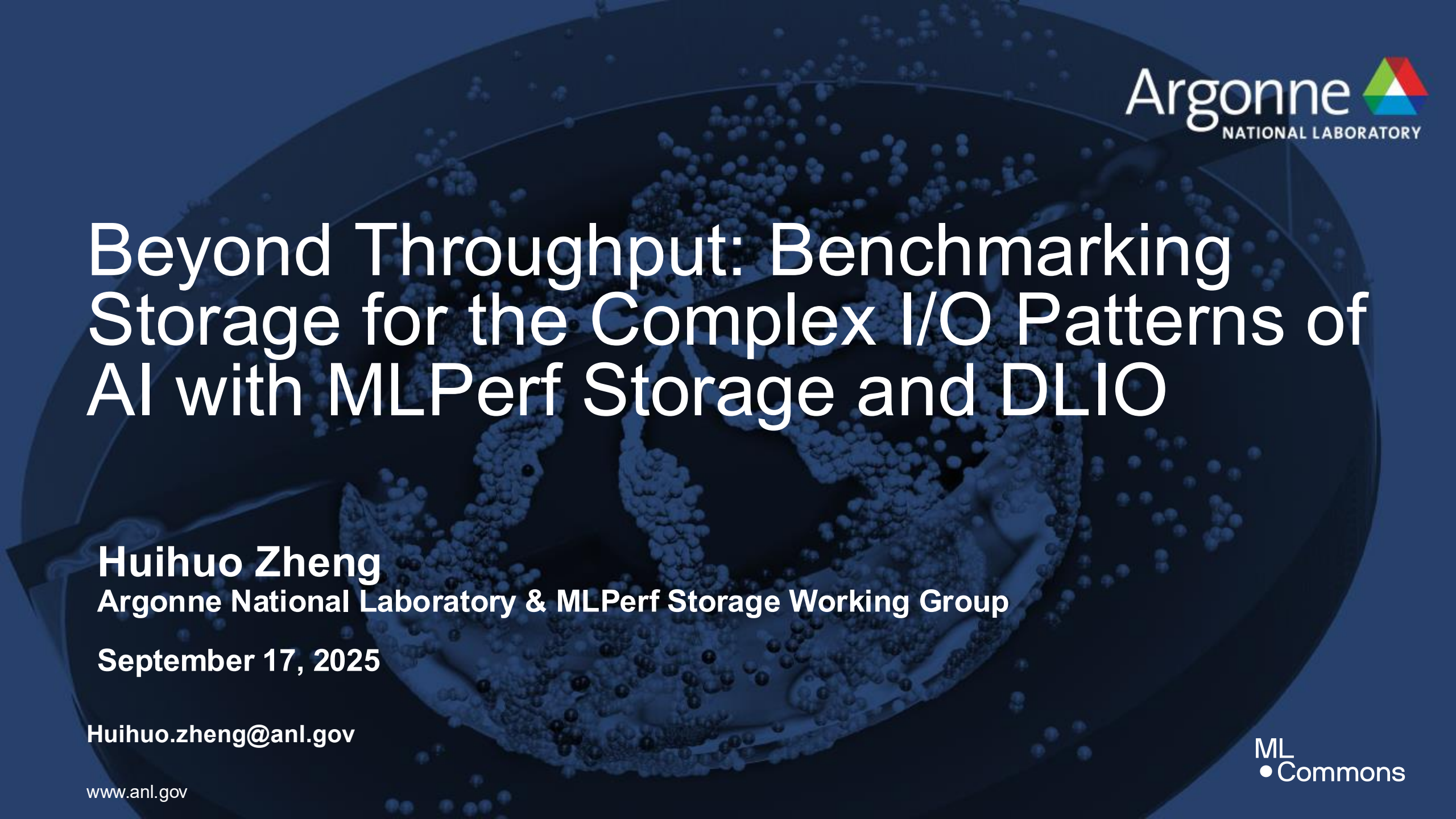


Beyond Throughput: Benchmarking Storage for the Complex I/O Patterns of AI with MLPerf Storage and DLIO

Huihuo Zheng
Argonne National Laboratory & MLPerf Storage Working Group

September 17, 2025

Huihuo.zheng@anl.gov

Argonne Leadership Computing Facility (ALCF)

ALCF, a U.S. Department of Energy (DOE) Office of Science User Facility located at Argonne National Laboratory, enables breakthroughs in science and engineering by providing supercomputing resources and expertise to the research community.

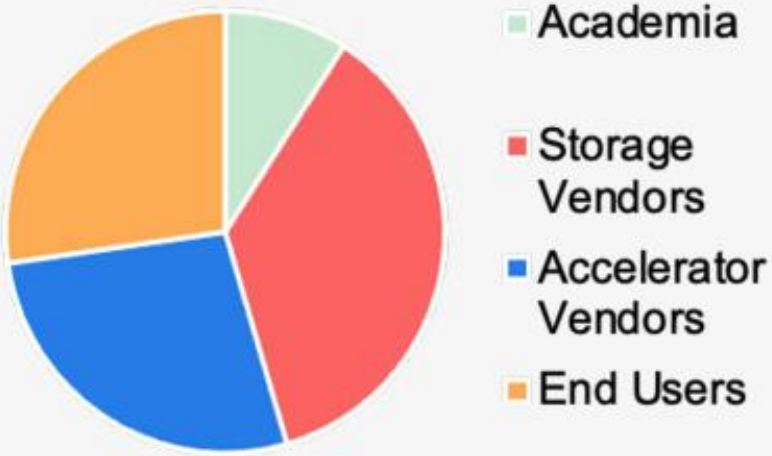
Aurora
63,744 Intel PVC GPUs
DAOS: 220PB @ 30 TB/s

Polaris
2240 Nvidia A100 GPU
Lustre: 100 PB @ 650GB/s

AI testbeds
CS-2, SambaNova,
Graphcore, GroqRack

MLPerf Storage Working Group

Who are we?

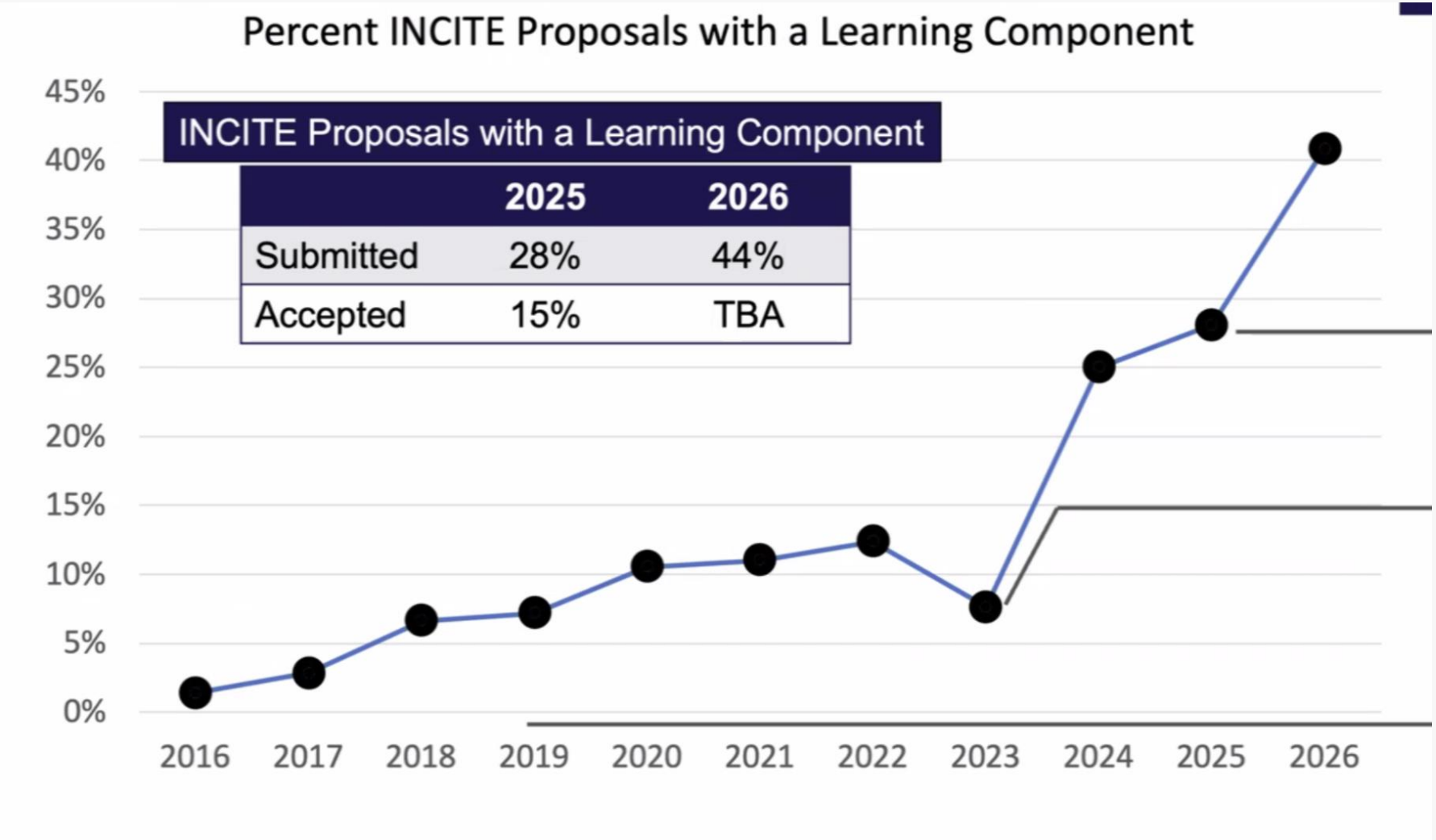


Get involved
https://github.com/mlcommons/storage
storage-chairs@mlcommons.org
storage@mlcommons.org

