The Role of Storage for efficient Machine Learning

CHPCConf 2020

Sven Breuner
Field CTO
sven@excelero.com

Excelero

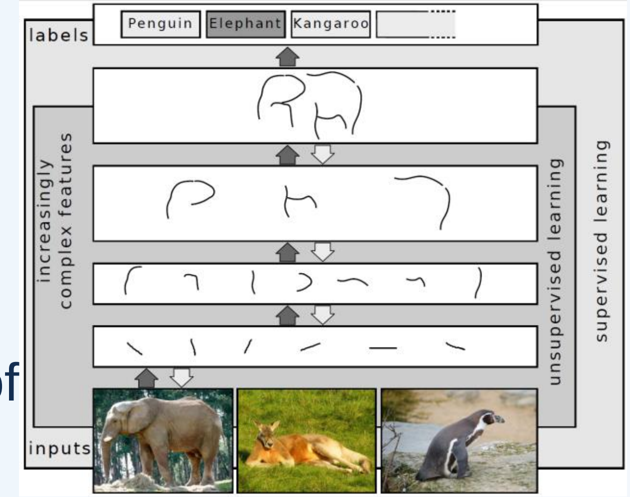
Overview

- Focus on Deep Learning (DL) as a very successful and storage-intensive AI subdomain
- Characterizing DL storage access patterns
- Similarities and differences between HPC & DL
- Why is the DL storage problem often noticed too late
- How to avoid DL storage problems
  - Typical storage requirements
  - Typical storage benchmarks
- What’s next for DL storage

Excelero

Deep Learning: Teaching a Computer to recognize Objects

- Animals are easy to recognize for humans, but difficult for computers
- Too many different variations to manually program an algorithm that detects e.g. dogs
- To *learn* what a dog is, a computer needs to see millions of different dogs in all colors, shapes and sizes. This is called *training*.
- The result of the training is a *model*, which contains the characterization of the dog



The 3 high-level phases of Deep Learning

Data preparation

- “Normalize data“: Use same color palette, resolution, rotation, annotate features, reduce to relevant objects, …

Training

- Typically based on GPUs for their high computational parallelism
- Look at millions of interesting “objects“ again and again
  - E.g. in different order, different rotation etc.
  - And to continuously improve the quality of the model
- Very storage-intensive, so our main focus

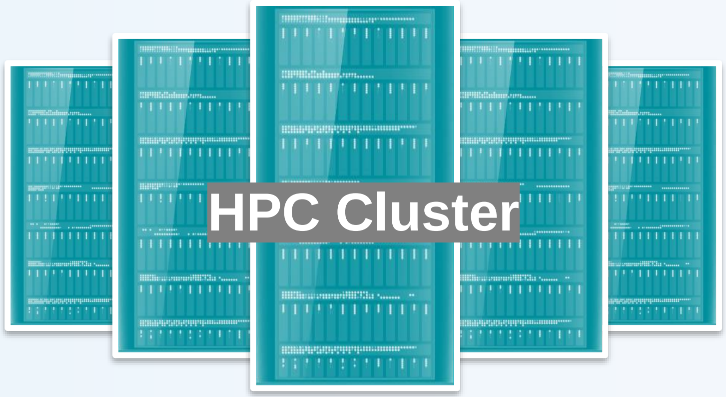
Inference

- Show something to the computer to check if it recognizes an object of interest after the training

HPC vs DL: Similarities and Differences

Similarities between HPC & AI/DL

- One host is not enough, so scale out through clustering with high-speed (RDMA) network
- Shared storage, so all nodes have access to same data
- Coordinated resource sharing



Differences between HPC & AI/DL

- HPC:
  - Strong scale out (100s to 100,000s of nodes);
  - High streaming read+write bandwidth
- AI:
  - Scale up (supercomputer in a box), then scale out
  - Primarily read intensive
  - Lots of small and highly concurrent storage accesses

