TECHNICAL WHITE PAPER

REAL-WORLD AI

DEEP LEARNING PIPELINE POWERED BY FLASHBLADE

PURESTORAGE®

TABLE OF CONTENTS

INTRODUCTION

Advances in deep neural networks have ignited a new wave of algorithms and tools for data scientists to tap into their data with artificial intelligence (AI). With improved algorithms, larger data sets, and frameworks such as TensorFlow, data scientists are tackling new use cases like autonomous driving vehicles and natural language processing.

Training deep neural networks requires both high quality input data and large amounts of computation. GPUs are massively parallel processors capable of operating on large amounts of data simultaneously. When combined into a multi-GPU cluster, a high throughput pipeline is required to feed input data from storage to the compute engines. Deep learning is more than just constructing and training models. There also exists an entire data pipeline that must be designed for the scale, iteration, and experimentation necessary for a data science team to succeed.

> This document describes the technical reasons for and benefits of an end-to-end training system and why the Pure Storage® FlashBlade™ product is an essential platform. It also shows performance benchmarks based on a system that combines the NVIDIA® DGX-1™, a multi-GPU server purpose-built for deep learning applications, and Pure FlashBlade, a scale out, high-performance, dynamic data hub for the entire AI data pipeline.

LIFECYCLE OF DATA

Data is the heart of modern AI and deep learning algorithms. Before training can begin, the hard problem is collecting the labeled data that is crucial for training an accurate AI model. A full scale AI deployment must continuously collect, clean, transform, label, and store large amounts of data. Adding additional high quality data points directly translates to more accurate models and better insights.

Data samples undergo a series of processing steps:

- **INGEST** the data from an external source into the training system. Each data point is often a file or object. Inference may also run on this data. After the ingest step, the data is stored in raw form and is often also backed up in this raw form. Any associated labels (ground truth) may come in with the data or in a separate ingest stream.
- **CLEAN AND TRANSFORM** the data and save in a format convenient for training, including linking the data sample and associated label. This second copy of the data is not backed up because it can be recomputed if needed.
- **EXPLORE** parameters and models, quickly test with a smaller dataset, and iterate to converge on the most promising models to push into the production cluster.
- **TRAINING** phases select random batches of input data, including both new and older samples, and feed those into production GPU servers for computation to update model parameters.
- **EVALUATION** uses a holdback portion of the data not used in training in order to evaluate model accuracy on the holdout data.

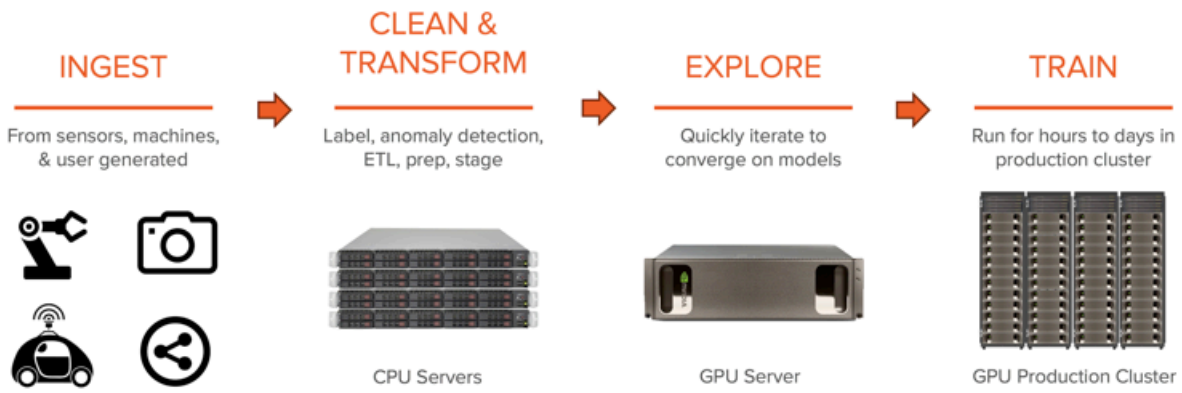

FIGURE 1: AI data sample processing steps

This lifecycle applies for any type of parallelized machine learning, not just neural networks or deep learning. For example, standard machine learning frameworks, such as Spark MLlib, rely on CPUs instead of GPUs, but the data ingest and training workflows are the same.

> A single shared storage data hub creates a coordination point throughout the lifecycle without the need for extra data copies among the ingest, preprocessing, and training stages. Rarely is the ingested data used for only one purpose, and shared storage gives the flexibility to train multiple different models or apply traditional analytics to the data.

Access Pattern	sequential	sequential or random	random	random
Access Type	write	read or write	read	read
File Size	metadata is small data is small to large	small to large	small to large	small to large
Concurrency	depends on # of sources	high	low	high

FIGURE 2: Storage requirements in the AI data pipeline

As seen above, each stage in the AI data pipeline has varying requirements from the data hub. Scale-out storage systems must deliver uncompromising performance for all manner of access types and patterns – from small, metadata-heavy to large files, from random- to sequential-access patterns, and from low to high concurrency. For any legacy storage systems, this is an impossible design point to meet. By contrast, FlashBlade is the ideal AI data hub, as it was purpose-built from the ground up for modern, unstructured workloads.

