Microsoft

The MLPerf Storage Benchmark and Lustre: Overview and Best Practices

Ellis Wilson, Wolfgang De Salvador

Myth-busting Lustre and AI

- OpenSFS took measures for Supercomputing 2025 to emphasize Lustre's effectiveness for not just classic HPC, but also modern AI workloads
- A great first step – but this messaging and presence in the AI conversation needs to be consistent
- **Purpose of this talk:** Educate Lustre developers and administrators on MLPerf Storage, the defacto AI storage benchmark today
  - *Let's scale-out our impact in the conversation*

Outline for Today's Talk

- System Architecture and Cost Considerations with GPUs
- DLIO: Simulating AI Workloads without GPUs
- MLPerf Storage: A Standardized AI Benchmark atop DLIO
- MLPerf Storage Version 2.0 vs 3.0
- I/O Patterns in MLPerf Storage 2.0
- Running MLPerf Storage
- Final Thoughts

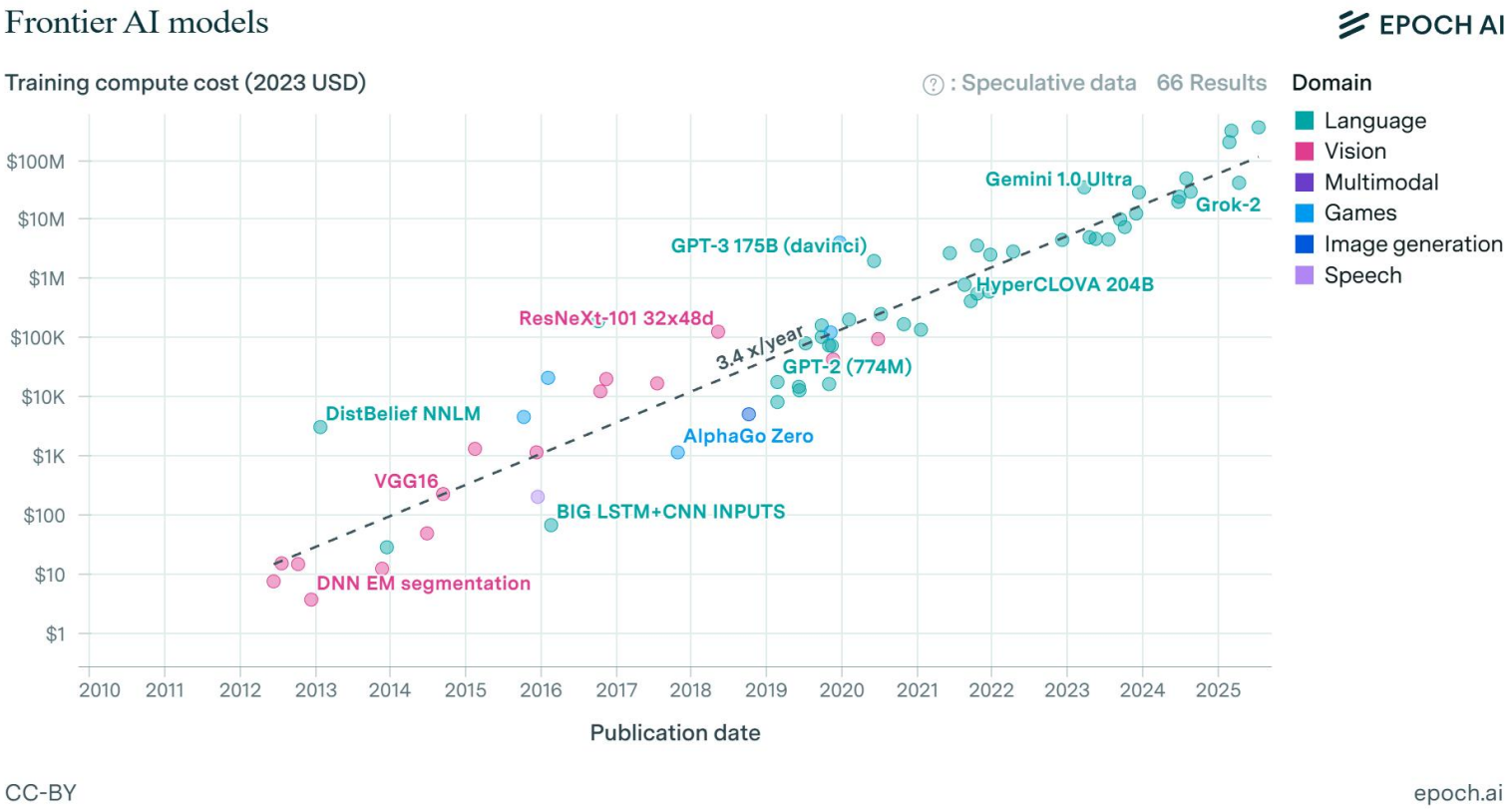
System Architecture and Cost Considerations with GPUs

