Optimization Strategies for the Deep Learning storage stack

The Deep Learning Data Lifecycle

01 Acquire

02 Process

03 Train

04 Production

05 Store

Let’s look at this bit

But remember there are more challenges in the other stages

So what is different about Deep Learning?

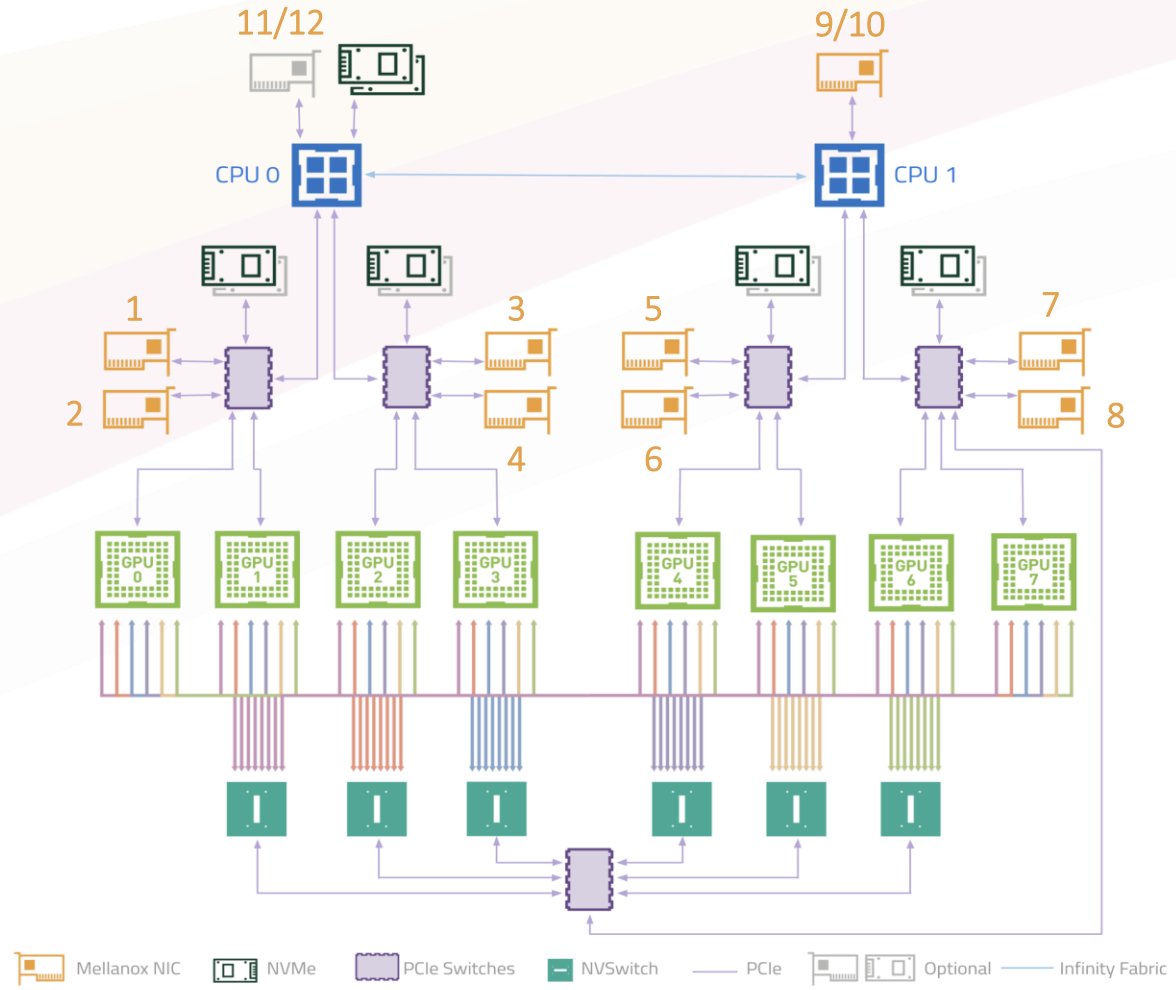
And how can we create a storage environment that keeps things efficient and fast?

