilon_and_dss_8440_servers_for_deep_learning.pdf
Dell EMC Isilon H500	https://www.dellemc.com/en-ca/collaterals/unauth/data-sheets/products/storage/h16071-ss-isilon-hybrid.pdf
Dell EMC Isilon OneFS Best Practices	https://www.emc.com/collateral/white-papers/h16857-wp-onefs-best-practices.pdf
Dell EMC Isilon OneFS SmartFlash	https://www.dellemc.com/resources/en-us/asset/white-papers/products/storage/h13249-isilon-onefs-smartflash-wp.pdf
Dell EMC Isilon OneFS Technical Overview	https://www.dellemc.com/en-tz/collaterals/unauth/technical-guides-support-information/products/storage/h10719-isilon-onefs-technical-overview-wp.pdf
Dell EMC Isilon Storage Tiering	https://www.dellemc.com/resources/en-us/asset/white-papers/products/storage/h8321-wp-smartpools-storage-tiering.pdf
Dell EMC Ready Solutions for AI, Machine and Deep Learning	https://www.dellemc.com/content/dam/uwaem/production-design-assets/en-gb/solutions/assets/pdf/dell-emc-ready-solutions-for-ai-and-dl.pdf
Gartner	https://www.gartner.com/en/newsroom/press-releases/2019-08-05-gartner-says-ai-augmentation-will-create-2point9-trillion-of-business-value-in-2021
ImageNet	http://www.image-net.org/challenges/LSVRC/2012/
TensorFlow	https://github.com/tensorflow/tensorflow
TensorFlow Benchmarks	https://github.com/tensorflow/benchmarks