Demystifying AI IO Performance

Silverton Consulting, Inc. StorInt™ Briefing

Silverton Consulting
Strategy, Storage & Systems

Introduction

Hardware and software technologies come and go, but workloads don't change much aside from getting faster – that is, until AI deep learning (DL) caught hold.

If there's one new workload that is transforming enterprise IT, it's DL. DL and the Neural Networks that underpin it came out of research labs and quickly leapt to the forefront of other AI functionality that has been evolving for the last 30 years or more.

Not surprisingly, enterprise organizations intend to adopt DL technology to solve business problems that could not be tackled before. The dominant areas for DL in production environments are for Computer Vision, Recommendation Engine, and Natural Language Processing. However, many specific problems in Life Sciences (for example) are also tackled using Neural Networks.

Although AI DL training and inferencing are well understood these days, the IO activity or bandwidth required to perform DL suffers from many misconceptions. This paper is an attempt to demystify DL IO. Here, we will review DL storage IO needs for training and inferencing, including parallel processing requirements, and will show some DL training and inferencing IO examples.

Why AI DL may need high-performing storage

AI DL uses high-performing storage for many reasons. AI DL IO patterns are characterized by almost 100% read workloads and are dominated by random reads, in often small to medium IO sizes, in contrast to traditional High-Performance Computing (HPC) which is large block sequential IO. These requirements are well suited for SSD or flash storage but not well suited for HDD or disk technology.

Higher performing storage can keep GPUs (and CPUs) busier, thus helping IT organizations justify the DL GPU hardware expense.

In addition, AI needs high-performing storage because DL model training traditionally consumes a lot of data. Image recognition models, recommendation engines and the like train on massive datasets. Large language models that train over protracted periods, require the fastest performing storage for, discussed below, checkpointing requirements.

Moreover, the nature of Neural Network (NN) training requires repeated passes over the entire dataset to calibrate models to perform effective inferencing. As such, having the training data dispersed over different tiers of storage may not perform well as this requires data movers plus orchestration and adds complexity to avoid resource conflicts.

Finally, to speed up AI DL model training and inferencing, enterprises often deploy multiple GPUs, and in many cases, multiple GPU servers working cooperatively to process data or models in parallel, which requires high-performing shared storage optimized for random rather than sequential IO.

AI DL models

Several diverse AI DL models are in use today. Three popular state-of-the-art DL models created in research labs are DALL-E (image generation from text), ChatGPT (chatbot) and AlphaFold 2 (protein structure predictions). Outside of labs, NVIDIA™, Hugging Face™, Google™ and other companies provide proprietary DL models available to enterprises interested in taking advantage of the technology.

From an IO perspective, the challenge with most research and proprietary models is that their training data is privately held. Moreover, if you had the data and models to evaluate IO bandwidth requirements independently, a cloud's worth of hardware and several months of processing time would be needed. Understanding what AI DL IO truly looks like thus requires analysis of simpler, non-proprietary DL models.

To that end, MLCommons®[1] publishes a set of AI benchmarks called MLPerf™ tests that use several open-source DL models.[2] MLPerf training benchmarks include eight data center DL models and three HPC DL models, as well as several inferencing benchmarks. While MLPerf benchmarks were designed to compare GPU and CPU hardware performance, some can also be used to study DL IO requirements.[3]

MLPerf AI DL training models

MLPerf AI DL training model data sizes vary greatly. MLPerf models that use modest amounts of training data include BERT for natural language processing (NLP) (13.5GB), Open Catalyst for quantum molecular modeling (23GB), RNN-T for speech recognition (30GB) and lightweight RetinaNet for object detection (38GB). Any of this data could be read into a single enterprise GPU's memory and then used to train repeatedly until the model achieves its required level of accuracy. While latency matters when reading training data into a GPU the first time, training for these models is primarily compute bound after that.

MLPerf DL models that use a greater amount of training data include ResNet-50 for image classification (160GB), DeepCAM for climate segmentation (8TB), CosmoFlow for cosmology parameter prediction (10TB) and DLRM for recommendation engine (23TB). Each of these models requires multiple read IO segments to train, using a one or more GPUs.