IO Data Path	Backtesting	Deep Learning
Application	Time Series Database	AI Framework
Application IO	Read, Widespread and Random Big mmap operations	Read (and Write!), often mmap. Many re-reads
Data Volumes	Typically 100s TB to PBs	PBs --> 100s of PBs,
Compute System Processor	CPU	CPU and many GPU
Compute System Network	Single connection to 100G/IB	Many connections to 200G/IB

The Deep Learning Compute Architecture is Different

An Example GPU Platform

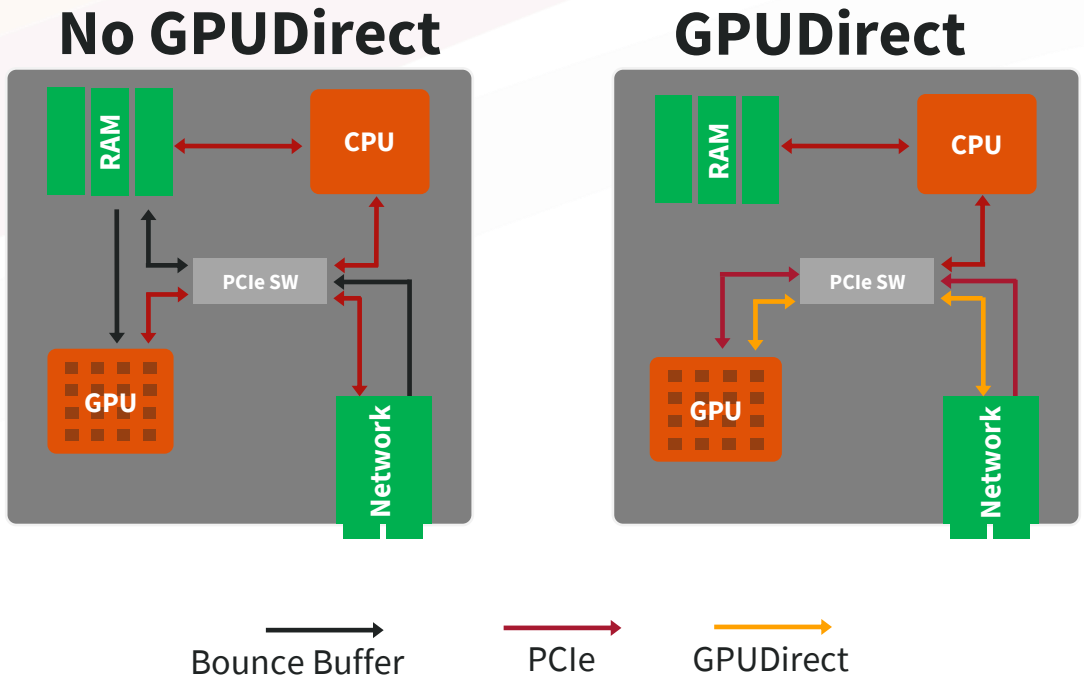
- ..is much higher demanding than conventional compute!
- Dual port NICs (storage) close to CPU, up to 28 GB/s per NIC
- Eight GPUs interconnected within the system through NVSwitches (NVIDIA proprietary interconnect, 10x higher bandwidth than PCIe Gen4, 600 GB/s GPU-to-GPU)
- Two single port NICs (cluster) and two GPUs per PCIe switch, up to 24 GB/s per NIC, optimal path for GPUDirect Storage
- GPUs from multiple systems can communicate through a cluster network, ideally via single port NICs on same PCIe Switch as GPU (ports 1-8)



How can we optimize the compute side datapath?