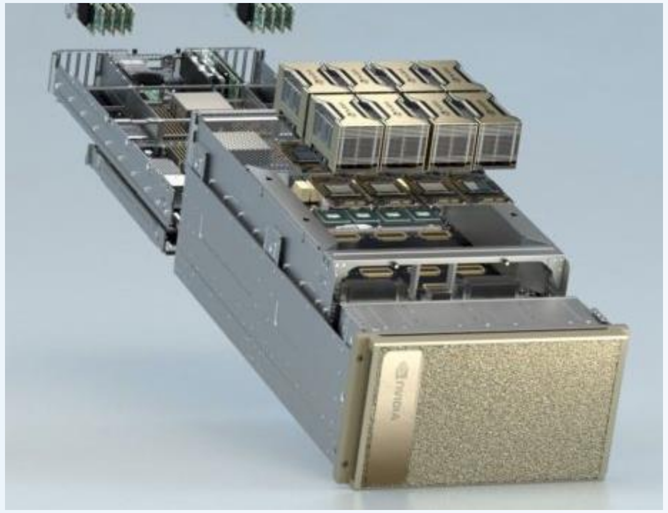
A closer Look at the Storage Specifics for DL

What do the differences from classic HPC mean?

- Scale up (supercomputer in a box), then scale out:
    - From the storage perspective, each client has much higher demands than before

- Primarily read intensive:
    - HPC storage concepts like “Burst buffers” don’t work for AI, because they are for writes, not reads

- Lots of small and highly concurrent accesses to shared storage:
    - Spinning disks fail by design, because they hate small and highly concurrent reads
    - This makes NVMe drives the standard technology for DL storage
    - Making NVMe performance available over the network is the new storage industry challenge