As shown above, the amount of data required for AI DL processing for some models is not as high as many have been led to believe. For instance, the BERT NLP model, a precursor to many of the large language models today, uses only 13.5GB of text data for training. GPT-3®, an exceptionally large and complex NLP model, used considerably more training data. However, as the training is very GPU intensive, rapid training is often done using a very large number of GPUs (GPT-3 for example is typically trained with 1024 GPUs) and this requires a lot of parallel IO for a storage subsystem.

As we will discuss, most model training and inferencing can be sped up by adding GPU (and CPU) hardware to process data and models in parallel. The challenge, as we will see, is scaling storage systems to support parallel IO performance.

[1] https://mlcommons.org/en/

[2] https://github.com/mlperf

[3] MLPerf is working on yet-unpublished storage IO DL benchmarks.

MLPerf AI DL inferencing models

Inferencing, in turn, has several multi-model MLPerf benchmarks. For example, MLPerf data center inferencing benchmarks include ResNet-50 (image classification), RetinaNet (object detection), 3D U-Net (medical imaging), RNN-T (speech to text), BERT (NLP) and DLRM (recommendation engine). As with training benchmarks, the quantity of data used for these benchmarks varies considerably but is only designed to drive model inferencing long enough to gather benchmark results.

All MLPerf data center inferencing models are tested in two modes of operation:

- **Server query mode** processes one inference for a request or transaction.
- **Offline mode** processes a file of multiple requests and performs inferencing on all of them.

From an IO perspective, the difference between the two modes is that one processes a single request or transaction at a time and the other processes a batch of requests or transactions at time.

Parallel processing

As mentioned, DL training and inferencing can be sped up through parallel processing. There are two standard approaches to DL parallelization:

- **Data parallelism** is when training or inferencing data is split across several GPUs, and training or inferencing is performed in parallel across all of them.
- **Model parallelism** is when model NNs are split across multiple GPUs and training or inferencing is performed across all of them.

Model parallelism is used when NNs exceed the size of GPU memory (up to 80GB). In these cases, layers of the model are assigned to each GPU, and the data is pipelined to each GPU in sequence. Data parallelism is more common. Here, training or inferencing data is split into segments, where each segment is processed on a different GPU. In some cases, both data and model parallelism are used to speed up model processing.

For model parallelism during training and inferencing, intermediate results must be communicated to other GPUs, i.e., results from training from one layer must be communicated to GPUs operating on the next. On the other hand, data parallelism training differs slightly between the two. For inferencing, data need only be split across GPUs, each independently inferencing on its data, whereas for training, data parallelism requires intermediate results to be combined with GPUs working on other segments.

For both DL training and inferencing, two types of additional IO may occur: checkpointing IO for model training and logging IO for inferencing.

- **Checkpointing IO** occurs during long-running training activity (days, weeks, months) to save processing state in case of hardware failure. When a failure occurs, model training can restart from the last checkpoint rather than having to restart from the beginning.
- **Logging IO** records inferences made by the model and the transactions used to generate them. Logging data allows DL models to be monitored in operation for drift or other problems and allows training data to be added, where applicable.

Checkpointing IO may or may not be significant depending on the size of the model, the number of GPUs operating in parallel and the frequency of hardware failures. Checkpointing can also be more sophisticated, e.g., only checkpointing model changes to minimize bandwidth. Logging IO is mostly driven by transaction or file processing rate but also depends on the number of GPUs in operation.

In addition, model training includes a **write pass** to record the DL model's final NN parameters, and inferencing includes **one or more read passes** to read the DL model's NN into GPU memory.

For most enterprise AI activity, writing out and reading back DL models should be trivial. That said, some large language transformer models have billions of NN parameters and next-generation transformers may increase the number of parameters by 100X or more.