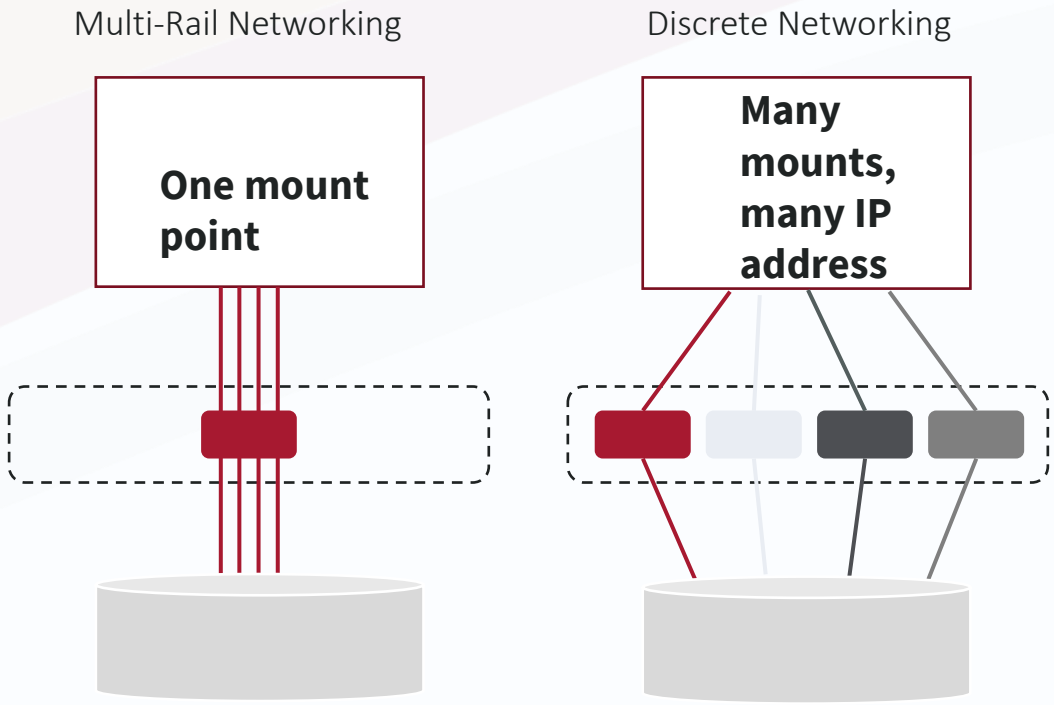
Optimizing the DataPath at the CPU/GPU level

- GPUDirect enables Direct Remote Memory Access from the filesystem to GPU memory
- 2x-8x higher BW data transfers between Storage and GPU
- 3.8x lower latency with no faulting and bounce buffers
- Stable and flat latencies as GPU concurrency increases
- Lower consumption of host CPU or memory subsystem
- The GPU is the computing element with the highest IO bandwidth, e.g. 215 GB/s vs. the CPU's 50 GB/s
- Very fast access to petabytes of remote storage faster than even the page cache in CPU memory



Optimizing the DataPath at the Network level

- In GPU systems we usually have access to multiple interfaces. How can we use them all, without making our lives complicated?
- Some filesystems enable the grouping of network interfaces to achieve full aggregate throughput capabilities on a Client
- This doesn't need networking changes – it all happens with the filesystem software
- This Improves peak performance with **single mount point and no complicated setup**



The Checkpoint Problem

Deep Learning Training Workflow

- Most Deep Learning workloads follow the same paradigm:
  - **Read Data** into GPU memory, compute and all-reduce
  - **Read Same Data** into GPU memory, compute and all-reduce
  - **Checkpoint** ← save the state for later use!
    - Maybe the job crashes and needs restarting
    - Maybe we want to change parameters (increasingly common)
  - Read Same Data into GPU memory, compute and all-reduce

Large Transformer Models create big challenges

- GPT-2 (Generative Pre-trained Transformer 2) is a model that translates text, answers, questions, summarizes passages and generates text output
- GPT-3 is 100 times larger (more sophisticated)
- These models scale from 1Billion to 1Trillion parameters in size
  - parameters are the number of layers, the number of neurons per layer, the number of training iterations, etc. – these are getting bigger and bigger every year by multiple factors
- This is a tough use case for storage because large models require large checkpoints to hold their state
- **Example:** A GPT-3 13Billion parameter model: (this is quite a small model today!)
  - 4-way tensor parallel, 2-way pipeline parallel, distributed workload
  - Size of checkpoint file is **172GB** across 8 files ← a small model…
- Every minute spent waiting for IO is wasted time