In the first stage, data is ideally ingested and stored on to the same data hub that following stages will use, in order to avoid excess data copying. The next two steps can be done on a standard compute server that optionally includes a GPU, and then in the fourth and last stage, full training production jobs run on powerful GPU-accelerated servers like the DGX-1. Often, there is a production pipeline alongside an experimental pipeline operating on the same dataset. Further, the DGX-1 GPUs can be used independently for different models or joined together to train on one larger model, even spanning multiple DGX-1 systems for distributed training.

If the shared storage tier is slow, then data must be copied to local storage for each phase, resulting in wasted time staging data onto different servers. The ideal data hub for the AI training pipeline delivers performance similar to data stored locally on the server node while also having the simplicity and performance to enable all pipeline stages to operate concurrently.

THE DATA SCIENTIST WORKFLOW

A data scientist works to improve the usefulness of the trained model through a wide variety of approaches: more data, better data, smarter training, and deeper models. In many cases, there will be teams of data scientists sharing the same datasets and working in parallel to produce new and improved training models.

The day-to-day workflow of a data scientist includes:

- Collating, cleaning, filtering, processing, and transforming the training data into a form consumable by the model training.
- Experimenting with, testing, and debugging a model on a small subset of the training data.
- Training the model with the full set of training data for longer periods of time.

> This workflow is iterative between these stages: development, experimentation, and debugging. The key development tool is a deep-learning framework like TensorFlow, Caffe2, CNTK, et al. These frameworks provide utilities for processing data and building models that are optimized for execution on distributed GPU hardware.

Often, there is a team of data scientists working within these phases concurrently on the same shared datasets. Multiple, concurrent workloads of data processing, experimentation, and full-scale training layer the demands of multiple access patterns on the storage tier. In other words, storage cannot just satisfy large file reads, but must contend with a mix of large and small file reads and writes.

Finally, with multiple data scientists exploring datasets and models, it is critical to store data in its native format to provide flexibility for each user to transform, clean, and use the data in a unique way. Ultimately, it is these experiments that yield more powerful models.

FlashBlade provides a natural shared storage home for the dataset, with data protection redundancy (using RAID6) and the performance necessary to be a common access point for multiple developers and multiple experiments.

Using FlashBlade avoids the need to carefully copy subsets of the data for local work, saving both engineering and DGX-1 system use time. These copies become a constant and growing tax as the raw data set and desired transformations constantly update and change.

SCALABLE DATASETS

A fundamental reason why deep learning has seen a surge in success is the continued improvement of models with larger data set sizes. In contrast, classical machine learning algorithms, like logistic regression, stop improving in accuracy at smaller data set sizes.

A recent quote from a leading AI researcher highlights the need:

"As of 2016, a rough rule of thumb is that a supervised deep learning algorithm will generally achieve acceptable performance with around 5,000 labeled examples per category, and will match or exceed human performance when trained with a dataset containing at least 10 million labeled examples."
Ian Goodfellow 2016.

> Recent research from Google has shown the advantages of increasing dataset size, revealing a logarithmic increase in performance for vision tasks as data sets increase in size to 300 million images. Even further, this research suggests that higher capacity models require proportionally larger datasets.

Separation of compute (DGX-1) and storage (FlashBlade) also allows independent scaling of each tier, avoiding many of the complexities inherent in managing both together. As the data set size grows or new data sets are considered, a scale out storage system must be able to expand easily. Similarly, if more concurrent training is required, additional GPUs or DGX-1 servers can be added without concern for their internal storage.