AI Training compute cost YoY

More powerful, massive models

Multi-modal, multi-task

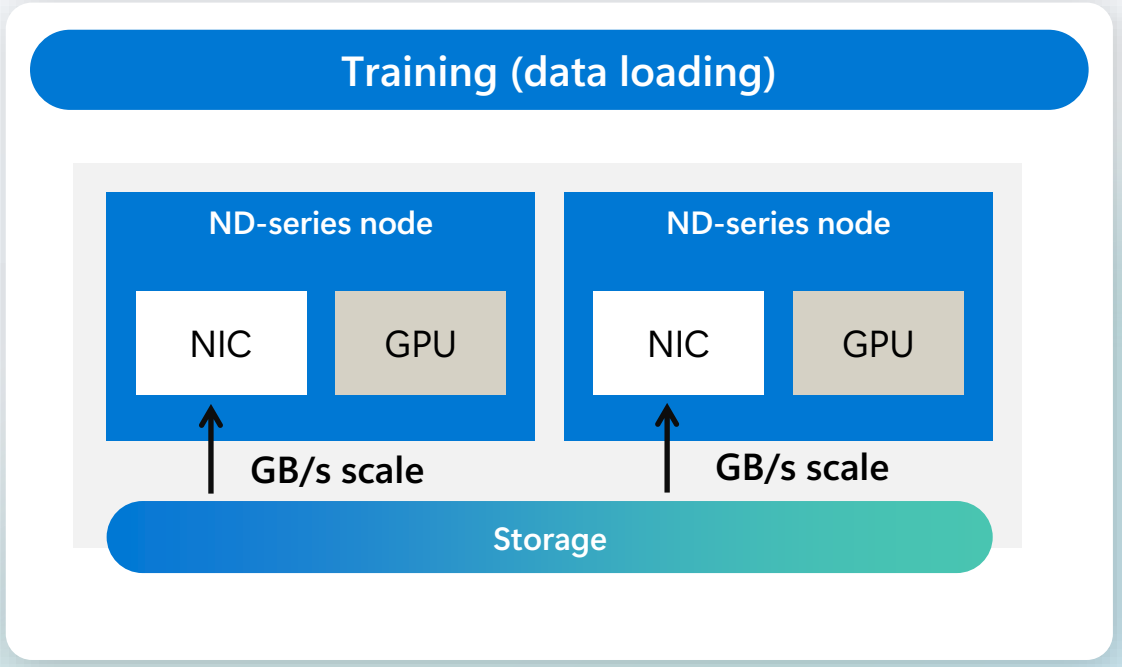
Constant YoY computational power increase