AI DL model IO drivers

Some characteristics of DL that drive both training and inferencing IO include the following:

- **Data parallelization, splitting data across GPUs in operation** – Organizations that need faster DL training or inferencing may deploy data parallelism across multiple GPUs. One GPU may consume MB/sec of training or GB/sec of inferencing data, but when hundreds or thousands of GPUs are devoted to the task, bandwidth needs quickly multiply for the underlying shared storage system.
- **Model parallelization, splitting DL model NNs across GPUs in operation** – For very large DL models, organizations are forced to perform training and inferencing across multiple GPUs, which, like data parallelism, increases both DL speed and associated IO bandwidth requirements.
- **Networking hardware and protocol** – Historically, InfiniBand hardware held the advantage over Ethernet. However, that's no longer the case, as the latest generation of Ethernet hardware and RDMA protocols have reduced the speed differential considerably. In addition, as we will see, the recently released **NVIDIA GPUDirect Storage** protocol for InfiniBand or Ethernet RDMA transfers data directly from storage to GPU or vice versa without landing in CPU memory at all.

Important considerations that drive DL training IO include the following:

- **Training data set size** – Smaller training data sets can be read once and processed multiple times, but larger training data sets may need to be read multiple times to complete training. While peak bandwidth may not change, data set size may determine how long that bandwidth needs to be supported.
- **Number of training passes** – Many DL models require several training passes, or **epochs**, to properly calibrate NNs. The number of epochs matters little when the whole training data set can fit inside GPU memory, but for larger data sets, each epoch requires a complete read pass over the training data to move it into GPU memory for processing. As with data set size, epochs may not alter peak IO bandwidth requirements as much as change the length of time that bandwidth needs to be supported.
- **Batch and model size** – Models are trained in batches. A batch of data is read in and processed, and at the end of the batch any changes required to increase model accuracy (provide better

inferencing) are made. While these floating-point calculations are done for each training batch, there's a delay before the next batch is read in. The more NN parameters (# of layers * # of nodes per layer), the more computations needed. For larger data set sizes, batch and model size will be the limiting factor in how often GPUs will request more data to refill GPU memory.

- **Checkpoints** – As discussed above, some long-running training cycles may use checkpoints. Depending on the hardware failure rate, the bandwidth required for checkpointing can be significant for some models. Large models take longer to train.[4]

Batch size depends on the DL model and can be as small as a few records to dozens or more records. NN size is also DL model specific. Most enterprise DL models are within the range of 100K to 250M NN parameters. However, increasing inferencing accuracy often requires more NN parameters.

The data types used for training are typically files or objects and are model dependent. They may have internal element constructs that define a correspondence between data and its classification. Alternatively, that correspondence may be defined through directory structures. For example, medical images might have their diagnosis (or classification) embedded in a DICOM file, or the diagnosis may be kept in a separate directory with a one-to-one correspondence with X-ray image files in another.

The challenge with training that uses data or model parallelism over hundreds to thousands of GPUs is that not all storage systems can support high levels of bandwidth to a single or few directories. Although these same storage systems may support high IO rates across many directories, they perform much slower when only one or a few directories are being accessed.

This slowdown typically occurs for storage systems that maintain **cache coherency** across nodes, which slows down access until cache state can be communicated across all nodes in a storage cluster. On the other hand, one storage system has a **shared everything architecture** that doesn't require cache communication between nodes, thus allowing for much higher bandwidth to a single or few directories.

VAST Data offers an all-flash storage system called **Universal Storage** built on a **Disaggregated, Shared Everythin**g (**DASE**) scale-out, modern architecture. DASE separates the storage media from the CPUs that manage it, including all metadata, which are shared by all the VAST servers in a cluster. VAST's DASE architecture allows users to scale the storage capacity independently from cluster compute resources and by sharing everything, can scale to high levels of IO performance against a single or few directories.

Inferencing considerations that drive IO are simple in comparison with training. Here, the focus is on the server query transaction or batch file processing rates required. For data and model parallelism, the number of GPUs in operation will determine bandwidth needs. These characteristics also drive logging IO bandwidth. In general, inferencing consumes data at ~10X the training rate for the same models.