WHY FLASHBLADE

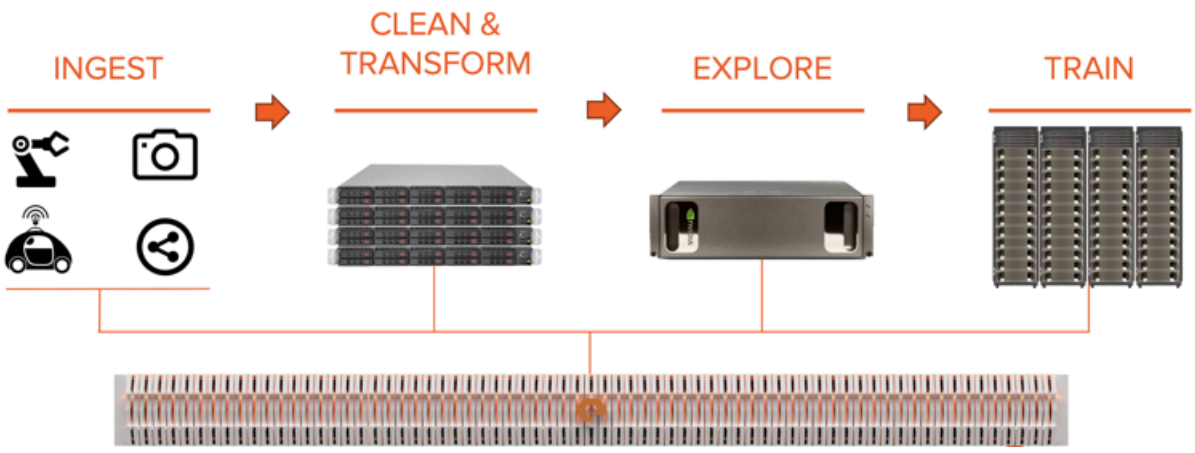

FIGURE 3: FlashBlade as a centralized hub for data

A centralized data hub in a deep learning architecture increases the productivity of data scientists and makes scaling and operating simpler. FlashBlade specifically makes building, operating, and growing an AI system easier for the following reasons.

- **PERFORMANCE:** With over 15GB/s of large-file read bandwidth per chassis and up to 75GB/s total, FlashBlade can support the concurrent requirements of an end-to-end AI workflow.
- **SMALL-FILE HANDLING:** The ability to randomly read small files (50KB) at 10GB/s from a single FlashBlade chassis (50GB/s with 75 blades) means that no extra effort is required to aggregate individual data points to make larger, storage-friendly files.
- **SCALABILITY:** Start off with a small system and then add a blade to increase capacity and performance as either the dataset grows or the throughput requirements grow.
- **NATIVE OBJECT SUPPORT (S3):** Input data can be stored as either files or objects.
- **SIMPLE ADMINISTRATION:** No need to tune performance for large or small files and no need to provision filesystems.
- **NON-DISRUPTIVE UPGRADE (NDU) EVERYTHING:** Software upgrades and hardware expansion can happen anytime, even during production model training.
- **EASE OF MANAGEMENT:** Pure1®, our cloud-based management and support platform, allows users to monitor storage from any device and provides predictive support to identify and fix issues before they become impactful. With Pure1, users can focus on understanding data and not on administering storage.
- **BUILT FOR THE FUTURE:** Purpose-built for flash, to easily leverage new generations of NAND technology – density, cost, and speed.

Small file performance of the storage tier is critical as many types of inputs, including text, audio, or images will be natively stored as small files. If the storage tier does not handle small files well, an extra step will be required to pre-process and group samples into larger files.

Storage, built on top of spinning disks, that relies on SSD as a caching tier, falls short of the performance needed. Because training with random input batches results in more accurate models, the entire data set must be accessible with full performance. SSD caches only provide high performance for a small subset of the data and will be ineffective at hiding the latency of spinning drives.

Ultimately, the performance and concurrency of FlashBlade means that a data scientist can quickly transition between phases of work without wasting time copying data. FlashBlade also enables the execution of multiple different experiments on the same data simultaneously.

SYSTEM ARCHITECTURE

This section describes the infrastructure and software components needed for a deep learning cluster.

Infrastructure Components

The physical infrastructure of a deep learning cluster with DGX-1 and FlashBlade includes the following hardware:

Primary Compute: DGX-1 server(s) with eight V100 GPUs and NVLink interconnected into a computation engine to train parameters for deep neural networks. Each system has both Ethernet and InfiniBand external connectivity. The GPUs can be grouped for a single large training or used independently to train multiple models.

Storage: FlashBlade provides a scale-out all-flash file or object store with usable capacity (for non-compressible data) that scales from 60TB to 2.5PB. Data can be accessed via high-performance NFS or S3 protocols.

Networking: Redundant top-of-rack ethernet switches connected to storage and compute via 10Gbps and 40Gbps ports in MLAG port channels for redundancy. DGX-1 systems are typically connected to each other through an InfiniBand fabric, though some customers may choose to use the same Ethernet switch fabric for both compute and storage traffic.

Additional Compute: Whitebox servers, optionally with GPUs, for data ingestion, pre-processing, and model debugging.

See the later section “Sizing Guide” on determining the relative ratio of FlashBlades to DGX-1s.

Software Pipeline

Training software needs to pipeline data from the storage system to the GPUs and ensure that the GPUs always have the next batch of training data available.

For an end-to-end model training session, the data flows as follows:

- **DECODE AND AUGMENT:** Load input files from storage media and convert to a form appropriate for training, e.g., a tensor. The decode step can be pre-computed or performed on the host CPU. Augmentation uses the host CPU to introduce dynamic changes in the input to avoid overfitting.
- **IO QUEUES:** Multiple threads read random batches of input records from storage and populate an internal queue. These threads run on the DGX-1 host CPUs and are responsible for pre-fetching batches and ensuring training records are available in DRAM.
- **TRAINING:** A second set of threads pulls data from internal queues to feed to the GPUs for computations (i.e., forward and backpropagation) needed for training and updating model parameters.