Outline

- Storage and I/O for AI
- Design of DLIO and MLPerf Storage Benchmarks
- Lessons learned from DLIO for AI training and checkpointing
- Future works

Growth of AI in Science (AI4Science)

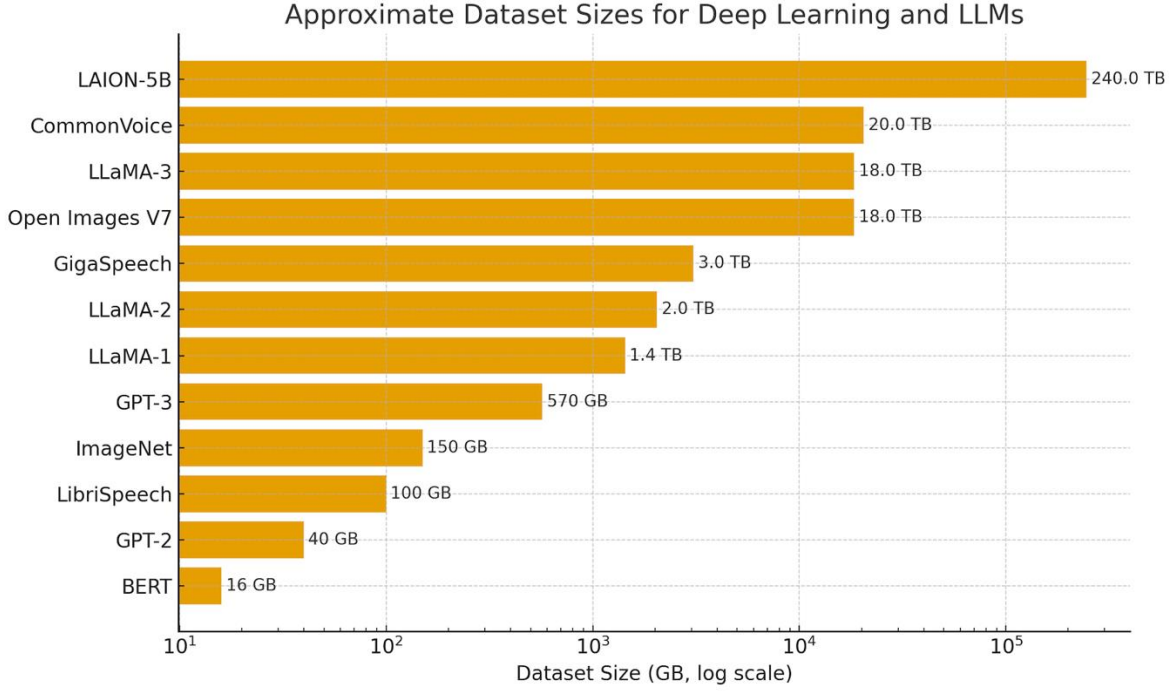


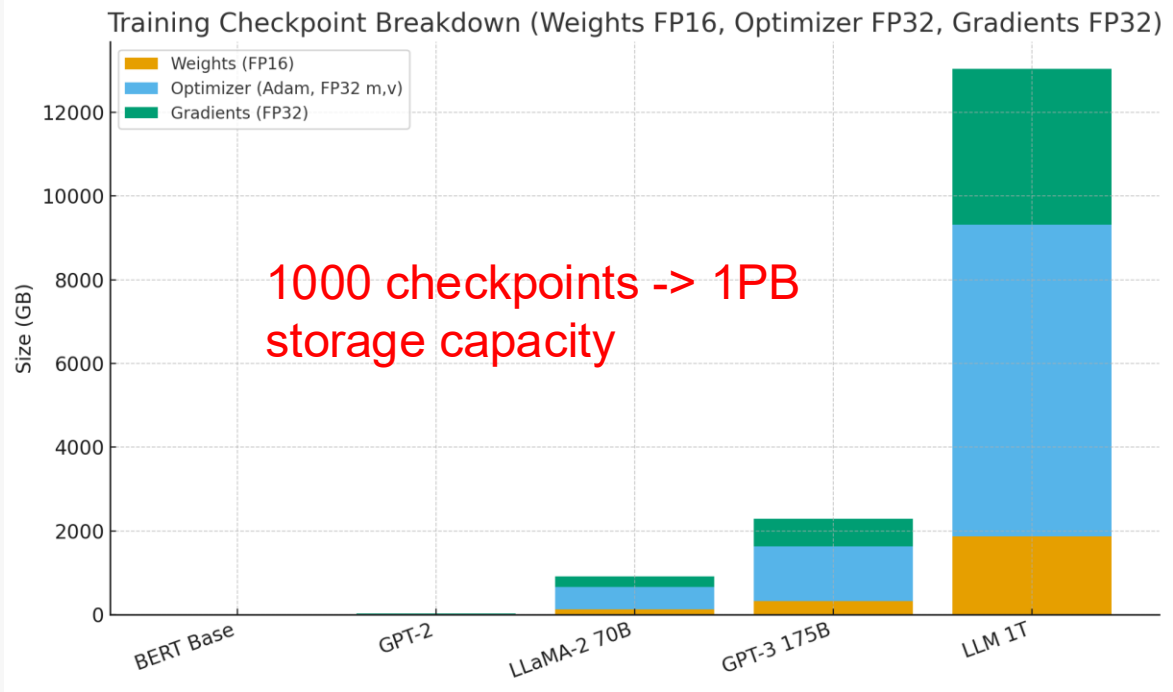
INCITE Proposals with a Learning Component

	2025	2026
Submitted	28%	44%
Accepted	15%	TBA

https://www.alcf.anl.gov/science/incite-allocation-program

Need for storage: large amount of training datasets and checkpoints





I/O Patterns: HPC vs AI

HPC workloads

- Write intensive (checkpointing)
- Bulk parallel I/O
- Multi-dimensional array (binary, HDF5, NetCDF)
- Single thread & sync. I/O
- Global file system + SSD (but SSD are rarely been used)

AI workloads

- Read intensive
- Small and sparse I/O & metadata intensive random access
- Complex data format (json, text, key-value)
- Multithreaded async. I/O
- Storage hierarchy + caching & staging
- Variety of needs for different stages: preprocessing, data loading, checkpointing, inferencing

Datasets and I/O access patterns of AI workloads

Workload	Domain	Sample size	Total size	I/O stress pattern
3D-UNet	Biomedical	100–150 MB (NPZ)	0.1–2 TB	Bandwidth + moderate metadata
ImageNet	Computer vision	~200 KB (JPEG)	150 GB	Many small reads + metadata
CosmoFlow	Cosmology	8–64 MB (HDF5) 2MB tfrecord	0.5–8 TB	Large contiguous reads, high BW
DeepCAM	Climate science	200–300 MB (netCDF/HDF5)	1 TB	Bandwidth -intensive
FFN	neuroscience EM,	16–64 MB (HDF5 or tfrecord)	TB - PB	Bandwidth (sub-volume access)
LLM training	Language	1–4 KB tokens (Binary index / text)	1–20 TB	Many small random reads
LLM Checkpoint	Language	Entire model (torch tensor)	1 GB – 16 TB	Large sequential writes

We need a storage benchmark for AI

- Understand storage access pattern in AI workloads
- Identify bottlenecks and propose optimizations (SW & HW)
- Help AI/ML researchers and practitioners make an informed *storage* decision
- Guide storage and I/O software and hardware codesign

FIO (Flexible I/O Tester) is a versatile benchmarking tool for measuring storage performance on a single node.

IOR: a parallel I/O benchmark used to measure the performance and scalability of HPC storage systems.

DLIO: Deep Learning I/O Benchmark:

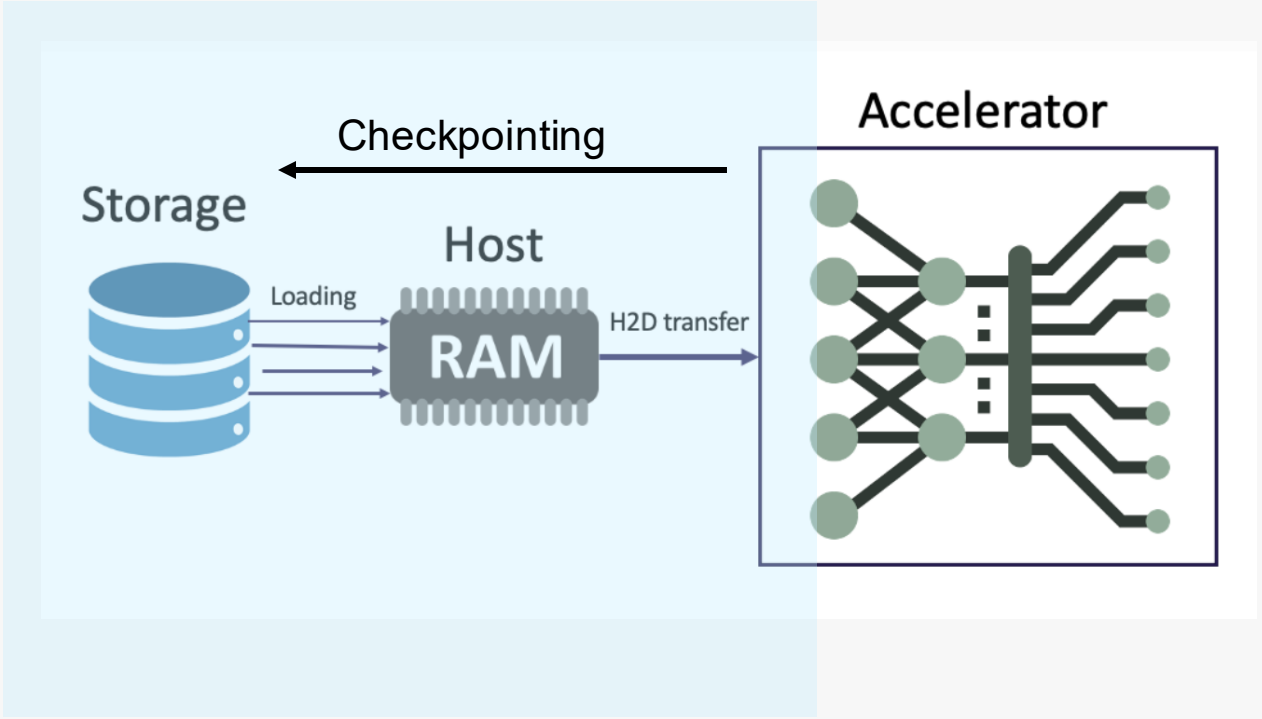
- Using actual framework data loading library, such as panda, h5py, and numpy
- Configurable framework setting: Multiple threading, sample size, training epochs / steps, realistic AI dataset layout