Thus, the higher the transaction or file inferencing rate and the higher the number of GPUs in use, the higher transaction read, and logging write bandwidth will be needed.

[4] As one example, OpenGPT (machine-generated text) took many months to train.

AI DL IO examples

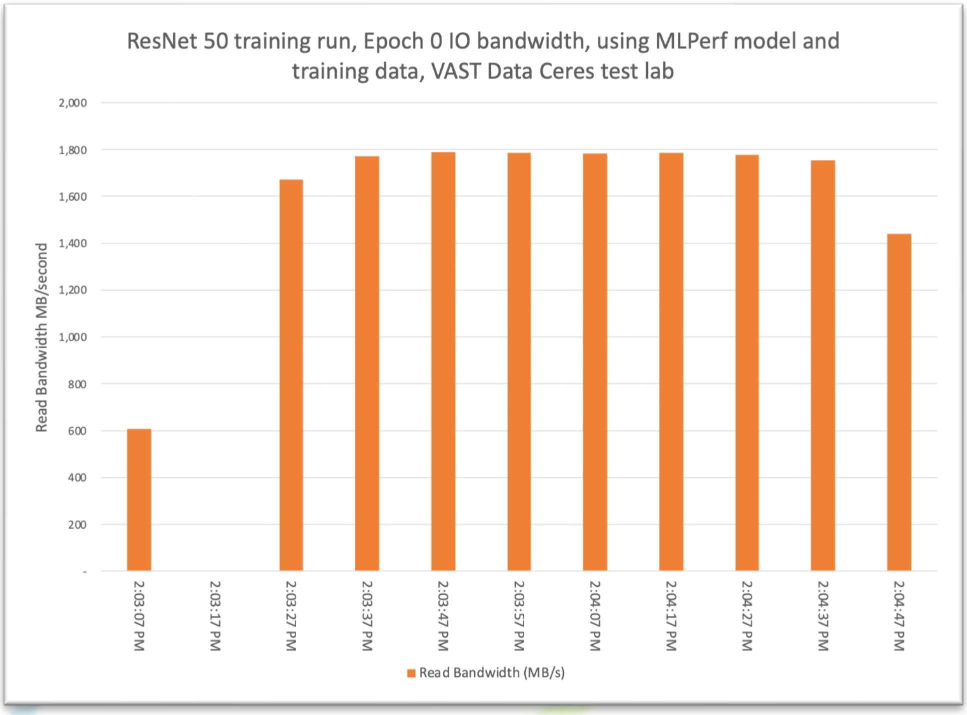
AI DL training IO example

Our training IO example uses the ResNet-50 (image classification) DL model. The model and training data are from MLPerf. The input dataset is the well-known Imagenet dataset of 1000 classes of images with 1300 images/class.

The chart shows read IO bandwidth. In this example, the VAST Data Lab used **eight A100 40GB** GPUs with data parallelism during training.

Read bandwidth is displayed but read IOPS would show a similar picture, as each read consists of a single image that averages ~115KB.[5]

As the chart exhibits, Epoch 0's burst of IO activity never exceeded 2,000MB/sec (2GB/sec). The average for max bandwidth intervals during training was ~1,715MB/sec (1.7GB/sec). As we will see below, VAST Data storage is capable of much higher bandwidth than the bandwidth seen here.



All other Epochs (2-N) required to train the ResNet-50 model performed no further read IO, as all 160GB of training data had already been read into the GPU or server memory. Thus, ResNet-50 with the standard Imagenet dataset is not IO bound even with eight GPUs. Note that the write IO needed to write out the trained model never exceeded 1MB/sec during the test.