Each training batch requires retrieving the data from persistent storage, then decoding and augmenting that data. If the GPUs are idle – waiting for IO and the host CPU to fetch and decode the inputs – then the compute power of the GPUs are not efficiently utilized. The pipelined training flow below ensures that the training devices always have the

next input set available upon completion of a training batch. As long as the IO and host CPU portion is faster than the training computations, then GPUs will operate at maximum utilization.

Typical Training Pipeline

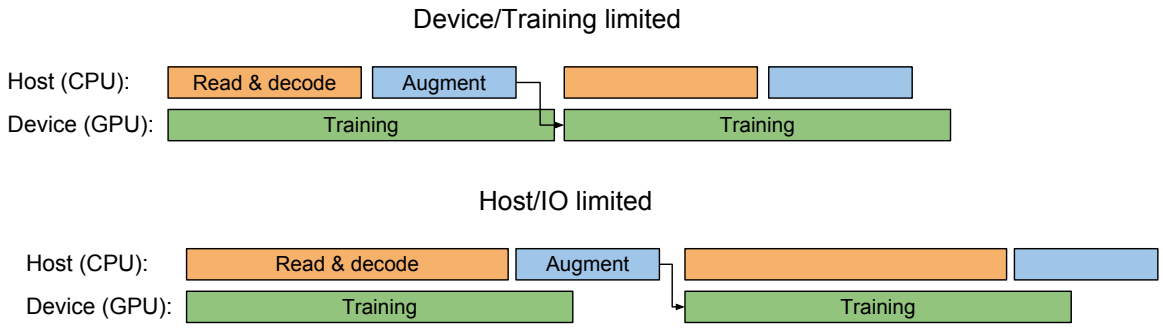

FIGURE 4: Flow of training data through software architecture

The diagram below illustrates the flow of training data through the software architecture to realize the pipelining described. The host CPUs on the DGX-1 orchestrate two sets of asynchronous threads. The fetch threads select random batches of input files and are responsible for reading the data into a queue as long as it is not full. The training threads run the model training, dequeuing the input data at each iteration. The point of coordination between these two processes is the in-memory queue. The fetch threads can also pre-process and transform the input data into the expected format. As long as IO and host CPU have sufficient throughput to ensure the queue never gets empty, the training threads can run at optimal performance.

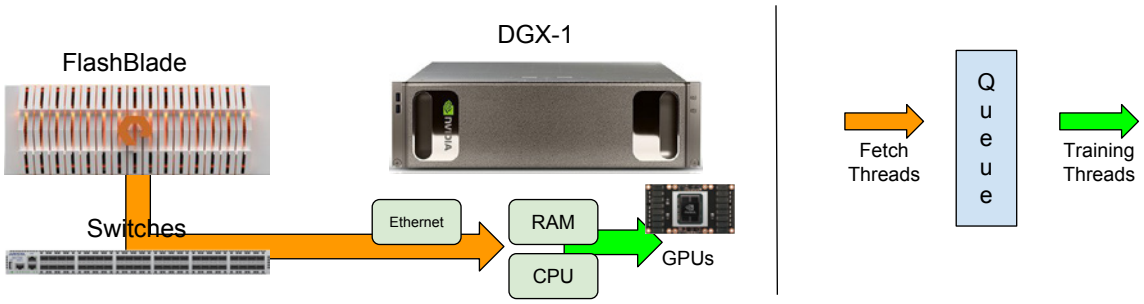

FIGURE 5: FlashBlade speed enables optimal training performance

TensorFlow documentation includes more detailed information on read pipelining.

Connectivity

The physical connectivity on FlashBlade is ethernet; the DGX-1 has both Infiniband (IB) and ethernet ports. FlashBlade connects to TOR switches via up to 8x 40Gbps ports in a single MLAG, presenting one logical link. All storage accesses are via these ethernet links and the NFS v3 protocol using the default mount options; the rsize and wsize are 512KB, and filesystem caching is not needed.

The IB ports on the DGX-1 can interconnect multiple DGX-1 servers in a low-latency network to scale-out training capacity, or alternatively they can be bridged to ethernet to increase the bandwidth to external storage. The internal NVLink connections interconnect multiple GPUs, and are used to coordinate updates to model parameters after each training iteration. For scaling to multiple DGX-1s, the IB ports can be used to connect multiple DGX-1 servers and coordinate multi-server training.

With most current models and GPUs, the two 10Gbps ethernet ports provide sufficient bandwidth to feed data into the GPUs. For any training that requires more than the bandwidth of a single 10Gbps link, there are two options to configure the DGX-1's ethernet interfaces:

- Bonding the links together and providing a single logical 20Gbps connection to the switches and FlashBlade.
- Assigning each separate interface to a different subnet and connecting to two different FlashBlade data VIPs (virtual IP addresses) on the respective subnets.

> In both cases, the goal is to fully utilize the ethernet bandwidth by multiplexing the reads of input data across both links.

Each training session connects to – and creates a TCP connection to – a single mount point. With either bonding or separate interfaces, a TCP connection will utilize only one of the two links. We utilize multiple data VIPs on FlashBlade to mount the same file system at multiple mount points; because each mount connects to a different IP endpoint, the kernel creates a different TCP connection for each.