Q. So how can storage cope?

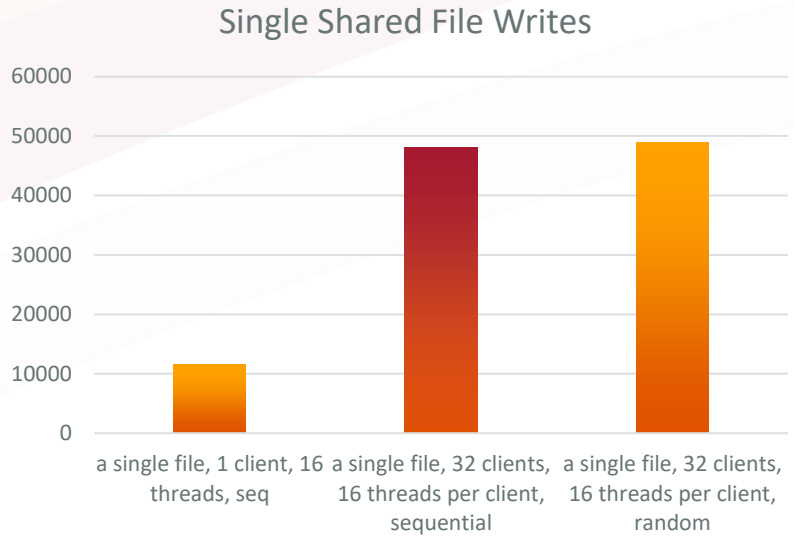
A. By having good Write Performance for (a modest number of) concurrent writers!

Read and Write Performance of Storage

Usually there is ***some*** discrepancy between read and write performance for a storage system, since

- the write path needs to have data protection applied (erasure coding) and be written safely to persistent media or protected media
- Flash is faster for reads vs writes
- Concurrent writers to a single file need to be handled with locks

But for some storage systems the write is many factors lower than reads. To handle large scale checkpoints, you want the storage system write performance to be more than 50% of the read performance.



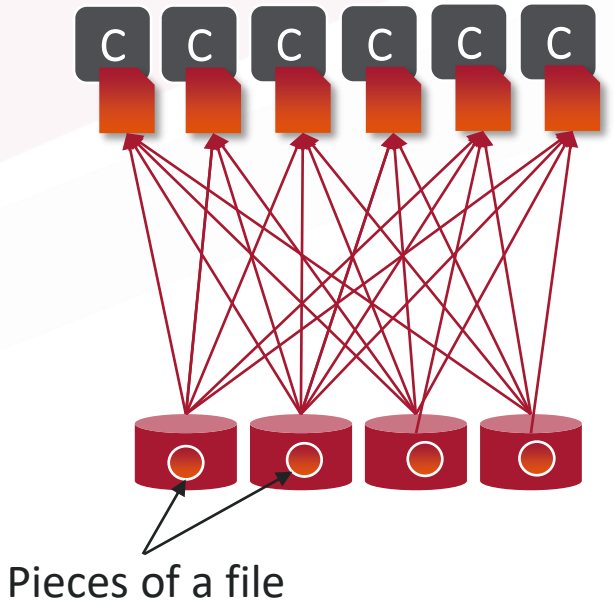
Example of good write performance (2RU storage system)

High Concurrency Operations to a single file

Shared File Operations

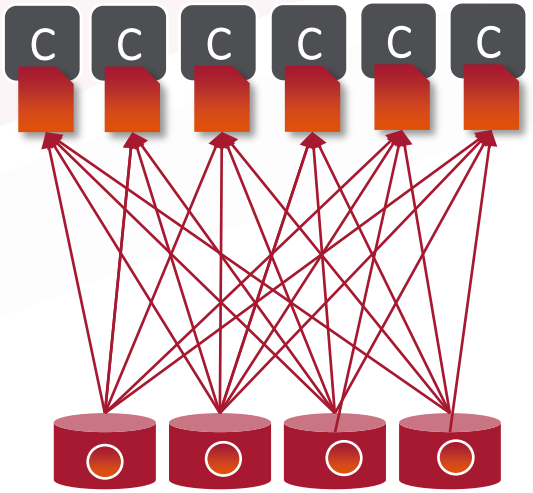
Many processes on a distributed system often want to read from a single file

- This can often be a problem for filesystems that aren't truly parallel
- Problem is that
  - the pieces of the file reside upon a limited set of physical devices
  - Even if each individual process is reading in a nice sequential fashion, because there are many competing readers, the IO pattern seen by the storage system will be random
  - If a storage system is not truly parallel, the devices can become an area for contention