H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.
https://github.com/argonne-lcf/dlio_benchmark

Design of DLIO

AI training pipeline

- Data preparation
- Loading data from the storage to the host and preprocessing (CPU initiated)
- Transferring data to the accelerators
- Training / inference on the accelerators
- Checkpointing

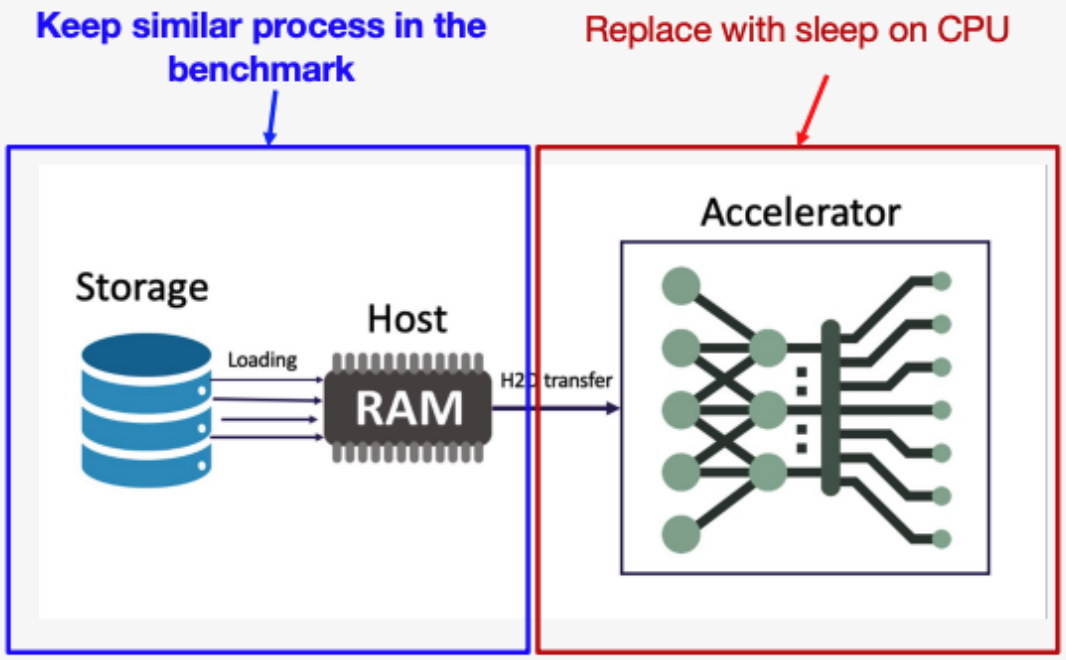


H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.
https://github.com/argonne-lcf/dlio_benchmark

Design of DLIO

Main features

- NO need of actual accelerators to run (replacing computation with sleep)
- Represent I/O patterns of realistic ML workloads by directly using framework dataloader
- Support data generation
- Configurable for different workloads
- Holistic performance monitoring and tracing across all software stacks



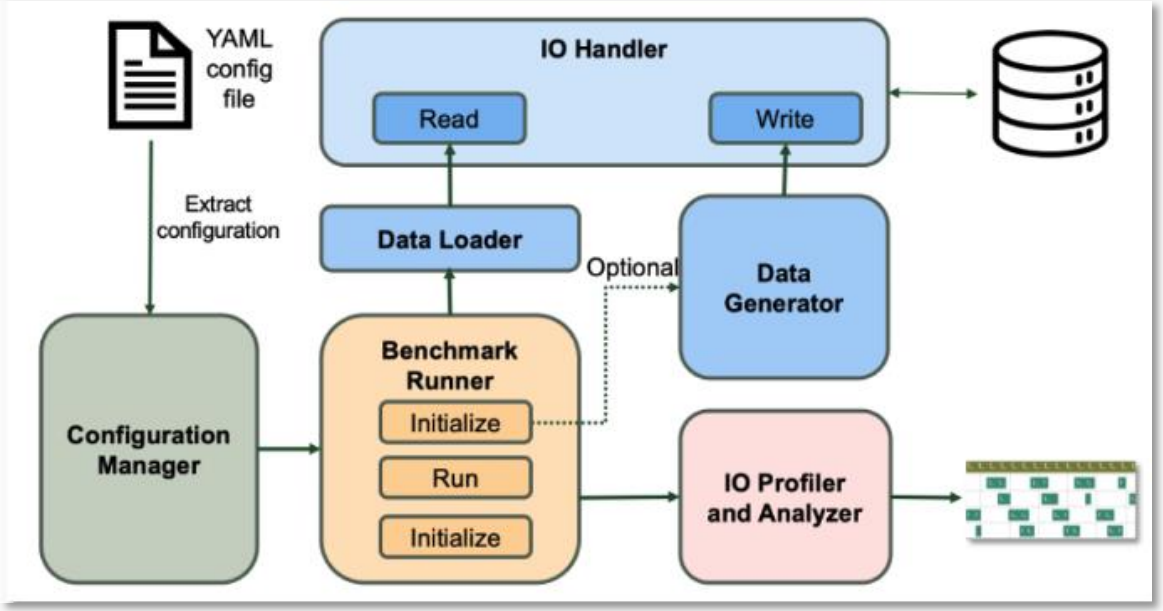
Schematic of the training process

H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.
https://github.com/argonne-lcf/dlio_benchmark

Design of DLIO

Main features

- NO need of actual accelerators to run (Replacing computation with sleep)
- Represent I/O patterns of realistic ML workloads by directly using framework dataloader
- Configurable for different workloads
- Support data generation
- Holistic performance monitoring and tracing across all software stacks



H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.
https://github.com/argonne-lcf/dlio_benchmark

Design of DLIO

Main features

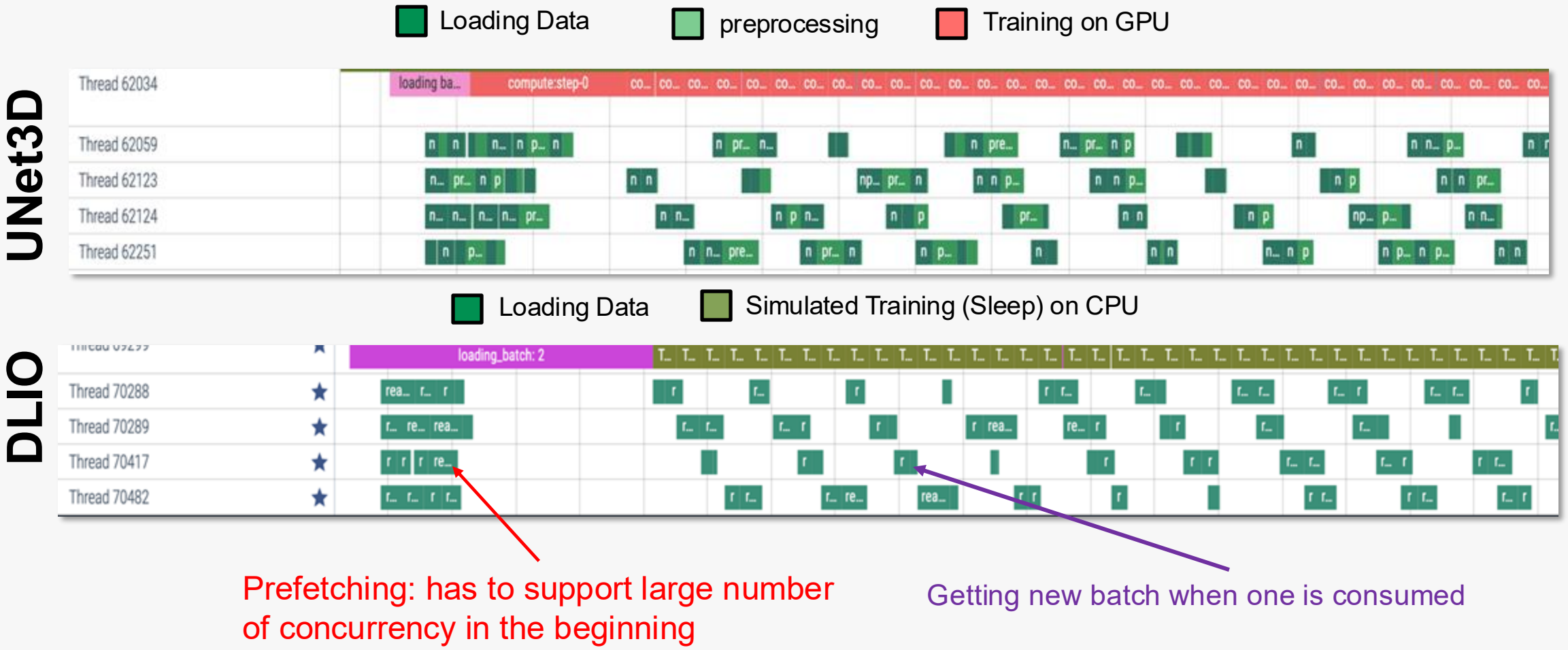
- NO need of actual accelerators to run (Replacing computation with sleep)
- Represent I/O patterns of realistic ML workloads by directly using framework dataloader
- Configurable for different workloads
- Support data generation
- Holistic performance monitoring and tracing across all software stacks

```yaml
model:
  name: unet3d
  type: cnn
  model_size: 499153191

framework: pytorch

workflow:
  generate_data: False
  train: True
  checkpoint: True

dataset:
  data_folder: data/unet3d/
  format: npz
  num_files_train: 168
  num_samples_per_file: 1
  record_length_bytes: 146600628
  record_length_bytes_stdev: 683418
  record_length_bytes_resize: 20971

reader:
  data_loader: pytorch
  batch_size: 7
  read_threads: 4
  file_shuffle: seed
  sample_shuffle: seed

train:
  epochs: 5
  computation_time: 0.636
```

```yaml
model:
  name: llama_405b
  type: transformer
  num_layers: 126
  model_datatype: fp16
  optimizer_datatype: fp32
  parallelism:
    tensor: 8
    pipeline: 32
    zero_stage: 1
  transformer:
    vocab_size: 128256
    hidden_size: 16384
    ffn_hidden_size: 53248
    num_attention_heads: 128
    num_kv_heads: 8

framework: pytorch

workflow:
  generate_data: False
  train: False
  checkpoint: True

checkpoint:
  checkpoint_folder: checkpoints/llama_405b
  time_between_checkpoints: 5
  num_checkpoints_write: 10
  num_checkpoints_read: 10
  fsync: True
```