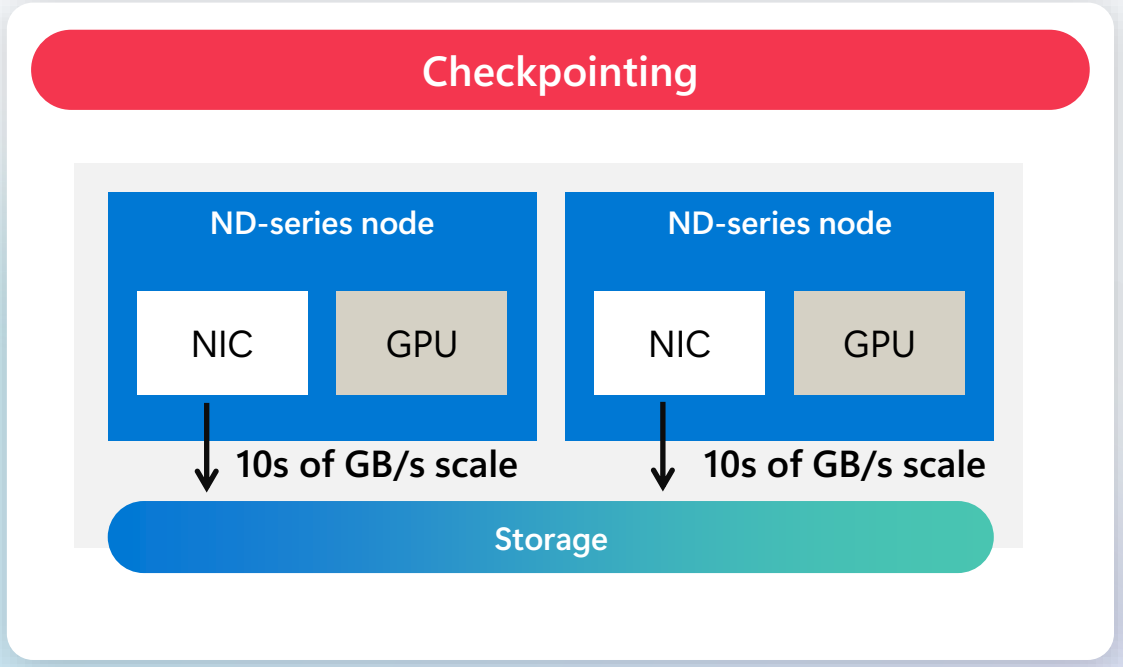


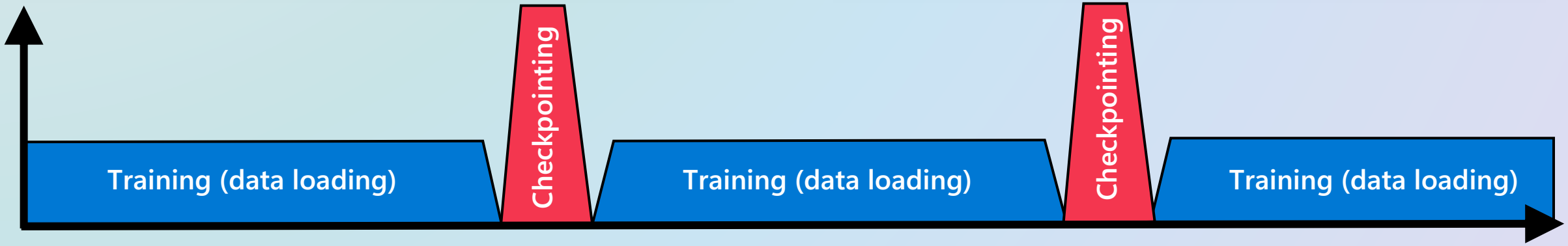
Epoch AI (2025), “Data on AI Models". *Published online at epoch.ai.* Retrieved from: 'https://epoch.ai/data/ai-models' [online resource]

Storage performance in AI Training

GPU is the most expensive resource in the datacenter







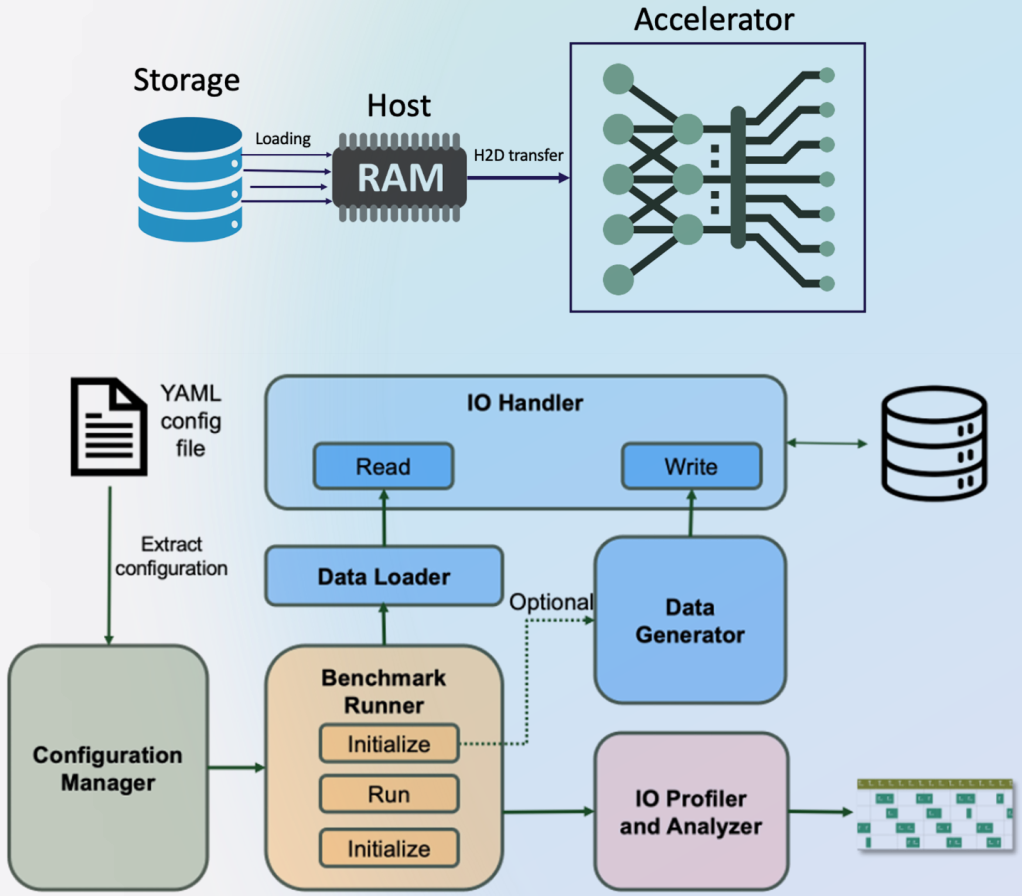
DLIO: Simulating AI Workloads without GPUs

What is DLIO?