Our formula for ResNet-50 training read IO peak bandwidth would look something like this:

ResNet-50 Training peak bandwidth = # (of 40GB) GPUs * 215MB/sec (for non-GPUDirect Storage)

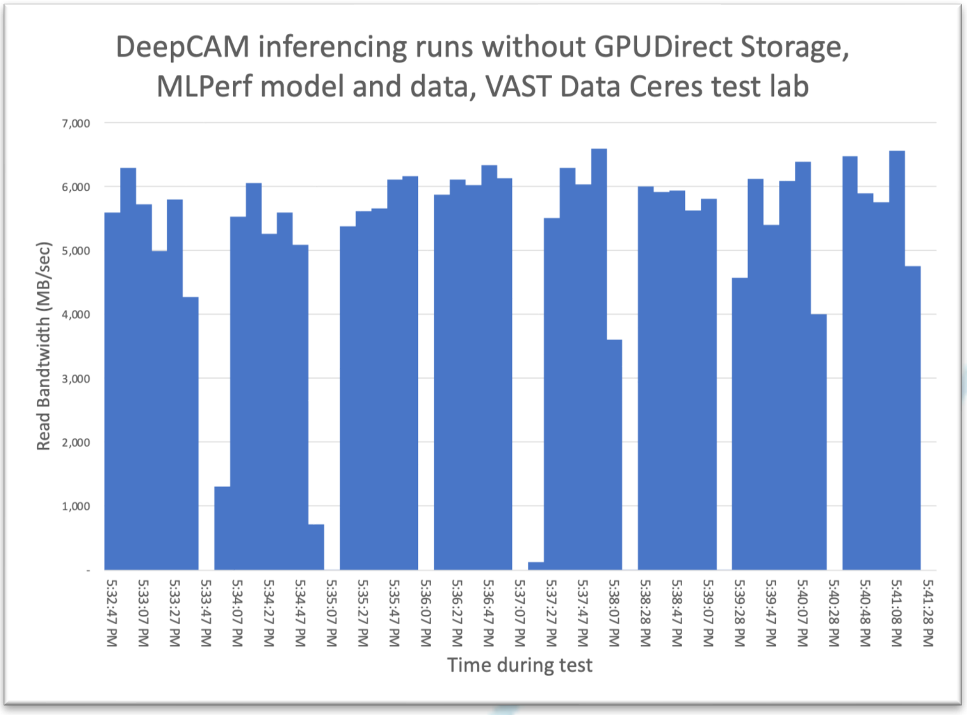
See below for an example of **inferencing** with and without NVIDIA GPUDirect Storage.

It's important to note here that for ResNet-50 training, the key limiting factors driving peak bandwidth are the model batch size-NN parameter count and the number of GPUs in operation. The training data sets size is not even a consideration for ResNet-50 peak bandwidth.

[5] As reported in Dell EMC PowerScale and NVIDIA DGX A100 Systems for Deep Learning | H18597.

AI DL inferencing IO example

Our inferencing example comes from another VAST Data test lab run, this time using the DeepCAM Climate Segmentation model. Both the model and inferencing data come from MLPerf but has been modified by NVIDIA.



The chart depicts offline inferencing and shows eight bursts of inferencing activity, each taking ~50 seconds at an average of ~5.5GB/sec. The test run used data parallelism and **two** A100 GPUs.

As with the training example above and as we will see shortly, VAST Data storage is capable of much higher bandwidth than 6GB/sec. As such, peak read IO bandwidth for DeepCAM inferencing does not seem to be storage limited.

Our formula for DeepCAM offline inferencing peak bandwidth would look something like this:

DeepCAM Inferencing peak bandwidth = # GPUs * 2.8GB/sec (for non-GPUDirect Storage)