H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.
https://github.com/argonne-lcf/dlio_benchmark

DLIO simulates I/O pattern of real workloads



Experiment setup: 1 V100 GPU @ JLSE

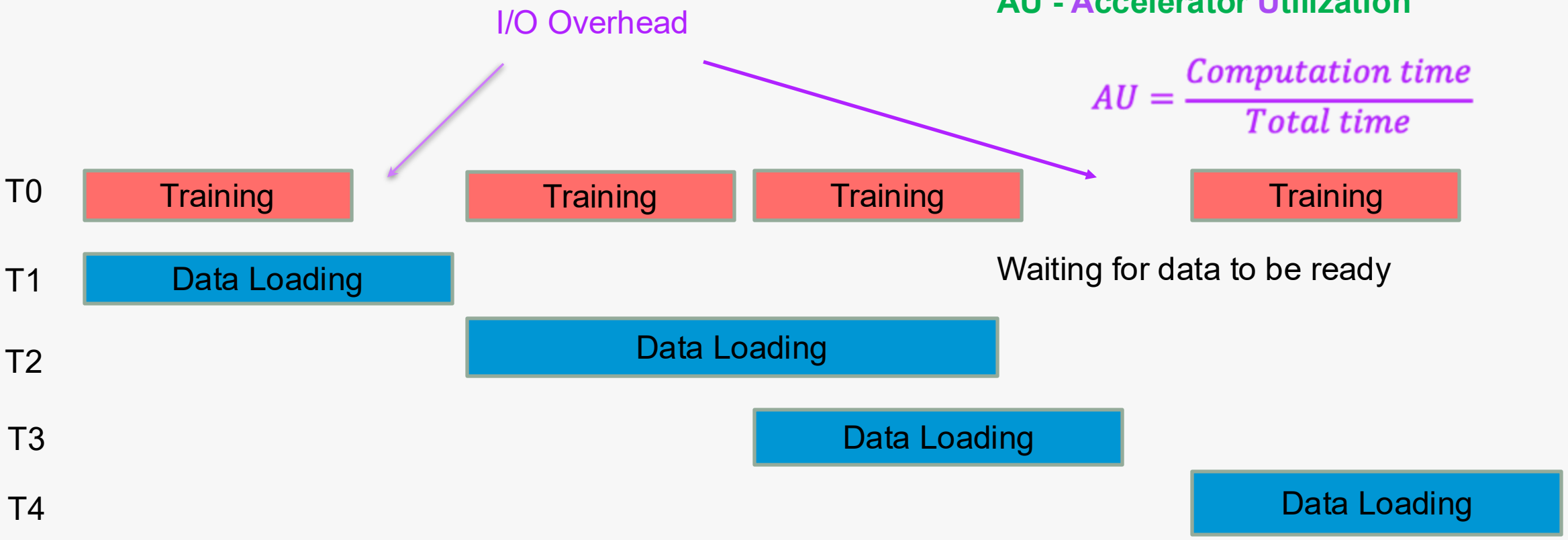
Plots were generated with: https://ui.perfetto.dev/

Metric of the benchmark

$$Throughput = \frac{num\_samples}{Time\ per\ epoch}$$

AU - Accelerator Utilization

$$AU = \frac{Computation\ time}{Total\ time}$$



Goal of benchmark: maximize samples / second, given a Accelerator Utilization > 90% at a certain scale.

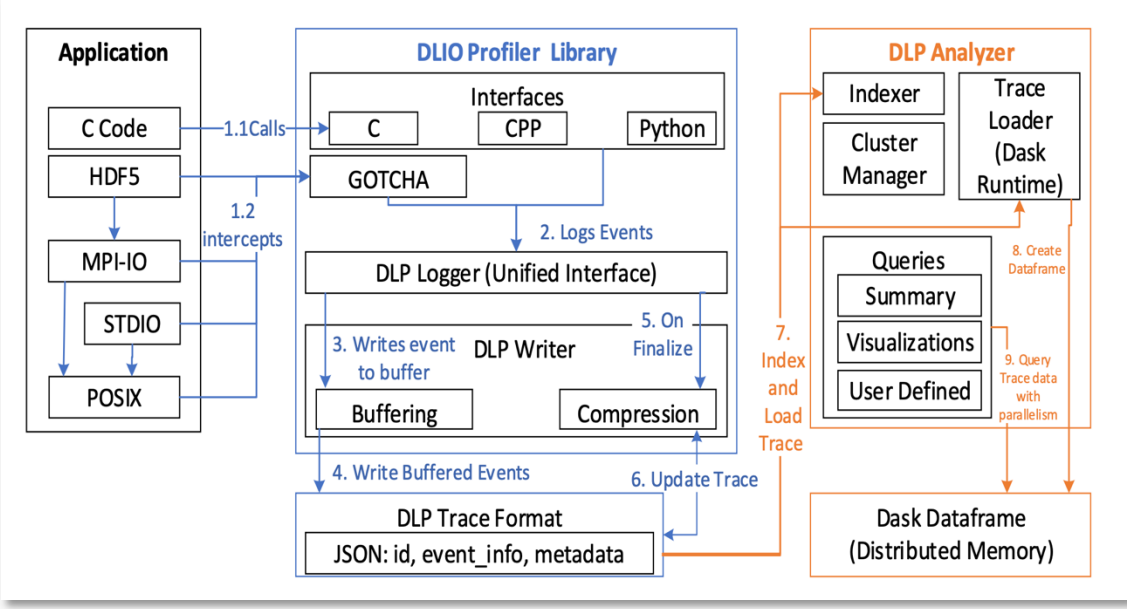
I/O Profiler and Tracer for AI: DFTracer

Diagnose the form, reveal the root, tune the whole.
诊其形，明其本，调其全。

Existing tools are not friendly to AI: darshan, recorder, perf, ioprof

We need a AI I/O profiler / tracer

- Easy to use
- Works with Python applications (multiple threads)
- Able to profile at different levels
- Easy to integrate with other tracing tools
- Easy to visualize the tracing (*Perfetto*)
- Small overhead



https://github.com/LLNL/dftracer

H. Devarajan *et al.*, "DFTracer: An Analysis-Friendly Data Flow Tracer for AI-Driven Workflows," *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, USA, 2024, pp. 1-24,

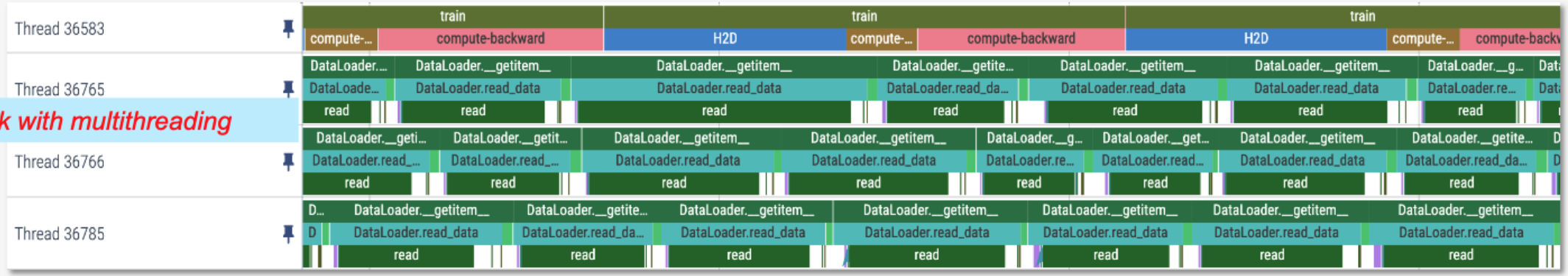
I/O Profiler and Tracer for AI: DFTracer