TRAINING BENCHMARK RESULTS

The following section presents benchmark results from training operations on a full FlashBlade and DGX-1 system. We measure the performance of popular ImageNet models to verify that **the data read directly from FlashBlade saturates the GPUs**.

Test Setup

The test set of training tests was performed with the DGX-1 and FlashBlade hardware as described in the system architecture. The DGX-1 has eight V100 GPUs, 2 host CPUs (80 cores total), and 512GB of system memory; the training data is stored on a 15-blade FlashBlade. All tests were run using the NVIDIA-supplied container: nvcr.io/nvidia/tensorflow v17.12.

The tests used the Imagenet 2012 dataset – one of the most influential in deep learning research – as the input for training. The dataset consists of 1.28 million images, 143 GB in total size. Input data was labeled jpeg images packed into larger files (~135MB each). Filesystem caching (fsc option) on the NFS mount was turned off. We followed common best practices for TensorFlow performance optimization and trained each model until we reached a steady state of images/sec processed.

While ImageNet serves an important purpose in the deep learning community, the file sizes and formats in ImageNet do not reflect real-world data requirements. ImageNet data is preprocessed and packed into large files, but this is contrary to the desire to have native formats (> 1 MB) at full fidelity in order to rapidly prototype new techniques to take advantage of this data.

For benchmarks, we compare three different input sources for the training data:

- Synthetic data is artificially created data in system memory, meaning that there is no storage IO involved or image decoding from jpeg format. This configuration tests to find the limits of the GPU processing to help understand if the GPUs are the bottleneck.
- Local 4x SSD stores the training set of data on the local SSDs inside the DGX-1 box. In this scenario, the local SSDs are used as a cache for the dataset, which would need to be staged on the DGX-1 from a different external storage repository. This approach does not scale beyond the local capacity (8TB currently).
- FlashBlade with 15 Blades as source of training data. Data is read from FlashBlade using multiple TCP connections. FlashBlade can scale up to 2.5PB, with performance consistent across the entire data set.

> The goal of the benchmark testing is to verify that the latency to external storage does not result in a bottleneck.

In order to utilize both 10Gbps links on the DGX-1 to connect to FlashBlade, it was necessary to access the input data using multiple TCP connections. We mounted the FlashBlade filesystem on two different data VIPs and multiplexed file read accesses across both mount points during the training phase.

Results

A baseline was created by comparing the read throughput of two storage options: internal SSDs and FlashBlade. The read throughput numbers of random data measured from the DGX-1:

- Local 4x RAID-0 SSD: 2.09 GB/s
- FlashBlade: 2.14 GB/s per 2x 10Gbps link

Using both 10Gbps interfaces, the read bandwidth was equivalent for local storage and FlashBlade. This 2GB/s read performance is an upper bound on the IO speed of either config, leaving additional performance available for other compute nodes, e.g., data ingest, experiments, and model debugging.

TensorFlow Benchmark Results

TensorFlow benchmarks are run for multiple iterations in order to get a stable measurement of the number of images processed per second in training. Each benchmark is run with varying numbers of GPUs in data parallel mode with SGD optimization.

The higher rates of images/sec – vertical axis in the charts below – indicate higher processing rates and throughput demands. Rates are measured in images/sec instead of throughput because each model must have a fixed size input regardless of the native input image size. Each pipeline includes a resize step before the data is sent to the GPUs; larger input files increase the load on storage but not on the GPU.