The Deep Learning IO (DLIO) benchmark is a Python package developed by ANL and replicates real data access patterns in AI workloads using **general purpose compute.**

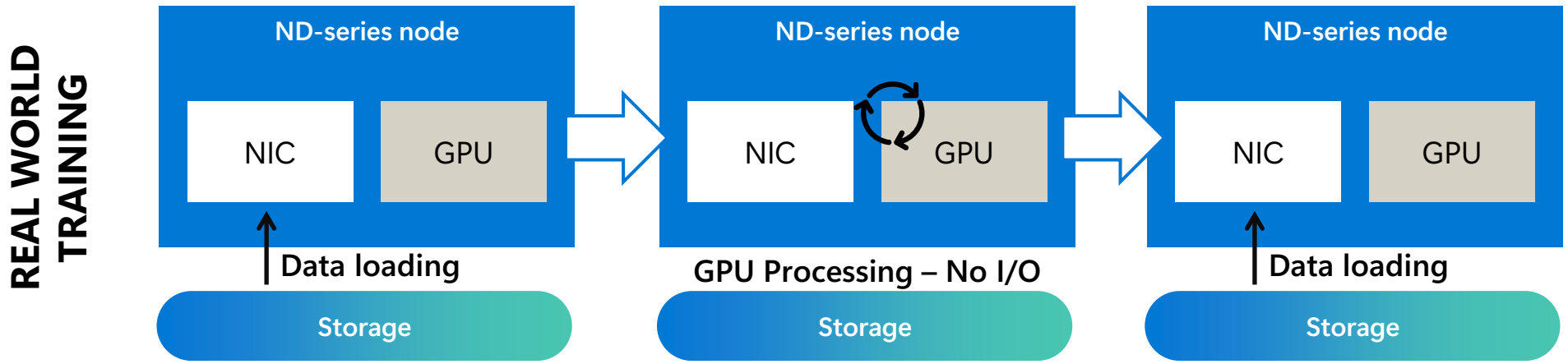
It is meant to be a benchmark that can be run easily by storage vendors, to assess **without expensive AI infrastructure** the performance of their storage solution.

dlio-benchmark.readthedocs.io

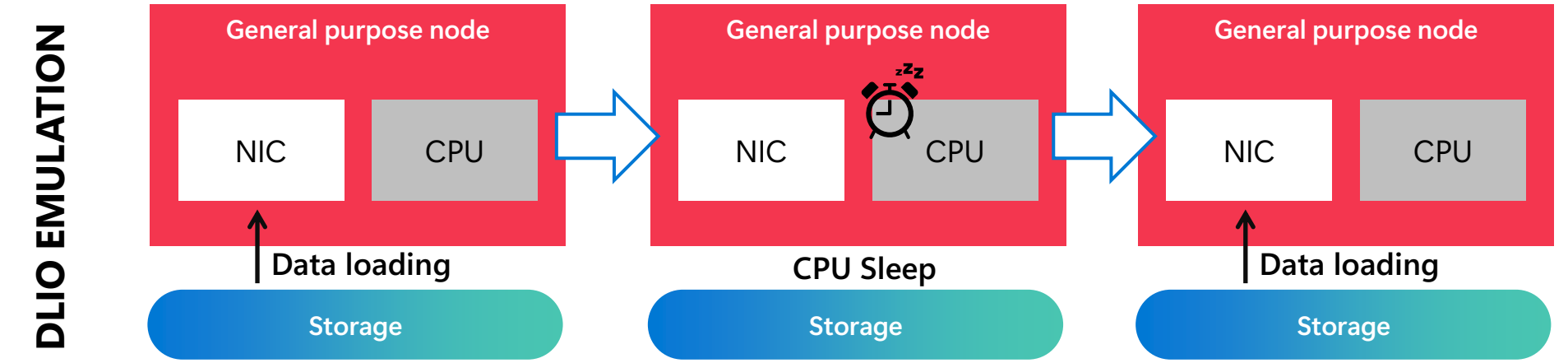


DLIO Principle

Emulating GPU I/O patterns without GPUs



In each step, all GPU nodes read an amount of data equal to batch-size and they perform the gradient optimization in the training step. Then they move to next batch and to the next step



DLIO simulates for specific training cases the GPU processing time of a specific GPU type and of a defined batch-size **with sleep commands.** This makes DLIO **executable on general purpose compute**.

MLPerf Storage: A Standardized AI Benchmark atop DLIO

Why MLPerf Storage?

On top of DLIO library, MLCommons has developed the **MLPerf Storage** benchmark suite. **MLPerf Storage** adds a standardization layer on top of DLIO.