Easy to use

```
from dlio_profiler.logger import fn_interceptor as Profile
dlp_data = Profile("DataLoader")
class DataLoader(datasets.ImageFolder):
    @dlp_data.log
    def preprocess(self, sample, target):
        if self.transform is not None:
            sample = self.transform(sample)
        if self.target_transform is not None:
            target = self.target_transform(target)
            return sample, target
    @dlp_data.log
    def read_data(self, index):
        path, target = self.samples[index]
        return self.loader(path), target
    @dlp_data.log
    def __getitem__(self, index):
        sample, target = self.read_data(index)
        sample, target = self.preprocess(sample, target)
        return sample, target
```

Name	Wall duration (ms)
	400698.238
DataLoader.__getitem__	142903.235
DataLoader.read_data	134743.702
read	110562.695
DataLoader.preprocess	7762.795
close	2356.922
open64	2335.804
__fxstat64	22.914
lseek64	10.171

Work with multithreading

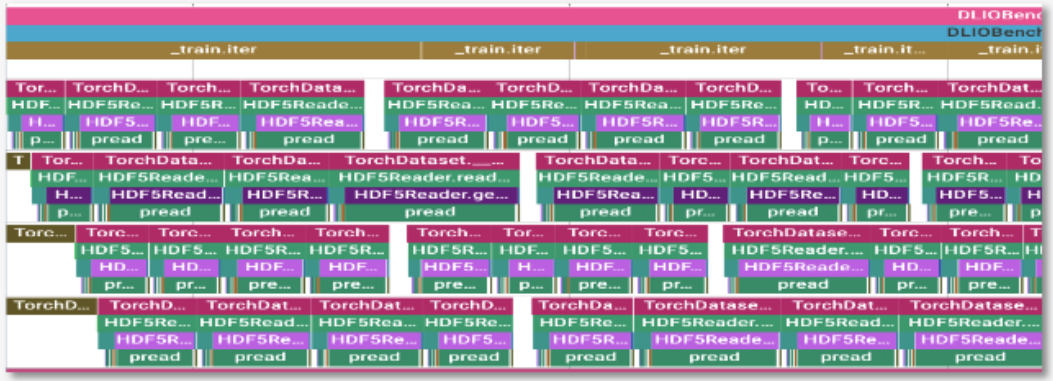


https://github.com/LLNL/dftracer

ML Commons

Argonne NATIONAL LABORATORY

Performance issue revealed through profiling



PyTorch

Name	Wall duration (ms)
	20049.987
TorchDataset.__getitem__	5042.087
HDF5Reader.read_index	5030.107
HDF5Reader.get_sample	4202.46
pread	4066.861

TensorFlow

Name	Wall duration (ms)
	17441.186
HDF5Reader.get_sample	7735.878
HDF5Reader.open	3764.007
pread	3134.925
HDF5Reader.close	2466.615

Performance comparison of Torch DataLoader and tf.data for HDF5 datasets. TensorFlow data loader performs worse for HDF5 because of thread locking issue

MLPerf Storage v2.0 – Workloads

Training Workloads

Measures the number of accelerators a given storage solution can support with a certain level of accelerator utilization (AU)

- 3D-UNet – Medical image segmentation
- ResNet-50 – Image classification
- CosmoFlow – Cosmology simulations

Checkpointing Workloads

Measures the checkpointing write and recovery for Llama3 LLM foundation training: 8B, 70B, 405B, 1T

https://mlcommons.org/2025/08/mlperf-storage-v2-0-results/

MLPerf Storage Benchmark Submissions

Version / Year	Number of Submitters	Number of Submissions
v0.5 (first round, 2023)	**5 organizations**	~28 performance results
v1.0 (2024)	**14 organizations**	**130 submissions**
v2.0 (2025)	**26 organizations**	>200 performance results

Storage Type Category	# Solutions
Local storage solutions	6
In-storage accelerators (e.g. programmable storage, computational storage)	2
Software-defined storage (SDS)	13
Block storage systems	12
On-prem shared storage systems	16
Object storage systems	2

https://mlcommons.org/2025/08/mlperf-storage-v2-0-results/

Benefit of community effort

Allowing all inclusive exploration of the landscape

- Different storage types (Spectrum, Lustre, DAOS, NVMe SSD, Cloud, etc)
- Data loader (TensorFlow, PyTorch, JAX, DALI, etc)
- Storage of data (single shared file vs multiple files)
- Compression / decompression
- Different data formats (NPZ, CSV, PNG, TFRECORD, HDF5, Parquet, etc)
- Different organization of data (single sample per file, multiple samples per file)

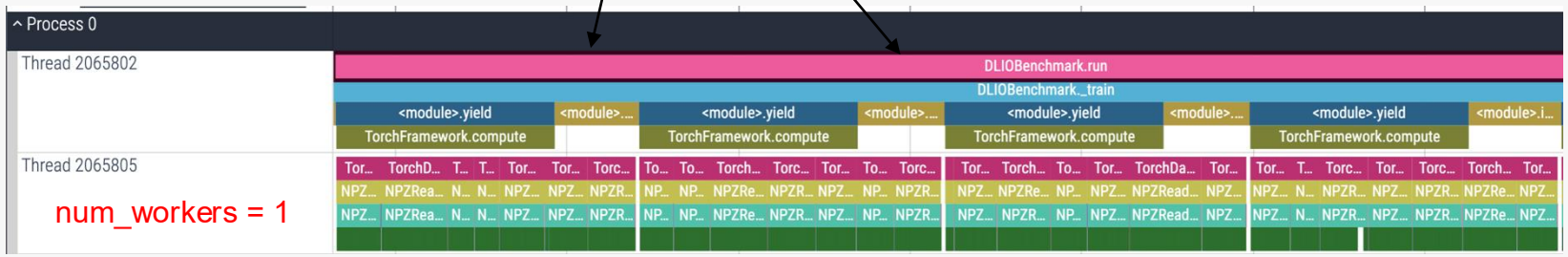
Some lessons learned with MLPerf Storage and DLIO

3D-UNet on A100 with SSD

Accelerator	A100	H100
Time (s)	0.636	0.323

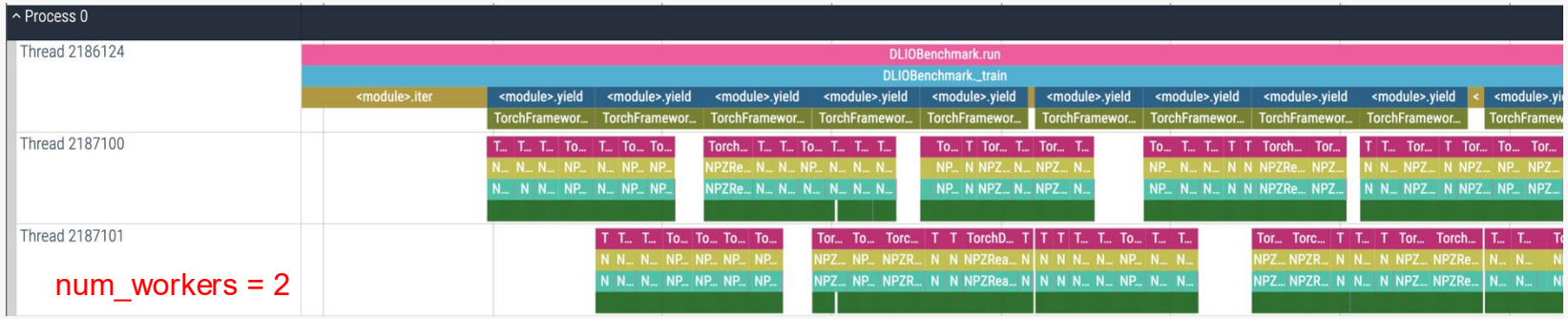
Waiting for data

Sample size: 100–150 MB (NPZ)