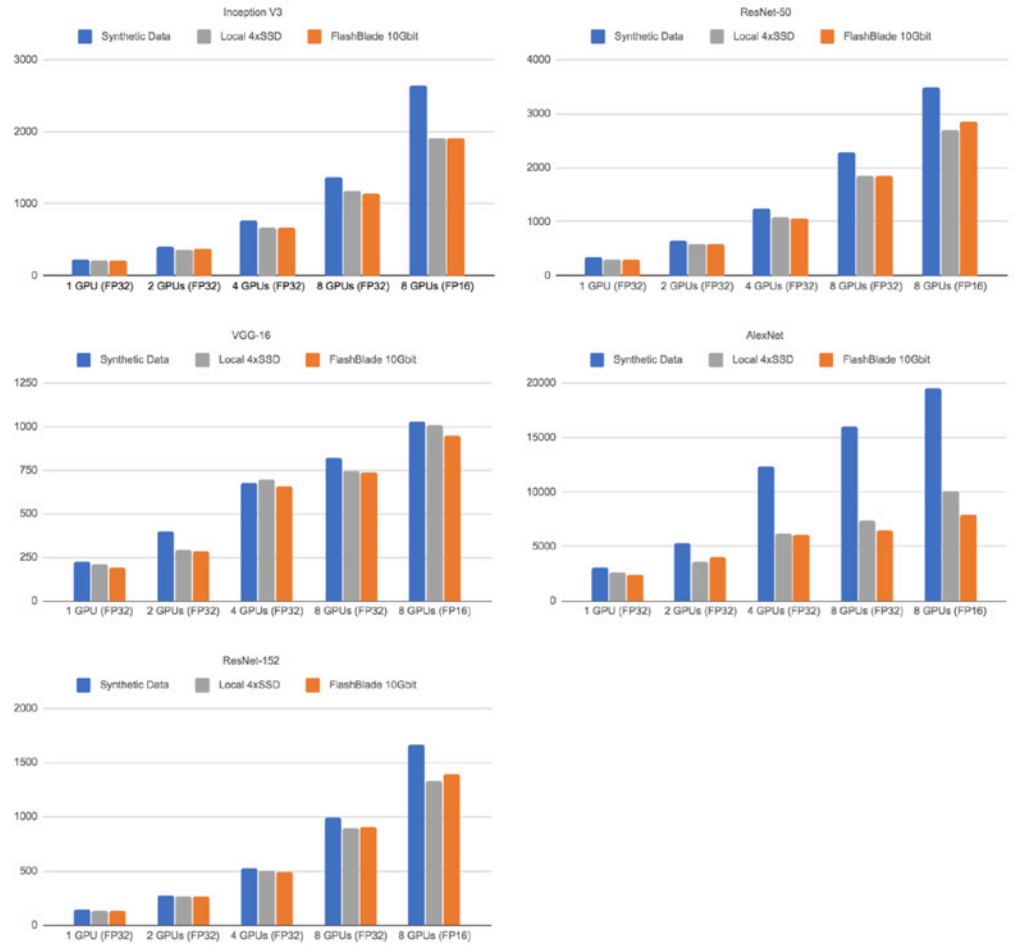

FIGURE 6: TensorFlow benchmark results by model. AlexNet is CPU bound due to image transformation. All other models are GPU bound.

> The more complex models – Inception-v3, ResNet-50, Resnet-152, and VGG-16 – were all GPU bottlenecked. This is evidenced by synthetic data and FlashBlade training performance being nearly equivalent, meaning that the in-memory queue was kept sufficiently full to satisfy throughput demands of the GPUs. The synthetic workload was slightly faster than local SSD and FlashBlade, due to the image transformations on the CPU that are a small part of the critical path.

Conversely, AlexNet on the DGX-1 was CPU bound due to the image transformation performed before training. Using synthetic generated data, the training processed 20k images/sec, whereas either IO source was maximized at around 9200 images/sec, or approximately 1.2GB/s. This speed was below the storage's measured read performance, suggesting that image file processing was the bottleneck instead of external storage IO.

Performance Sizing Considerations

Appropriate FlashBlade sizing depends on 1) the model complexity being trained; 2) the native sizes of input data; and 3) the number of GPUs operating concurrently. The third factor directly relates to the number of data scientists on the team using the cluster resources. Each of these factors influences the performance required and therefore the number of blades needed to support deep learning in addition to the other pipeline phases.

- More complex models require more computation for each training example, and consequently require less read throughput compared to simpler models.
- Larger file sizes result in more read throughput from the storage tier for each input batch. Note that this is independent of model complexity; images will be resized before training.
- Additional GPU servers increase load by allowing multiple independent models to be trained simultaneously or a single model training to scale-out. Faster GPUs in the future will also increase demand for data.

> Using the native input sizes provides the most flexibility for the data scientist to transform and process the data into the most useful form.

A key point is the relationship between input sizes and model complexity. For example, ImageNet models like ResNet-50 scale the inputs to a 225x225 image. Preprocessing all images to this size before training reduces throughput but enforces rigid assumptions about how the data will be used. Creating larger models to utilize larger input sizes does not work unless the number of inputs also scales; if there are more model parameters than input points, the model will simply memorize the training data and not truly learn.

To provide sizing guidance, the table below presents the most well-known ImageNet models, which vary widely in size and complexity. The measured throughput with 8-GPU training uses native input sizes of 150KB, and an extrapolated throughput demand is also included for 1MB-sized images. To size the number of DGX-1 systems that can be fed from FlashBlade, we assume that 50% of the available throughput goes to training while the other 50% is used for ingest, transformations, and experiments.

IMAGENET MODEL	DGX-1 READ THROUGHPUT		# OF DGX-1 SYSTEMS PER FLASHBLADE	
	150KB IMAGE SIZES, FROM IMAGENET	1MB IMAGE SIZES, EXTRAPOLATED	7-BLADE SYSTEM, USING 1MB IMAGES	15-BLADE SYSTEM, USING 1MB IMAGES
INCEPTION V3	290 MB/S	1.90 GB/S	2	4
RESNET-50	430 MB/S	2.90 GB/S	1	2
RESNET-152	210 MB/S	1.40 GB/S	2	5
ALEXNET	1200 MB/S	8.00 GB/S	–	1
VGG16	140 MB/S	1.00 GB/S	3	7

The performance achievable by FlashBlade depends on the size of the input files.