This suite is continuously updated, with periodic submission rounds. The code base is **locked prior to each submission round** with a number of defined AI workloads to be assessed.

mlcommons/storage: MLPerf® Storage

MLPerf Storage v2.0

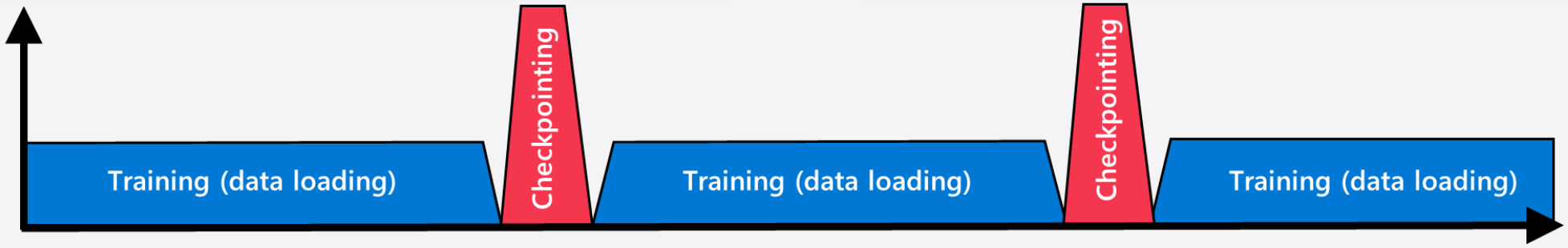
Last version's benchmarks

Training (data loading)

Problem	Model	Data Loader	Dataset seed	Min AU%
Image segmentation (medical)	3D U-Net	PyTorch	KiTS 19 (140MB/sample)	90%
Image classification	ResNet-50	TensorFlow	ImageNet (150KB/sample)	90%
Cosmology	parameter prediction	TensorFlow	CosmoFlow N-body simulation (2MB/sample)	70%

Checkpointing

Model	8B	70B	405B	1T
Hidden dimension	4096	8192	16384	25872
FFN size	14336	28672	53248	98304
num_attention_heads	32	128	128	192
num_kv_heads	8	8	8	32
Num layers	32	80	126	128
Checkpoint size	105 GB	912 GB	5.29 TB	18 TB
Subset: 8-Process Size	105 GB	114 GB	94 GB	161 GB



Understanding the Benchmark Results: Training

MLPerf Results

MLCommons data as of: 8/4/2025 5:11:37 PM

Benchmark: **Storage**
Division/Power: **Closed**
Availability: **Available**
Round: **v2.0**

Storage System Type

Cloud: Any file or block deployment in a public cloud
Direct-Attached Block: Block device attached inside a compute-node via PCIe, eg: NVMe
Fabric-Attached Block: Block device attached to the compute node via any form of networking, eg: NVMeoF
Shared File: A network-attached shared file store
Local File + Cloud: A combination of a direct-attached block device in the compute node in the cloud plus any block or file solution in a cloud
Local File + Shared File: A combination of a direct-attached block device in the compute node plus a network-attached shared file store ..

Integrated Client Storage

If the storage solution supports using storage drives that are present inside the compu clients, this describes the total number of such drives used in this benchmark results.

Checkpoint Mode

Does this benchmark result show a 'full' checkpoint written and read by all the clients r the simulated LLM training, or does it show a 'subset' checkpoint where because the st solution architecture is linearly scale-out only one client wrote and read it's portion of th checkpoint...

Public ID	Organization	Availability	Storage System Name	System Description PDF	Storage System Type	Storage System RU	Integrated Client Storage	Accelerator Type	# Client Nodes
v2.0-0012	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0013	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0014	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0015	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0016	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0017	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0018	Alluxio	Available	aws_c5n9xlarge_i3en12xla..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0019	Alluxio	Available	aws_c6in12xlarge_i3en12xl..	https://github.com/mlcomm..	Shared File + Cloud			H100	0
v2.0-0022	DDN	Available	AI400X3_24x7TiB_nvme_4..	https://github.com/mlcomm..	Shared File	2		H100	15
v2.0-0023	DDN	Available	AI400X3_24x7TiB_nvme_4..	https://github.com/mlcomm..	Shared File	2		H100	15

Understanding the Benchmark Results: Training (2)

Benchmark / Model MLC / Units					
Storage					
3D U-Net		CosmoFlow		ResNet-50	
# Simulated Accelerators	Read B/W (GiB/s)	# Simulated Accelerators	Read B/W (GiB/s)	# Simulated Accelerators	Read B/W (GiB/s)
				208	35.8408
320	444.023				
60	159.945				
		96	53.9434		
48	70.8897				
26	74.6594				
38	103.992				
136	190.717				
76	216.543				
255	698.396				
		80	41.9909		
				1144	202.972