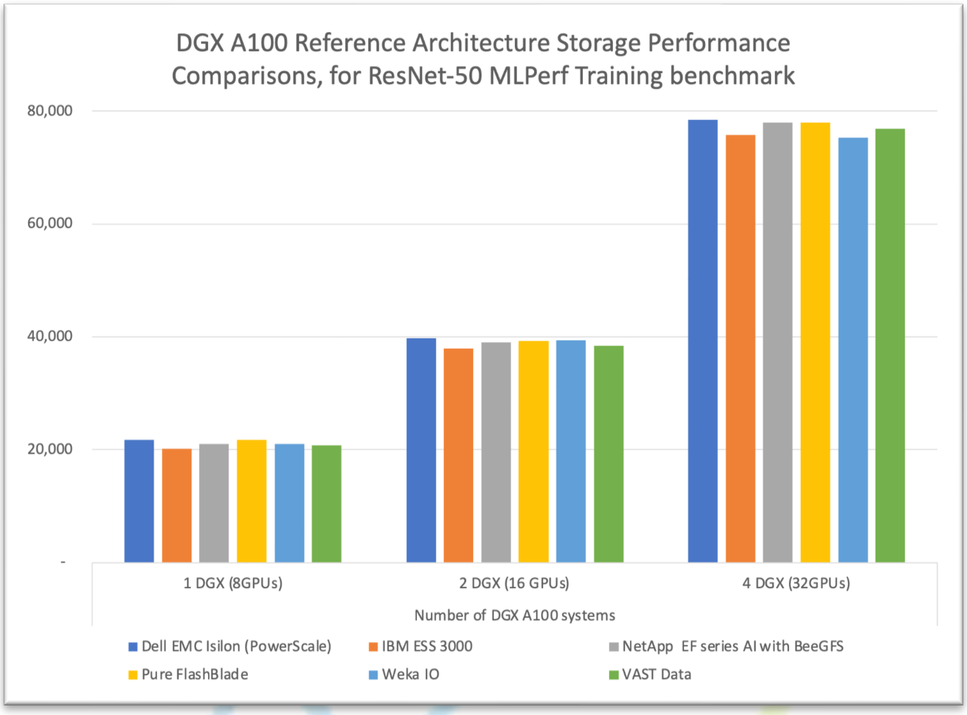
Later, we will show that DeepCAM inferencing using NVIDIA GPUDirect Storage is much higher.

NVIDIA DGX™ reference architecture IO performance

NVIDIA, together with several storage system vendors, has published DGX-A100 system reference architecture reports showing ResNet-50 (image classification) training performance. These systems all use A100-80GB GPUs.

All the reports assess training performance using the same ResNet-50 training runs. Below, we consolidated the IO performance from each report onto one chart.

Performance shown is in images per second.



Recall that the average image size for ResNet-50 training is ~115KB. As such, the 20K images/sec on the chart correspond to ~2.3GB/sec, 40K images/sec correspond to 4.6GB/sec and 80K images/sec correspond to 9.2GB/sec.

None of the systems supplied more than 10GB/sec to DGX-A100 systems during ResNet-50 training. We assume that data parallelism was in use as each new group of columns show higher bandwidth for the same ResNet-50 model.

What's striking on this chart is that **all storage systems perform nearly the same**. Thus, ResNet-50 training is not IO driven, at least not up to 4 DGX-A100 or 32 A100-SXM-80GB GPUs.

Our formula for ResNet-50 peak bandwidth needs would be (using the 2 DGX bracket):

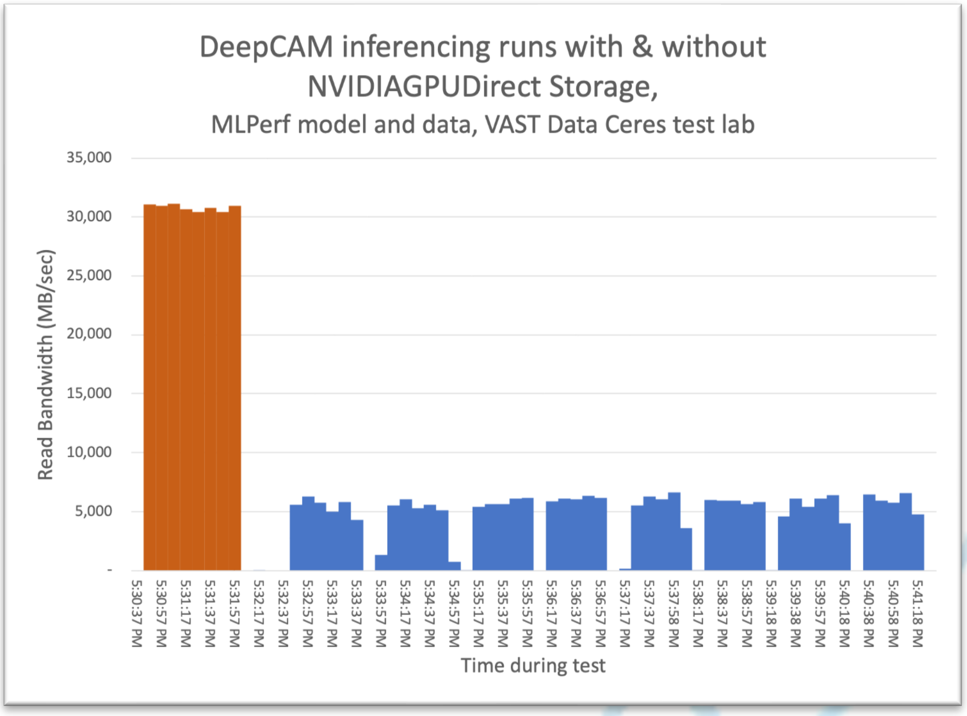
ResNet-50 Training peak bandwidth = # (of 80GB) GPUs * 280MB/sec (for non-GPUDirect Storage)