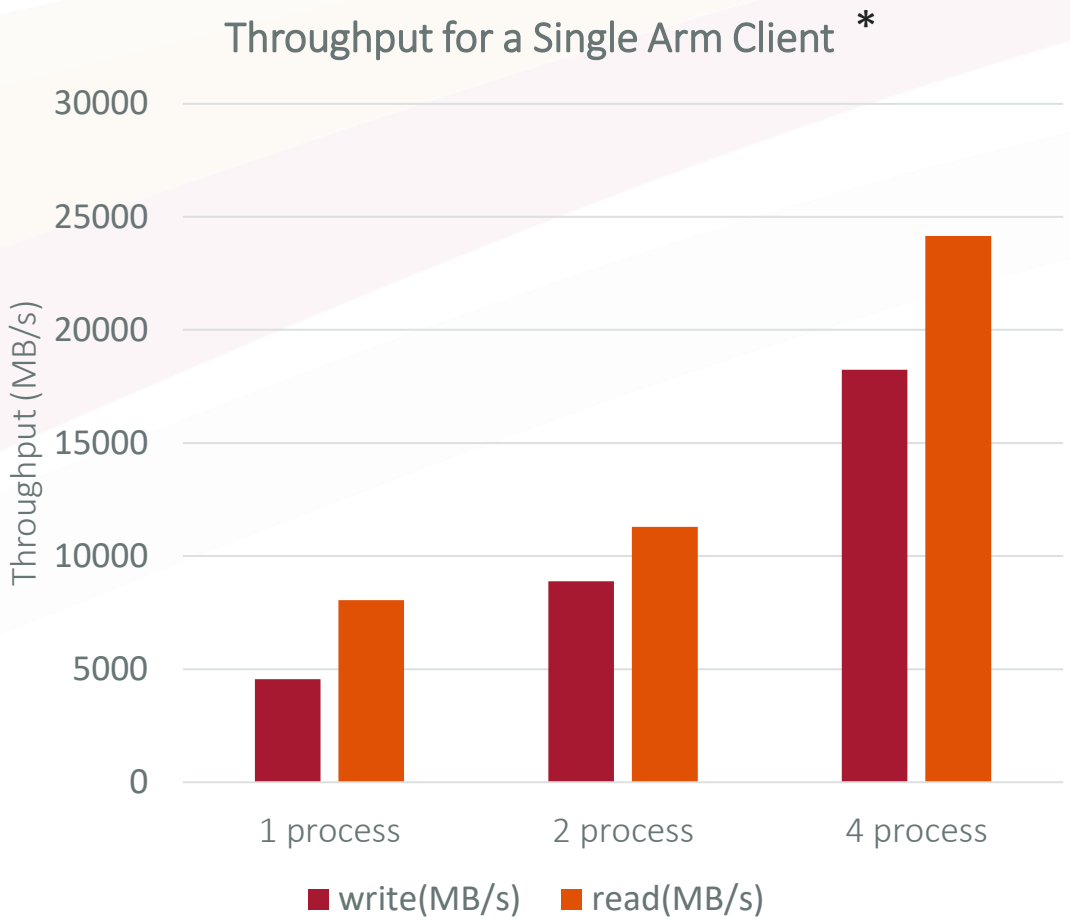
So now that we have good single thread/single client performance, lets tackle concurrency

- Flexible striping policies allow a filesystem to contend with the most problematic concurrent IO patterns better.



Single Arm Client Performance **

- Single thread performance can be critical, particularly for poorly optimized code
- With some filesystems, even a single thread can read over 5GB/s – up to 20+GB/s from a single server
- Ampere 2 x 80 CPU cores with 2 x IB-EDR