AU = 70%

IO / compute = 1.3, num_workers = 2 should be sufficient



AU = 98.7%

I/O threads were busy most of the time, but were able to meet the need

ML Commons

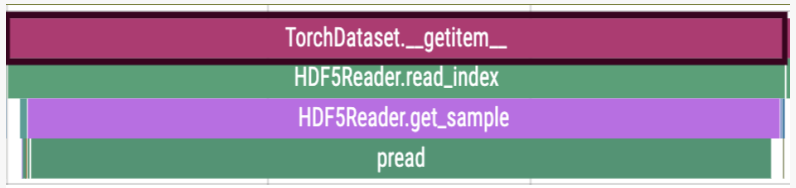
Argonne NATIONAL LABORATORY

3D UNet on H100 with SSD

Accelerator	A100	H100
Time (s)	0.636	0.323

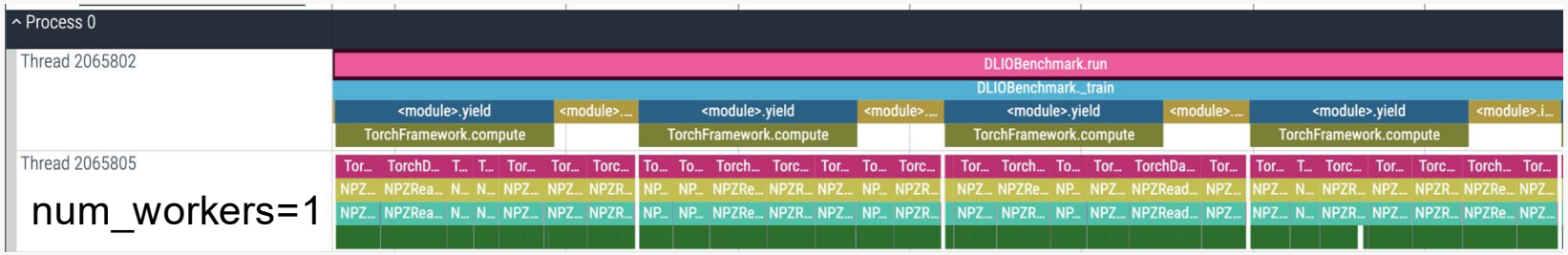
NPZ is inefficient -> changing to HDF5

Single NPZ file is loaded in chunks of 256 kB, which is very inefficient

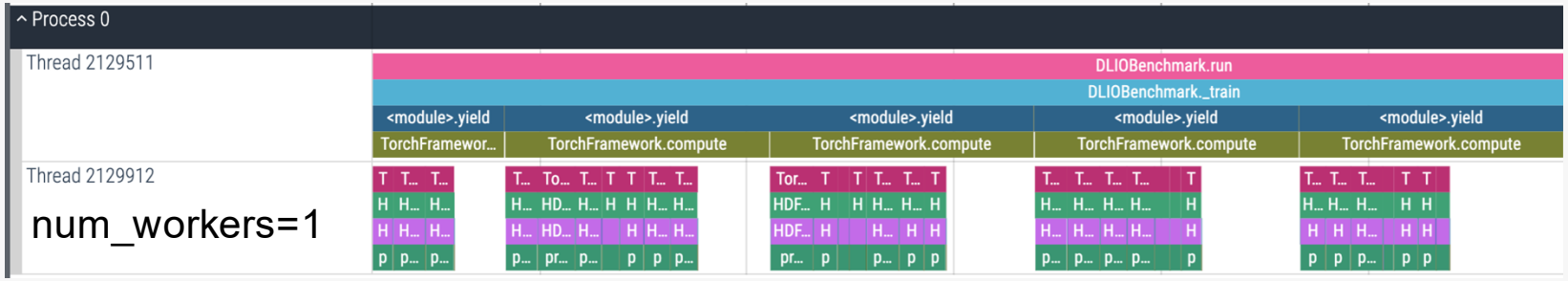


HDF5 is loaded in one chunk (~150MB)

NPZ



HDF5



With HDF5, num_workers = 1, we can achieve AU = 99% !