- For small files (50KB), a 7-blade FlashBlade system can do roughly 5 GB/s of read throughput on random data, and a 15-blade system can exceed 10GB/s.
- For larger files (1MB+), the read performance is slightly above 1GB/s per blade, i.e., 15GB/s on a 15-blade system.

For the ResNet-50 model with images kept in native format at 1MB per file, the required performance is 2.9 GB/s; two concurrent trainings would take approximately 6 GB/s of read throughput, leaving additional headroom for data ingest, data processing, model debugging, and testing.

In the future, more powerful GPUs will be released which can accelerate the training rate, increasing the demand on storage system performance. More powerful GPUs also lead to deeper neural networks, which increase the amount of computation per input batch. The increase in performance necessary will be somewhat offset by deeper models.

Alternative: Staging data to local storage (SSDs)

For small datasets, an alternative approach to high-performance shared storage is to stage data to the local SSDs on the DGX-1. Any time the training set needs to change, the user must stage the new data. For datasets larger than local storage, this approach is not feasible.

As an example, staging 1 TB of data consisting of 200 kB images to DGX-1 SSD drives takes 1.5 - 2.5 hours. This staging represents time the GPUs on the DGX-1 are unused. When measuring total time to solution, shared storage can deliver significantly faster end-to-end performance.

APPENDIX: REAL-WORLD SYSTEM ARCHITECTURE

The full system architecture needs to store the rich data sets required for deep learning and connect those to the GPUs for model training. The diagram below is a real-world architecture from a currently deployed deep learning system for an autonomous driving use case.

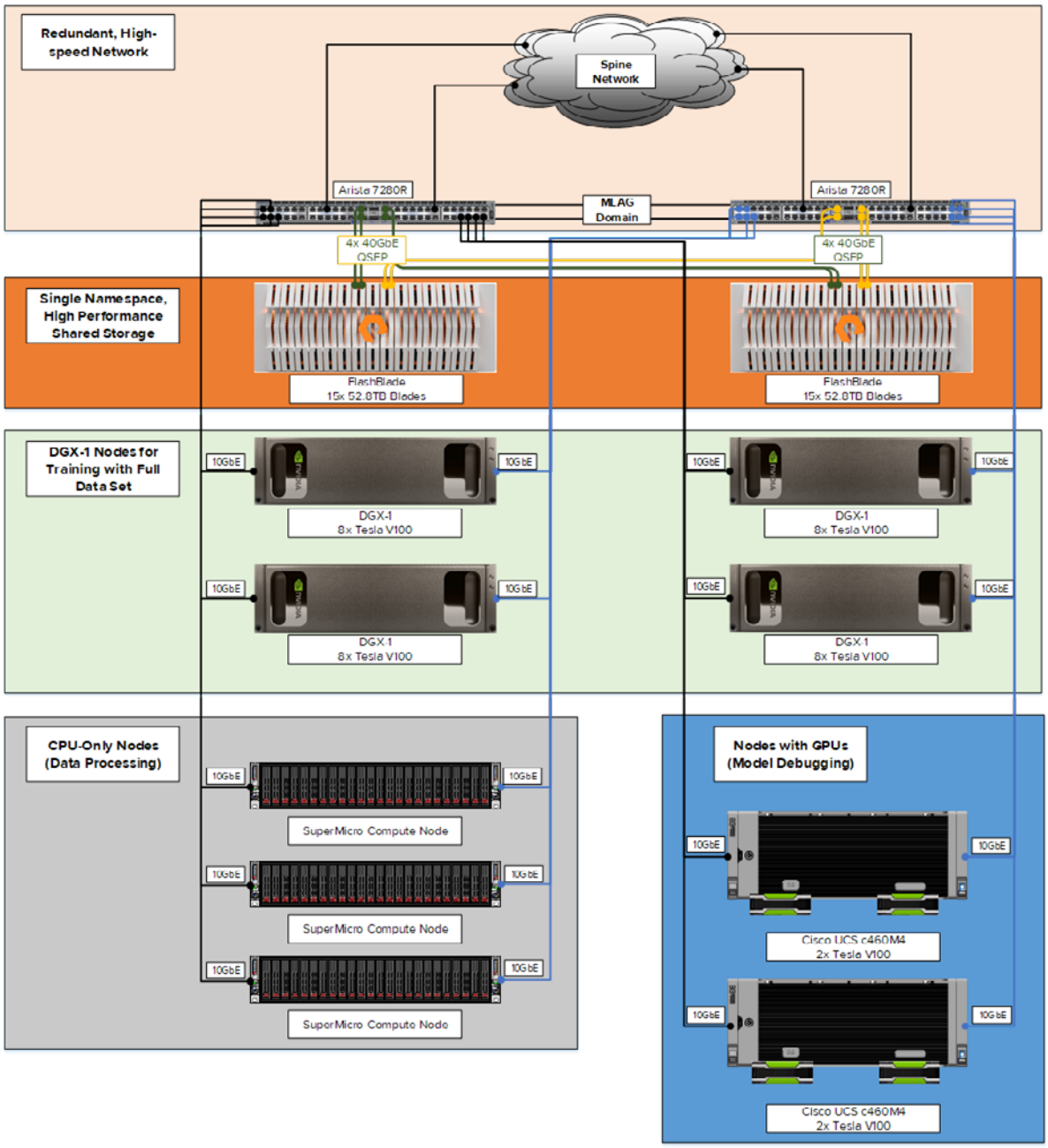

FIGURE 7: Example of a full system architecture

Physical Infrastructure

The topology in Figure 7 includes the following elements for the complete infrastructure.