The increase in bandwidth with these runs vs. the VAST Data lab run above appears mostly due to the increase in GPU memory size.

NVIDIA GPUDirect Storage IO inferencing performance

Another VAST Data Storage inferencing test run shows an example of NVIDIA GPUDirect Storage vs. non-GPUDirect Storage. These results use a version of the DeepCAM Climate Segmentation Inference Benchmark from NVIDIA that support GPUDirect Storage but based on MLPerf models and data.

The chart below shows a continuation of the previous testing. It uses the same hardware, same data, and same model. The testing was also done across **two** GPUs using data parallelism.



Here, we show both NVIDIA GPUDirect Storage in the first high burst of IO in orange and non-GPUDirect Storage in the following eight blue bursts. Both use the same VAST Data storage.

With NVIDIA GPUDirect Storage, the VAST Data system was capable of ~30,800MB/sec or 30.8GB/sec over eight intervals of approximately 10 seconds each. The eight bursts (blue), previously discussed, use ~5.6GB/sec without GPUDirect Storage.

With NVIDIA GPUDirect Storage VAST Data is capable of 5.5X the bandwidth of non-GPUDirect Storage.

Our formula for DeepCAM inferencing peak bandwidth with NVIDIA GPUDirect Storage would look something like this:

DeepCAM inferencing peak bandwidth = # GPUs * 15.4GB/sec (for GPUDirect Storage)

Summary

AI DL model training and inferencing IO can vary greatly and depend on several factors. Considerations driving IO rates and bandwidth requirements include the model used, the data set size, the protocol in use and the number of GPUs in operation.

To provide a better understanding of DL model IO, we have shown several examples of IO bandwidth requirements. Any MLPerf benchmark model could be similarly tested, If desired.

NVIDIA GPUDirect Storage can make a big (>5X) difference in IO bandwidth, as can the number of GPUs used with data and model parallelism. However, with parallelism, a storage system's ability to scale parallel IO to few directories also matters.

The random read intensive nature of DL workloads, along with uniform access to the entire dataset requires a single all-flash storage system. Moreover, IO latency seems to matter for training and inferencing, also suggesting the need for all-flash storage for DL.

Another critical aspect for the storage is to be able to scale with linear performance gains only feasible with shared-everything storage. Storage systems with cache coherency that inhibits linear performance scaling over few directories are not suitable for large scale GPU based computing.

VAST Data's Universal Storage is a disaggregated, shared-everything (DASE) solution that scales performance linearly against a single or few directories that also supports the NVIDIA GPUDirect Storage protocol.

Given that the AI DL IO bandwidth requirements for training and inferencing are well within the IO performance capabilities of VAST Universal Storage, enterprise organizations should consider taking advantage of VAST Data’s all-flash storage performance to simplify and accelerate AI adoption.

Silverton Consulting, Inc., is a U.S.-based Storage, Strategy & Systems consulting firm offering products and services to the data storage community.