Understanding the Benchmark Results: Checkpointing

			Benchmark / Model MLC / Units			
			age			
			Llama3_405b			
Usable Capacity (TiB)	Checkpoint Mode	# Client Nodes	Write B/W (GiB/s)	Write Duration (secs)	Read B/W (GiB/s)	Read Duration (secs)
30000	Full	1				
30000	Full	64	866.762	6.10824	581.434	9.4482
30000	Full	8				
102400	Full	128				
102400	Full	1				
102400	Full	64	236.213	22.5364	237.378	22.3947
102400	Full	8				
	Full	0				
	Full	0				
120	Full	1				
42	Full	1				
184.32	Subset		11.4721	8.59262	20.2544	4.92957

Not Another Marketing Benchmark

- Storage solutions are extremely varied:
  - Direct-attached block devices to cloud native shared file solutions with client local caches
  - Capacities from 6TiB to 102PiB
  - Throughputs from 2GiB/s to 700GiB/s

- You cannot (easily) sort the list to highest throughput
  - Intent is to demonstrate exemplary numbers for various solutions of different sizes – not to simply be weaponized by marketing departments
  - Not even required to submit to all benchmarks – some only publish checkpointing, some only training, and some just a subset of either category
  - System designers can then select the best storage solution for their accelerator type, count, and location

I/O Patterns in MLPerf Storage 2.0

Detailed Training Workload Characteristics

3D U-Net
Vision – Medical Image Segmentation

- Min file sizes and counts:
  - 3500 files @ 140MiB each
  - 479GiB / accelerator
- Bandwidth requirements
  - A100: 1.5GiB/s
  - H100: 3GiB/s
- I/O Patterns:
  - 512KiB read requests
- **Total BW Hog**

CosmoFlow
Scientific – Cosmology parameter prediction

- Min file sizes and counts:
  - 500 files at 2.7MiB each
  - 1.3GiB / accelerator
- Bandwidth requirements
  - A100: 0.32GiB/s
  - H100: 0.51GiB/s
- I/O Patterns:
  - 170KiB read requests
- **Very latency sensitive**

ResNet-50
Vision – Image classification

- File sizes and counts:
  - 159 files @ 136MiB each
  - 21GiB / accelerator
- Bandwidth requirements
  - A100: 90MiB/s
  - H100: 192MiB/s
- I/O Patterns:
  - 170KiB read requests
- **Multiple reads/file**

Training Benchmark Key Take-aways

- Dataset size is scaled against client memory
  - It is legitimate to artificially reduce the client memory to bound the test
- Dataset generation immediately precedes benchmarking
  - Storage subsystem caches, both GPU-local and remote, can have a big impact
- Achieving on all three workloads requires a powerful filesystem:
  - High bandwidth (100s of GiB/s for a few dozen GPUs and 3D UNet)
  - Moderate capacity (iff VMs have low memory, 10s of TiB at minimum for 3D UNet)
  - Low latency (consistently low single-digit millisecond for CosmoFlow)

MLPerf Storage Version 2.0 vs 3.0

MLPerf Storage v3.0 – What's new?

Training

MLPERF STORAGE v2.0

Problem	Model	Data Loader	Dataset seed	Min AU%
Image segmentation (medical)	3D U-Net	PyTorch	KiTS 19 (140MB/sample)	90%
Image classification	ResNet-50	TensorFlow	ImageNet (150KB/sample)	90%
Cosmology	parameter prediction	TensorFlow	CosmoFlow N-body simulation (2MB/sample)	70%

Metric	UNET3D	RESNET-50	Cosmoflow
Sample size [bytes]	146,600,628	114,660	2,828,486
Samples per file	1	1,251	1
File Size [MiB]	140	137	3
Batch size	7	400	1
Compute time A100 [s]	0.63600	0.43500	0.00551
Compute Time H100 [s]	0.32300	0.22400	0.00350
BW per Accelerator for AU 100% A100 [MB/s]	1,614	105	513
BW per Accelerator for AU 100% H100 [MB/s]	3,177	205	808

MLPERF STORAGE v3.0

Problem	Model	Data Loader	Dataset seed	Min AU%
Recommendation Model	DLRMv2	PyTorch (parquet)	3.3 GiB/sample	70%
Image generation	FLUX	PyTorch (parquet)	500MiB/sample	90%
Object detection	Retinanet	PyTorch (JPG)	322KiB/sample	85%

Metric	FLUX	DLRM	Retinanet
Sample size [bytes]	2,164,832	760	322,957
Samples per file	256	4,718,592	1
File Size [MiB]	529	3,420	0.8
Batch size	48	12,288	24
Compute time GB200 [s]	1.35000	0.00038	0.04755
Compute Time MI355X [s]	2.02500	0.00056	0.07133
BW per Accelerator for AU 100% GB200 [MB/s]	77	24,905	163
BW per Accelerator for AU 100% MI355 [MB/s]	51	16,603	109

Accelerators

B200 + MI355

Readers

Parquet + JPG

Models

3 new models

MLPerf Storage v3.0 – What's new?