Primary Compute: DGX-1 (3U)

- 8 Nvidia V100 GPUs
- L2 direct connectivity from FlashBlade to DGX-1 via switched ethernet
- Network configured on the same subnet as FlashBlade data VIPs

Storage: FlashBlade (4U)

- 7-15 blades depending on performance requirements
- 17TB or 52TB blades depending on data set size
- 17TB blades have 3x the performance of 52TB blades measured in MB/s per TB usable
- Configured with multiple data VIPs and a single file system

Additional Compute Servers

- Data ingest pipeline that preprocesses, cleans, and normalizes new data
- Small scale validation with experimental NN models
- Optional GPU-enabled servers for debugging model training performance without requiring time on a DGX-1

Networking: TOR Switch Pair in MLAG

- Each DGX-1 in a separate port channel, each interface connected to a different TOR switch for reliability
- Example wiring for two DGX-1 servers (DGX-1-1 and DGX-1-2):
  - DGX-1-1.port1 -> switch1
  - DGX-1-1.port2 -> switch2
  - DGX-1-2.port1 -> switch1
  - DGX-1-2.port2 -> switch2
- FlashBlade in its own port channel connected to both TOR switches
  - Connectivity: FM1 -> switch1, FM1 -> switch2, FM2 -> switch1, FM2 -> switch2

FlashBlade Configuration

A Single Filesystem for All Training Data

PUREUSER@SN1-FB-E02-33-1:~$ PUREFS LIST							
NAME	SIZE	USED	% USED	CREATED	PROTOCOLS	RULES	FAST REMOVE
NFS0	1T	138.40G	14%	2017-08-03 14:08:04 PDT	NFS	*(RW,NO_ ROOT_ SQUASH)	FALSE

Multiple Data VIPs for Accessing FlashBlade

PUREUSER@SN1-FB-E02-33-1:~$ PURENETWORK LIST							
NAME	ENABLED	SUBNET	ADDRESS	VLAN MASK	GATEWAY	MTU	SERVICES
NFS	TRUE	NET3	10.21.115.9	2115	10.21.115.1	1500	DATA
NFS0	TRUE	NET3	10.21.115.6	2115	10.21.115.1	1500	DATA
NFS1	TRUE	NET3	10.21.115.7	2115	10.21.115.1	1500	DATA
NFS2	TRUE	NET3	10.21.115.8	2115	10.21.115.1	1500	DATA
VIR0	TRUE	NET2	10.21.112.113	2112	10.21.112.1	1500	MANAGEMENT

DGX-1 Configuration

Mounts for FlashBlade

```
pureuser@sn1-dgx-1-e02-37:~/tensorflow-benchmarks$ cat /etc/mtab
/dev/sda2 / ext4 rw,errors=remount-ro 0 0
proc /proc proc rw,noexec,nosuid,nodev 0 0
sysfs /sys sysfs rw,noexec,nosuid,nodev 0 0
.....
/dev/sda1 /boot/efi vfat rw 0 0
/dev/sdb1 /raid ext4 rw 0 0
rpc_pipefs /run/rpc_pipefs rpc_pipefs rw 0 0
systemd /sys/fs/cgroup/systemd cgroup rw,noexec,nosuid,nodev,none,name=systemd 0 0
10.21.115.6:/nfs0 /mnt/nfs0 nfs rw,addr=10.21.115.6,_netdev 0 0
10.21.115.7:/nfs0 /mnt/nfs1 nfs rw,addr=10.21.115.7,_netdev 0 0
10.21.115.8:/nfs0 /mnt/nfs2 nfs rw,addr=10.21.115.8,_netdev 0 0
10.21.115.9:/nfs0 /mnt/nfs3 nfs rw,addr=10.21.115.9,_netdev 0 0
```

Routing Table to Split the Mount Point Traffic Across Both Interfaces

```
pureuser@sn1-dgx-1-e02-37:~/tensorflow-benchmarks$ ip route
default via 10.21.115.1 dev em1
10.21.115.0/24 dev em1  proto kernel  scope link  src 10.21.115.120
10.21.115.6 dev em1  proto kernel  scope link  src 10.21.115.120
10.21.115.7 dev em2  proto kernel  scope link  src 10.21.115.113
10.21.115.8 dev em1  proto kernel  scope link  src 10.21.115.120
10.21.115.9 dev em2  proto kernel  scope link  src 10.21.115.113
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1
```


ps_wp18p_AI-reference-architecture_ltr_02