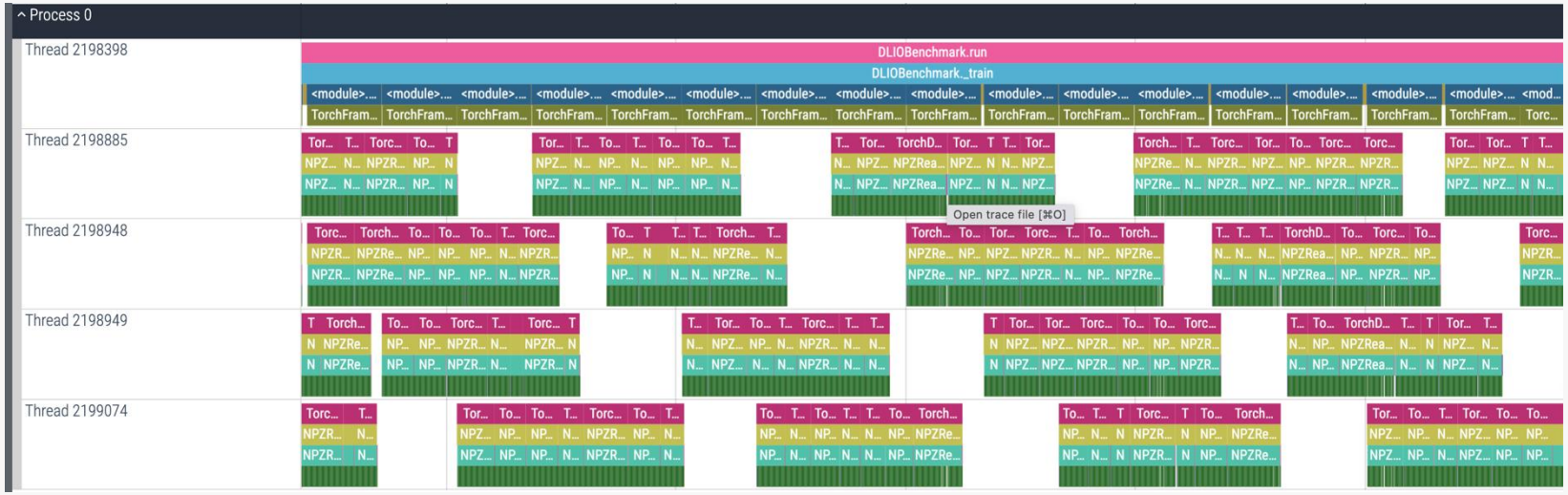
ML Commons

Argonne NATIONAL LABORATORY

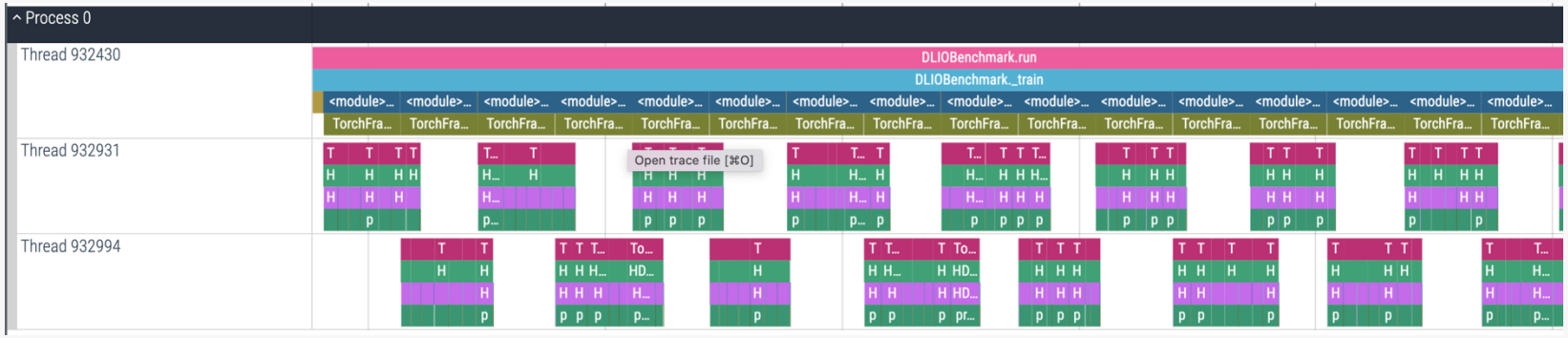
3D UNet on H100 with SSD

Accelerator	A100	H100
Time (s)	0.636	0.323

Compute is faster, we have to increase *num_workers*



NPZ format
num_workers = 4
AU = 98.6%

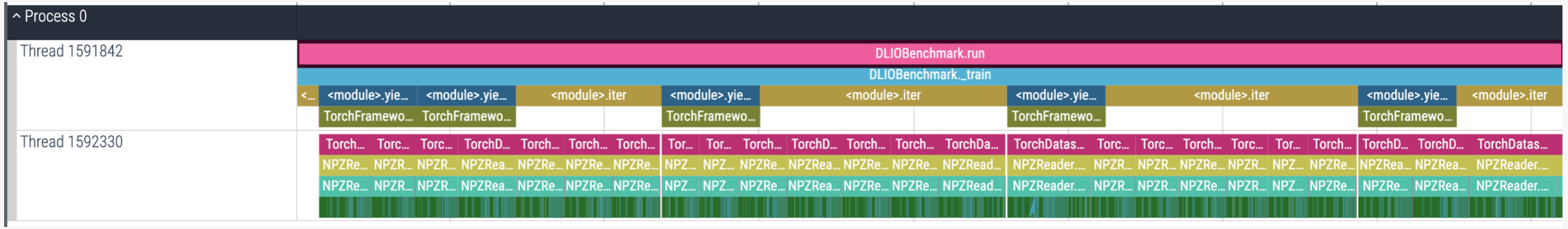


HDF5 format
num_workers = 2
AU = 98.3%

ML Commons

Argonne NATIONAL LABORATORY

3D UNet on A100 with Lustre



Single thread performance on Lustre is much worse than that on SSD

Single NPZ file is loaded in chunks of
4MB with a lot of lseek calls in between

Multiple GPUs per node

Per node bandwidth to support compute

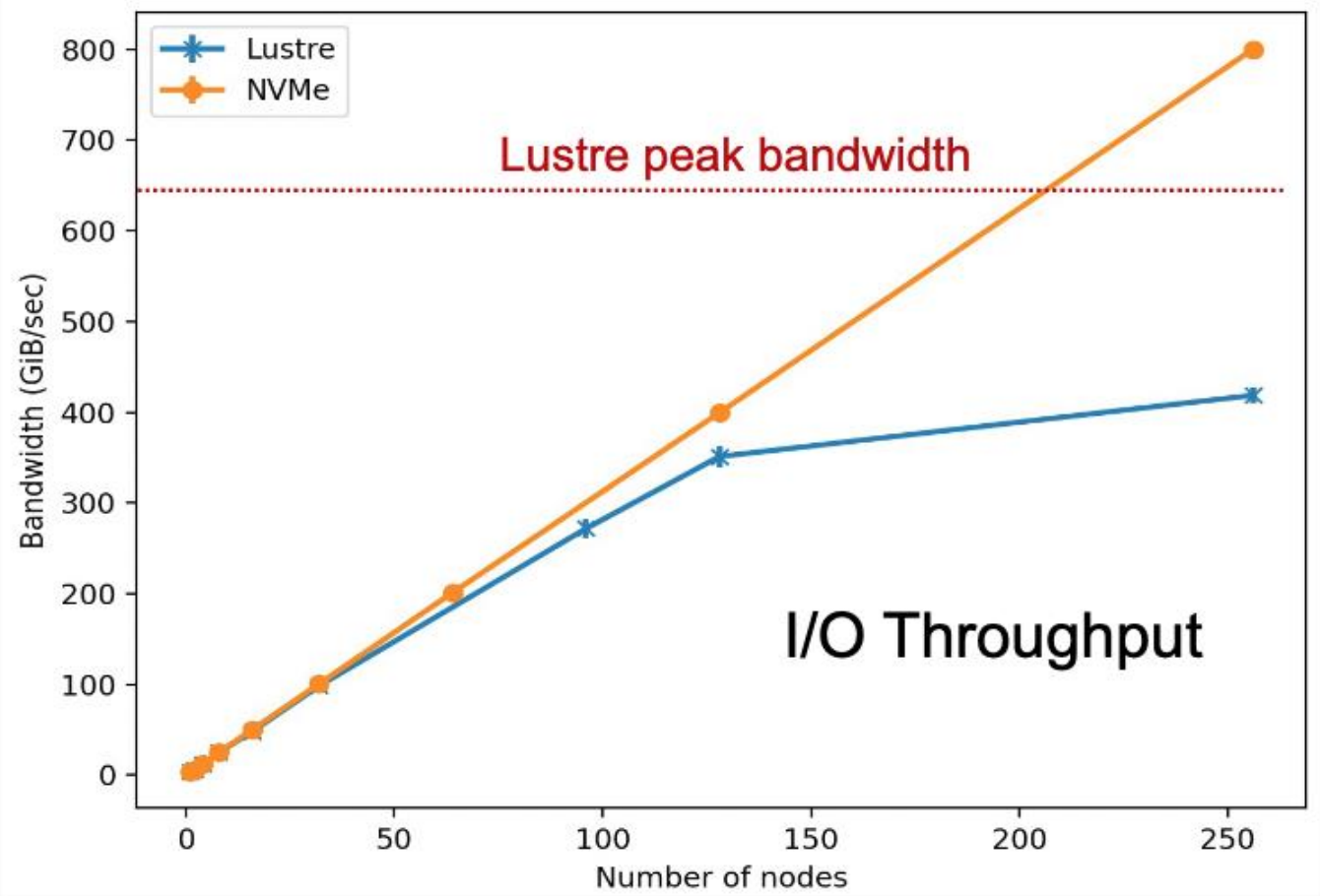
#GPU	A100	H100
1	1.50	2.66
2	3.01	5.92
4	6.01	11.84
8	12.02	23.67
12	18.03	35.51

Aggregate per node (rough peaks; controller/CPU/FS can cap)

Node NVMe Config	PCIe Gen	Drives	Est. Read BW	Est. Write BW
1× NVMe	Gen4 x4	1	**~7 GB/s**	~6 GB/s
2× NVMe	Gen4 x4	2	**~14 GB/s**	~12 GB/s
4× NVMe	Gen4 x4	4	**~28 GB/s**	~24 GB/s
8× NVMe	Gen4 x4	8	**~55 GB/s**	~48 GB/s
1× NVMe	Gen5 x4	1	**~13 GB/s**	~11 GB/s
4× NVMe	Gen5 x4	4	**~50–55 GB/s**	~44–48 GB/s

- Be aware of the overhead from python libraries (numpy)
- Remote storage will be bounded by #NICs on a node
- Need enough CPU cores to drive the data loading

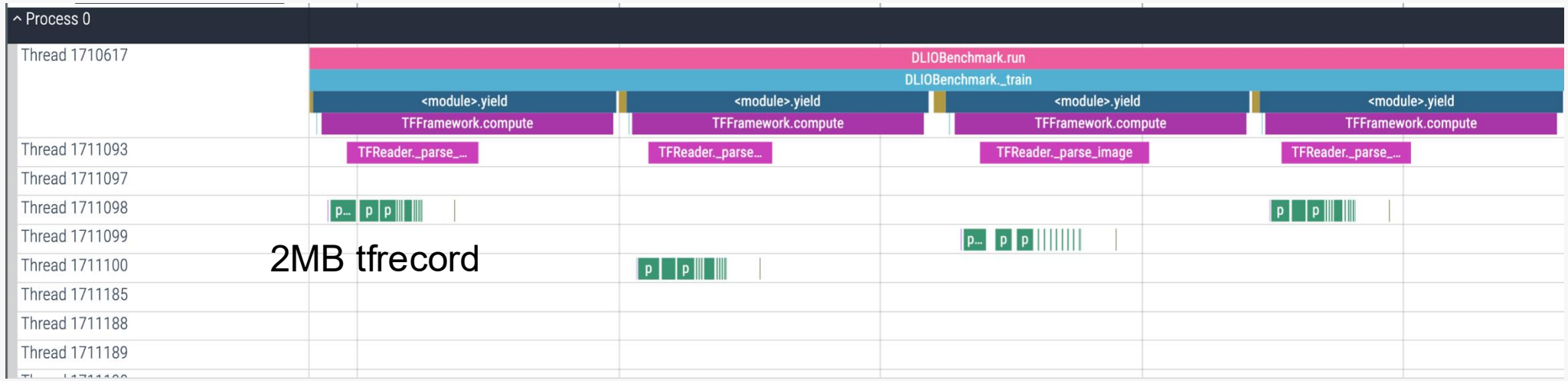
Filesystem peak bandwidth matters in scaling out



UNet3D (V100) weak scaling and I/O throughput at different scale on Polaris w/ Lustre file system and NVMe

CosmoFlow no A100 with SSD

Accelerator	A100	H100
Compute (s)	0.0055	0.0035

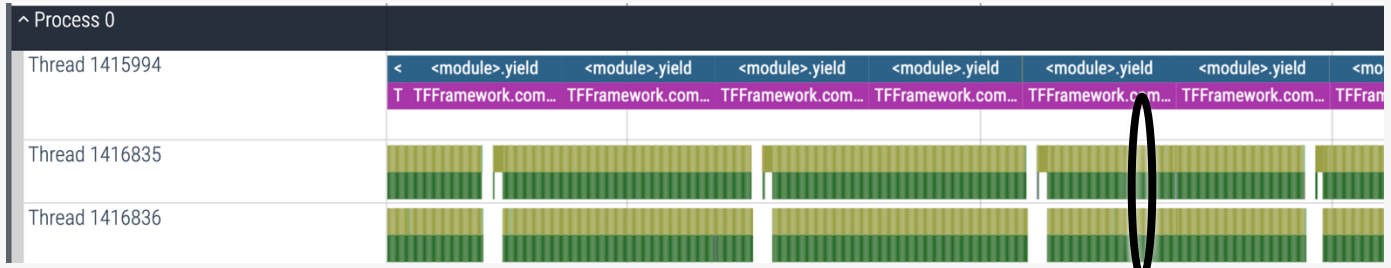


Bandwidth per node to support compute

#GPU	a100	h100
1	0.33	0.53
2	0.67	1.05
4	1.34	2.11
8	2.68	4.21
12	4.02	6.32

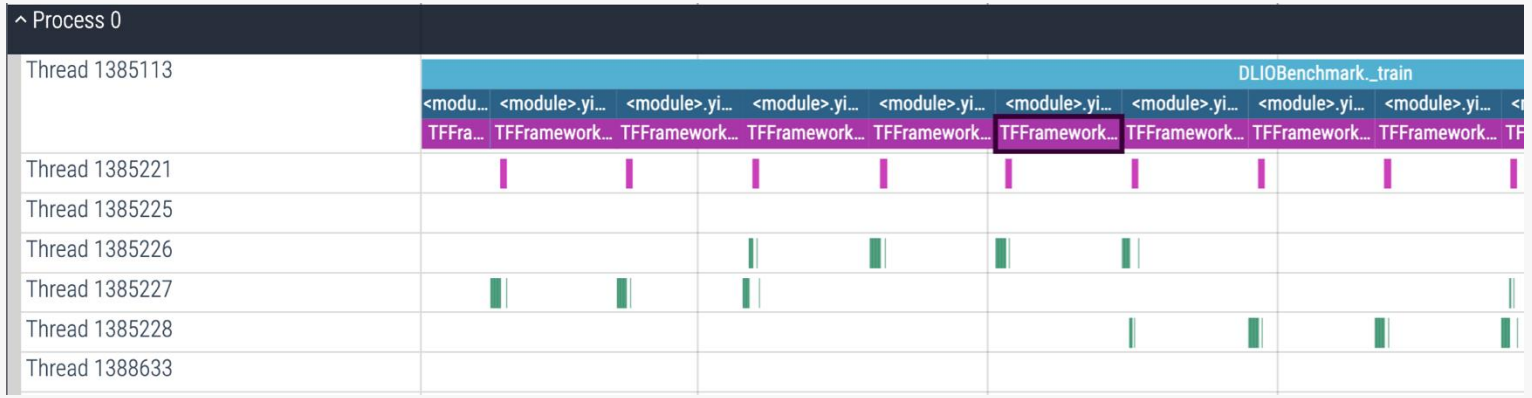
- CosmoFlow is not less I/O intensive then 3D-UNet
- Because compute is so small, small overhead from frameworks and python can cause low AU