Checkpointing

Checkpointing

Same limitation of min 4 GPU per node

The **direct_io** options has been implemented in checkpointing, delivering interesting possibilities

Data generation has been improved, reducing the tensor generation time

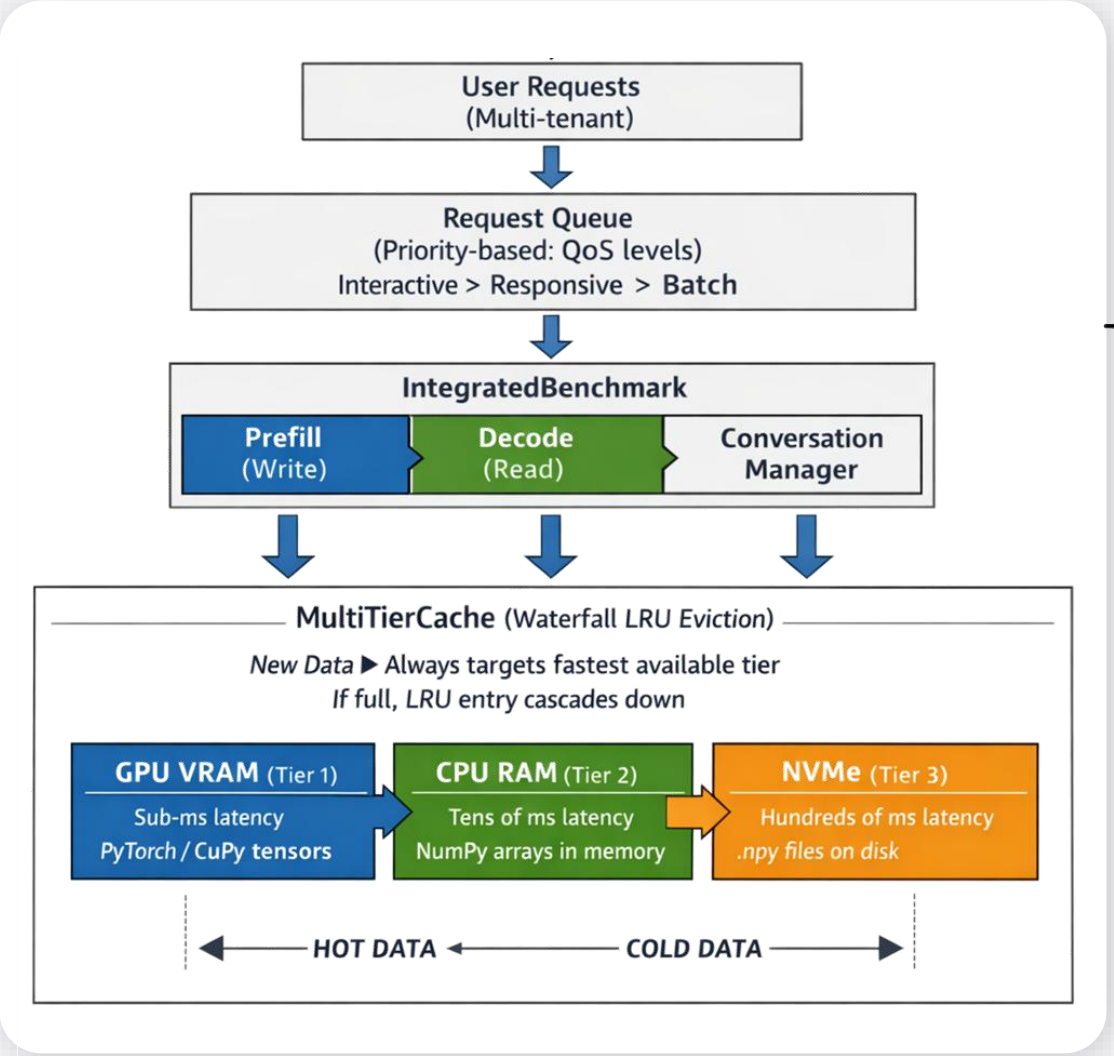
Performance even without direct_io improved