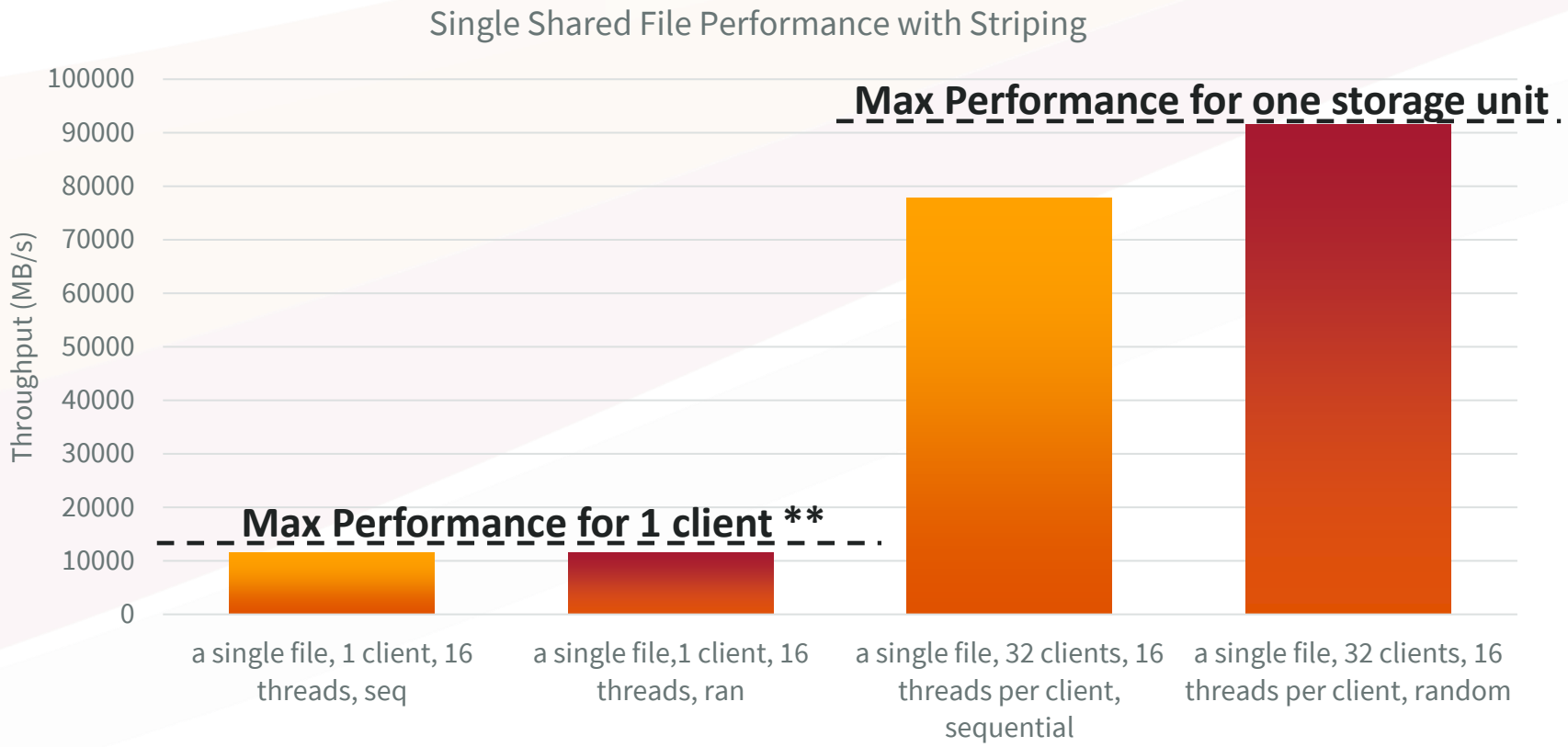



```
mpirun -np $n --allow-run-as-root ior/src/ior -w -r -t 1m -b ${(512 / n))g -e -F -vv -o /exafs/file
```

*Not a STAC benchmark

** limited by NIC performance

Parallel Filesystems can max-out the hardware even with just one file *



*Not a STAC benchmark

** limited by NIC performance

Optimization Strategies for the Deep Learning Storage

- Deep Learning can involve very large amounts of data subject to:
  - Very high demanding compute-side environments
  - Checkpointing (writes) that can stop productive output
  - Large scale re-reads of data
  - Highly concurrent access to individual files
- To find a suitable storage solution one needs to know:
  - High Throughput/IOPs with low thread count
  - High Throughput/IOPs even to small client count
  - Concurrency Results for shared file access
  - Good write Performance
  - Cost and Complexity of the storage system to get the best numbers!


ddn