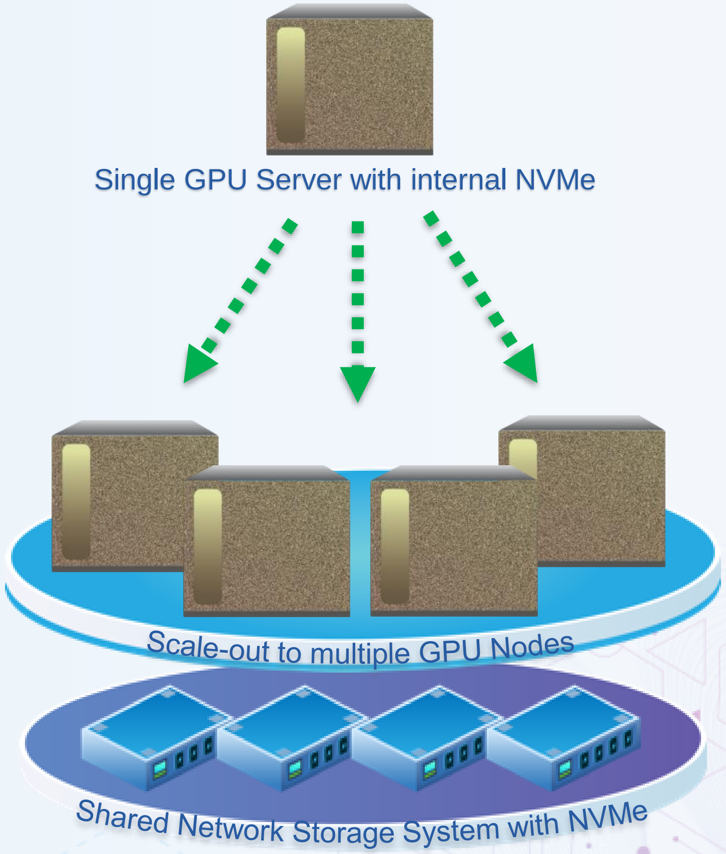


Nvidia DGX A100

NVMe Drive Examples

Why is the Storage Problem often recognized too late

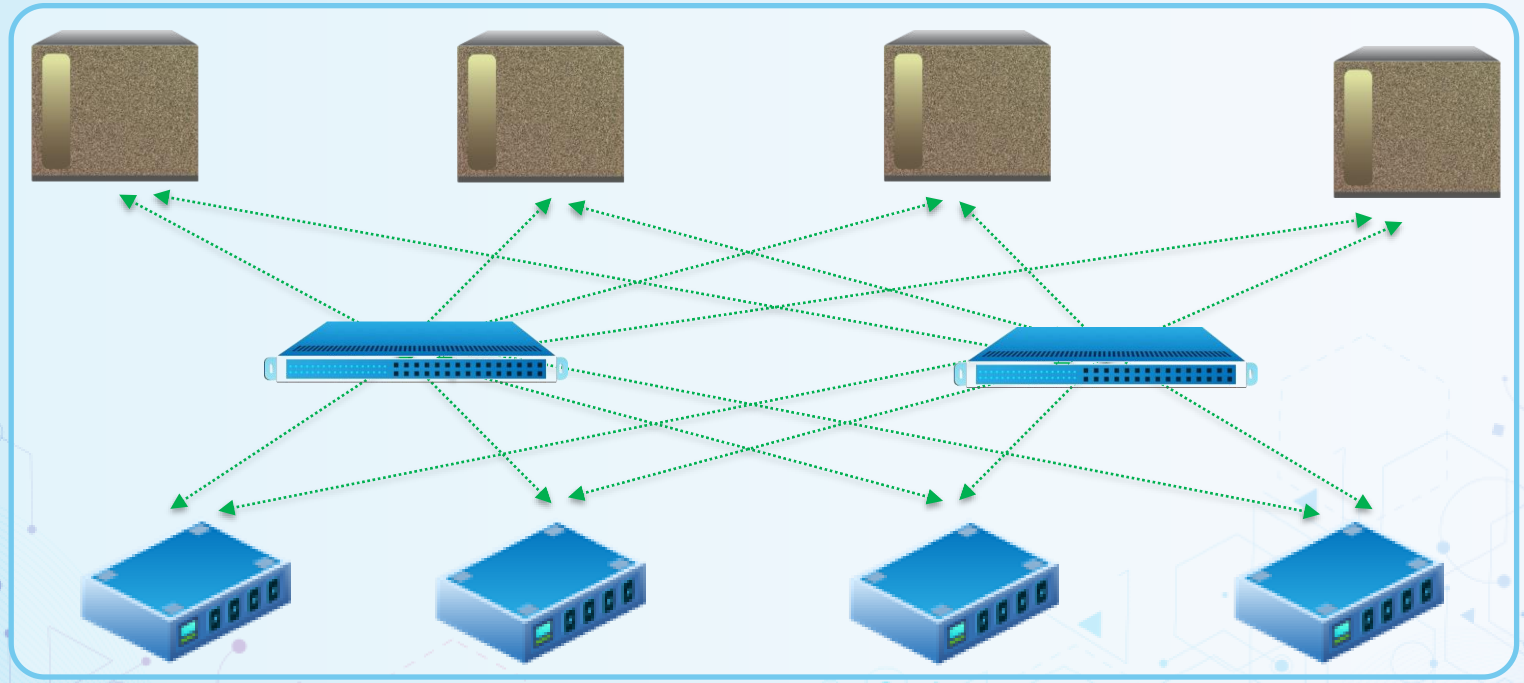
- Initial Deep Learning testing/evaluation is usually done on small systems
  - E.g. a single Nvidia DGX or a few compute nodes with internal flash storage
  - Training dataset is “local” inside the compute nodes
- For more serious DL, scale-out is required and keeping local copies of the data inside the compute nodes is no longer feasible
  - Network attached NVMe storage often has different performance characteristics compared to local/internal NVMe storage
- Specifying/designing such storage systems requires understanding of the special DL demands



Typical Deep Learning Storage Requirements

- Shared network storage system specialized for:
  - High read IOPS for lots of small random accesses
  - Often: High read IOPS for lots of small files

- Ability to scale to higher performance and capacity when needed
  - Because you will want more when you discover the amazing possibilities of DL ☺



How to test Storage Performance for Deep Learning? (1)