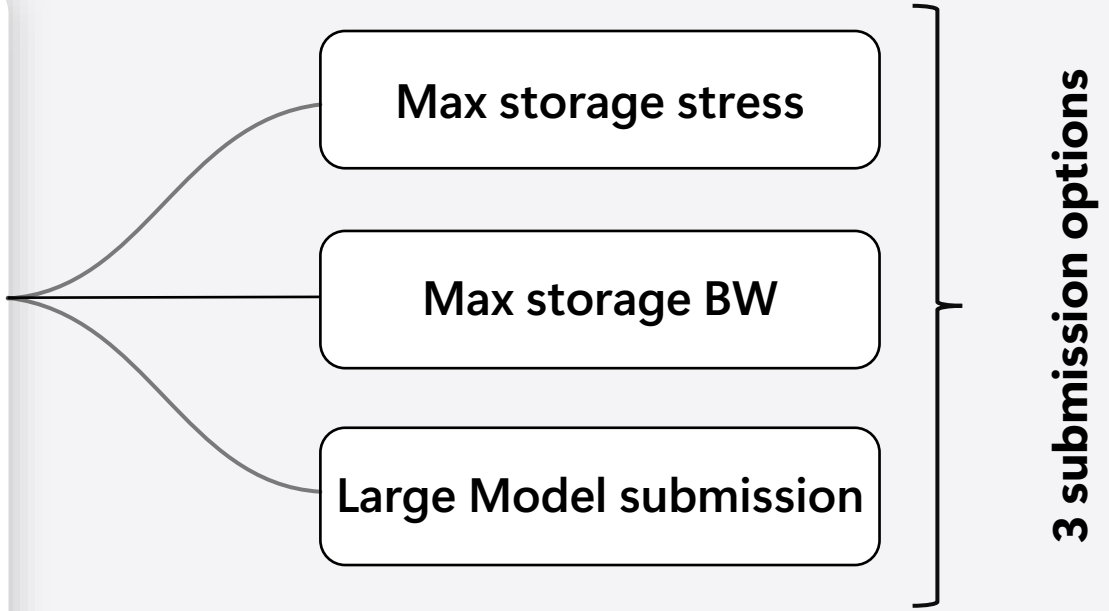
LLAMA3

Model	8B	70B	405B	1T
Hidden dimension	4096	8192	16384	25872
FFN size	14336	28672	53248	98304
num_attention_heads	32	128	128	192
num_kv_heads	8	8	8	32
Num layers	32	80	126	128
Checkpoint size	105 GB	912 GB	5.29 TB	18 TB
Subset: 8-Process Size	105 GB	114 GB	94 GB	161 GB

MLPerf Storage v3.0 – What's new?

KV-Cache





PRIMARY METRICS for MLPerf Storage Comparison:

- **Storage Tier Read Device P95 (ms)**
- **Storage Tier Write Device P95 (ms)**
- **Tier Storage Read Bandwidth (GB/s)**
- **Tier Storage Write Bandwidth (GB/s)**

storage/kv_cache_benchmark at main · mlcommons/storage

Running MLPerf Storage on Lustre

MLPerf Storage Setup and Execution

```
# On the main node, the datasize subcommand computes dataset required:
$ mlpstorage training datasize -m unet3d --client-host-memory-in-gb 768 --max-accelerators 8 --num-client-hosts 4 --accelerator-type a100  --results-dir ~/mlps-results
```

2025-09-24 04:59:52|RESULT: Minimum file count dictated by dataset size to memory size ratio.

2025-09-24 04:59:52|RESULT: Number of training files: 112500

2025-09-24 04:59:52|RESULT: Number of training subfolders: 0

2025-09-24 04:59:52|RESULT: Total disk space required for training: 15359.90 GB

```
# It will also emit the command to run to generate the data
```

```
# Then generate the dataset:
$ mlpstorage training datagen --hosts 10.50.32.16 10.50.32.14 10.50.32.20 10.50.32.10 --model unet3d --num-processes 64 --param dataset.num_files_train=112500 --results-dir ~/mlps-results --data-dir /lustre/
```

```
# And finally run the benchmark:
$ mlpstorage training run --hosts 10.50.32.16 10.50.32.14 10.50.32.20 10.50.32.10 --model unet3d -na 8 -cm 32 -g a100 --param dataset.num_files_train=112500 --results-dir ~/mlps-results --data-dir /lustre/
```

Final Thoughts

Final Thoughts: Lustre, AI, and MLPerf Storage

- Correcting and maintaining Lustre's perception in the AI conversation requires more than a one-time OpenSFS push
- The MLPerf Storage benchmark is a great way Lustre users and vendors can help show Lustre is ready and able to serve AI I/O
  - The list isn't just about the "top" N like IO500
  - You can pick and choose which benchmarks you submit to
  - Already some Lustre submissions in the results for v2.0 – should be plenty more for v3.0
- Lustre developers should pay attention to the benchmark I/O
  - If the MLPerf Storage benchmark developers are doing it right, the benchmark very closely approximates how most people are doing AI on storage
  - Feature prioritization for the community should be evaluated against highest impact without overfitting (e.g., PCC-RO for MLPerf Storage would be highly effective)

Thank you! Questions?