- ImageNet (annotated image database), TensorFlow (DL framework) and ResNet-50 (neural network) are often used to test storage systems
- Unfortunately, the result is of very limited practical relevance, because the workload too easily gets compute bound
  - Real-world applications of DL are often more optimized and thus have higher storage requirements

TensorFlow	ResNet-50 V1.5	112	76.98 Top 1 Accuracy	17,343 images/sec	8x A100	NVIDIA DGX-A100	20.09-py3	Mixed	256	ImageNet2012	A100-SXM4-40GB

Source: https://developer.nvidia.com/deep-learning-performance-training-inference

- Average file size in ImageNet is about 100KB.
- This test first converts the lots of small image files to a few large files (TF Records)
- 17,000 images per sec means 1.7GB/s of 100KB random reads for a single system of 8x A100 GPUs
  - A single HDD can do ~100 random reads per sec, so would require 170 HDDs
  - A single NVMe drive can deliver 3-7GB/s for this type of access pattern
- ImageNet is only 150GB, so can be cached too easily

How to test Storage Performance for Deep Learning? (2)

- It is generally desirable to have a flexible test for DL storage performance
  - Without being bound by the specifics of ImageNet/TensorFlow/ResNet50
  - To predict scalability
  - To see what the maximum possible objects/sec value for training is
  - To experiment with different file formats, e.g. directly with small files vs. extra conversion time to records in large files
  - To see the difference between host memory read speed and GPU memory transfer speed

elbencho

A distributed storage benchmark for file systems and block devices with support for GPUs

elbencho was inspired by traditional storage benchmark tools like fio, mdtest and ior, but was written from scratch to replace them with a modern and easy to use unified tool for file systems and block devices.

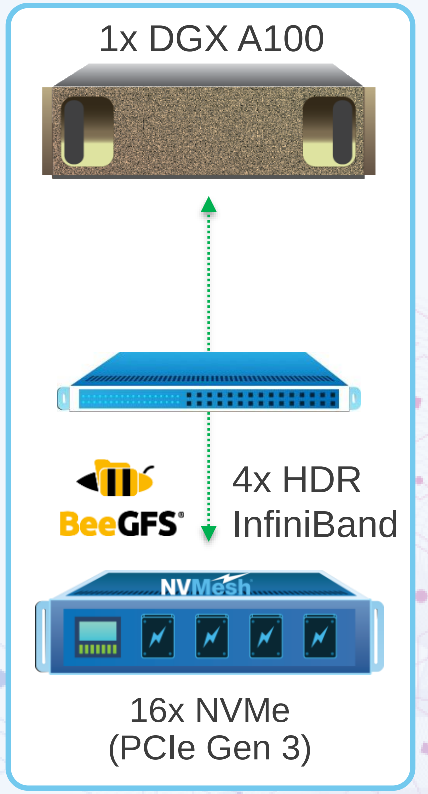
Elbencho on github: https://github.com/breuner/elbencho

Elbencho Test Examples with/without GPU Transfer

Test System:

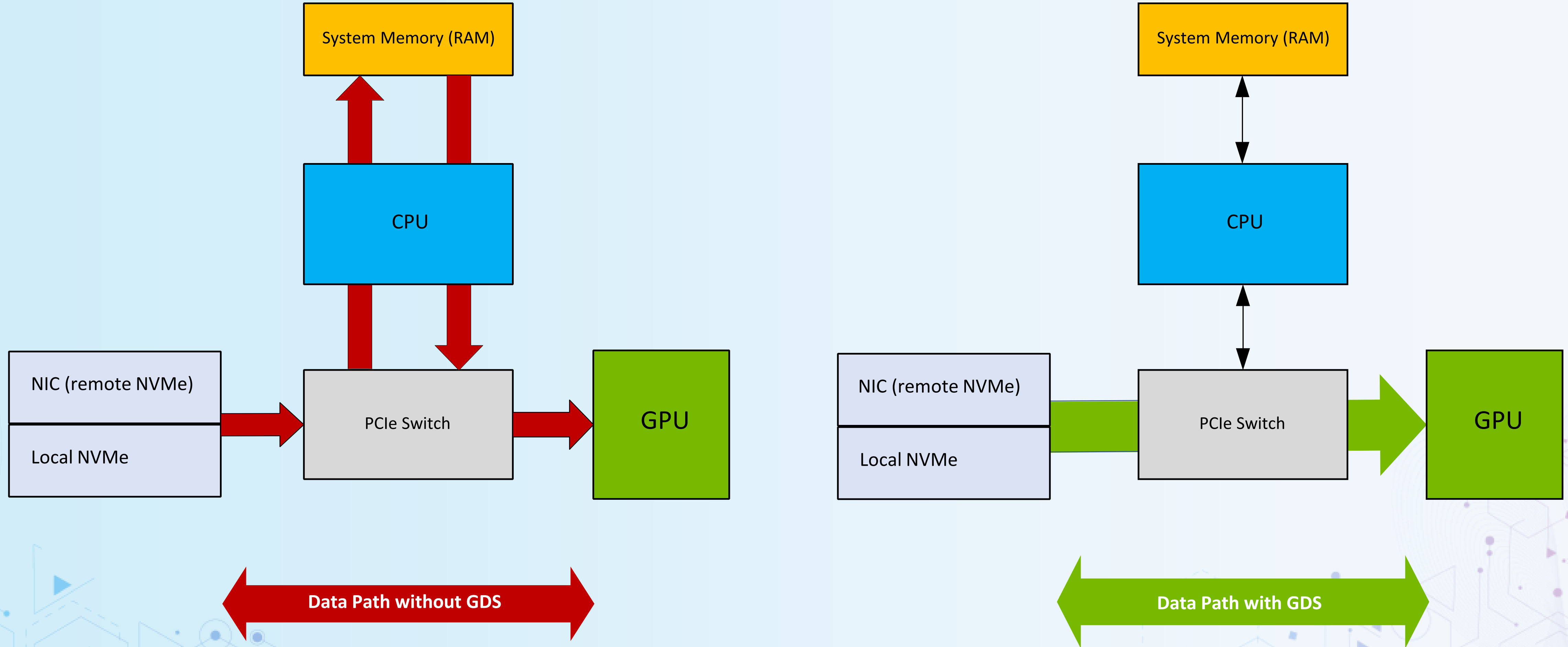
- 1x Nvidia DGX A100 at University of Pisa, Italy
- 1x Quad-Server: 16x PCIe Gen3 NVMe drives & 4x HDR InfiniBand
- BeeGFS parallel file system + NVMesh NVMe RAID

```
# 1MB random reads from large files into host memory (no GPUs involved)
dgx-a100$ elbencho -t 128 -r -s10g -b 1m --direct --rand --cpu
        /mnt/beegfs/file{1..128}
Result: 50.3 GB/s (5% CPU utilization)

# 1MB random reads via host memory into GPU memory
dgx-a100$ elbencho -t 128 -r -s10g -b 1m --direct --rand --cpu
        /mnt/beegfs/file{1..128} --gpuids "0,1,2,3,4,5,6,7" --cuhostbufreg
Result: 45.7GB/s (7% CPU utilization)

# Read 512000 small files (128KB file size) into host memory (no GPUs involved)
dgx-a100$ elbencho -t 128 -r --direct -n 40 -N 100 -s 128k --cpu
        /mnt/beegfs
Result: 142005 files per sec (6% CPU utilization)

# Read 512000 small files (128KB file size) via host memory into GPU memory
dgx-a100$ elbencho -t 128 -r --direct -n 40 -N 100 -s 128k --cpu
        /mnt/beegfs --gpuids "0,1,2,3,4,5,6,7" --cuhostbufreg
Result: 139444 files per sec (7% CPU utilization)
```



Elbencho on github: https://github.com/breuner/elbencho

What's next for DL Storage? GPUDirect Storage (GDS)



Great to see that Nvidia is raising awareness for GPU storage performance to prevent people from noticing such issues too late.

Thank you!

And looking forward to seeing you do great things with AI in the future!

Sven Breuner
Field CTO
sven@excelero.com

Excelero