ResNet50 on A100 with



PNG format, 1 image per file

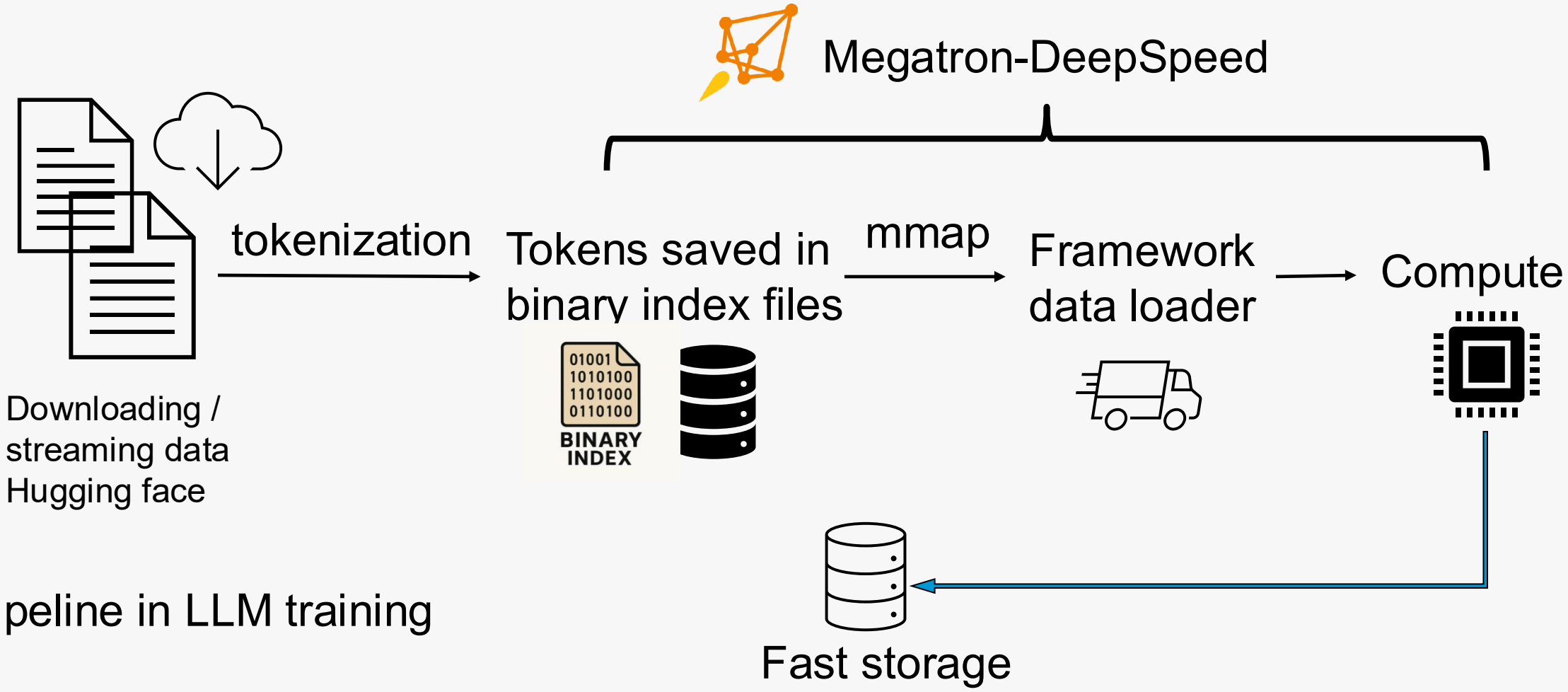
Inefficient loading for PNG files



tfrecord format, 1250 images per file

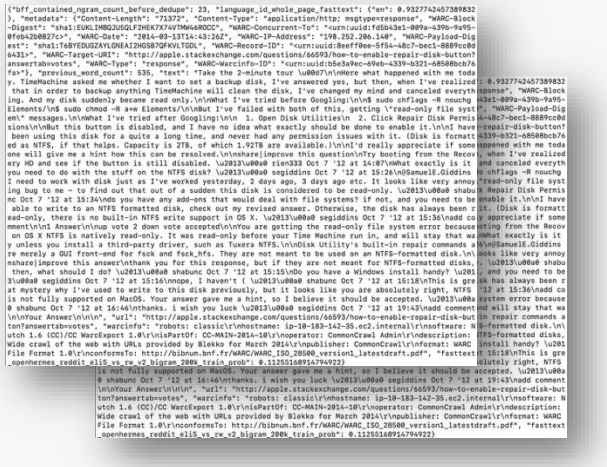
#GPU	a100	h100
1	0.09	0.17
2	0.18	0.34
4	0.35	0.69
8	0.71	1.37
12	1.06	2.06

Large Language Models



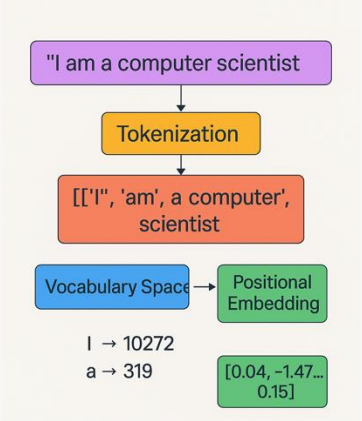
Pipeline in LLM training

Tokenization – compute bound



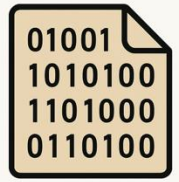
Raw data in json

read



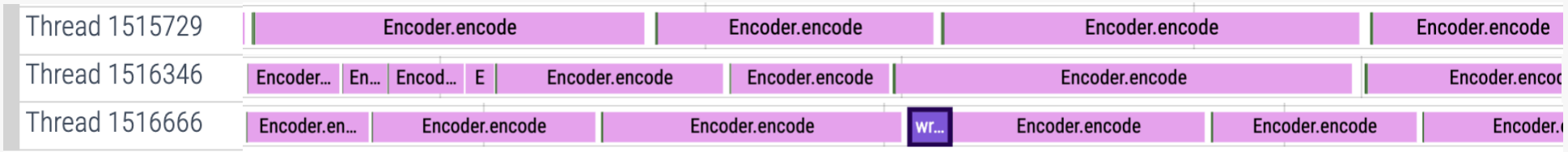
compute

write



*.bin
*.idx

Binary index files



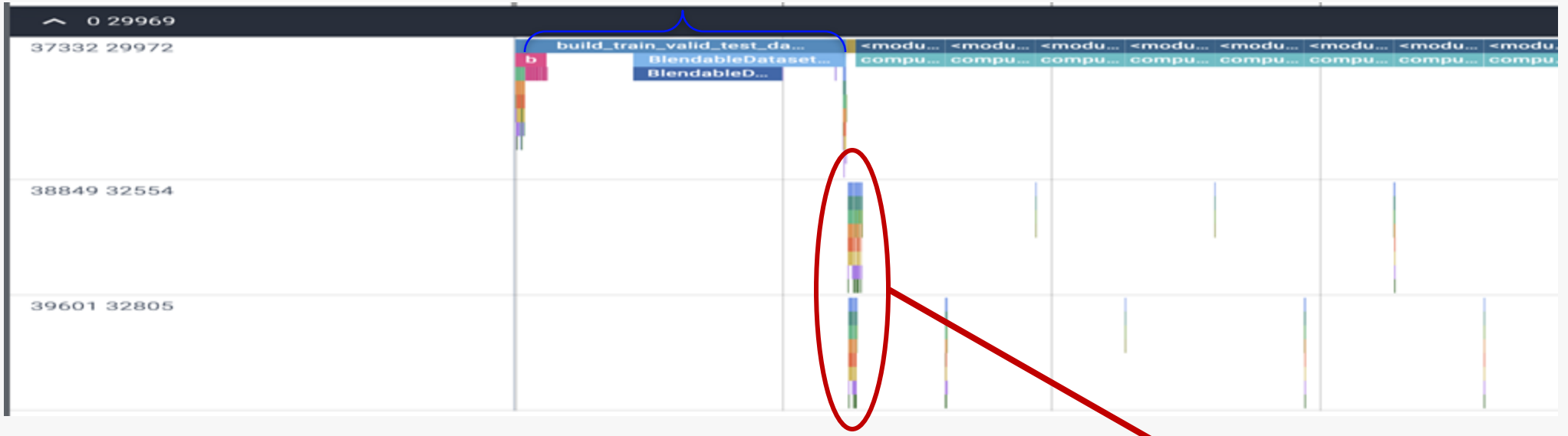
Tracing of tokenization process

Llama2Tokenizer: 2 MB/s pre thread (AMD EPYC 7742)
Read: 8k size; write: 4MB

ML Commons

Argonne NATIONAL LABORATORY

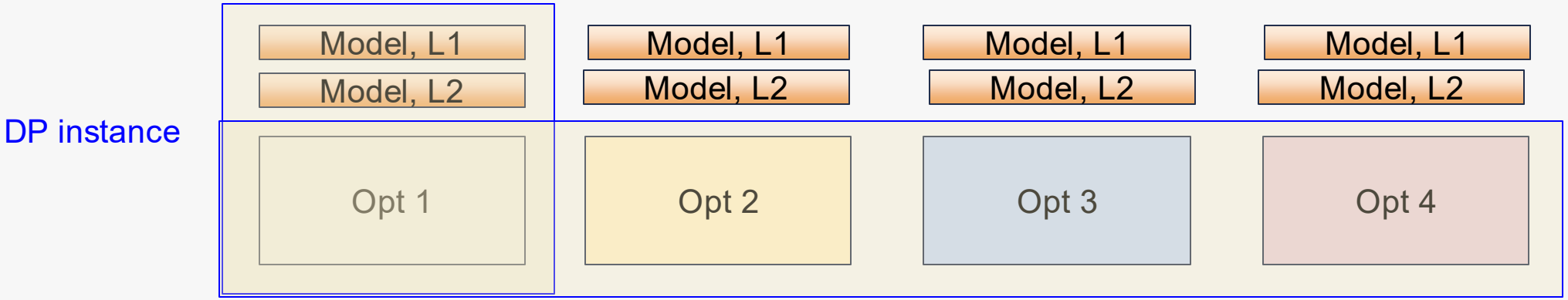
Data loading during training



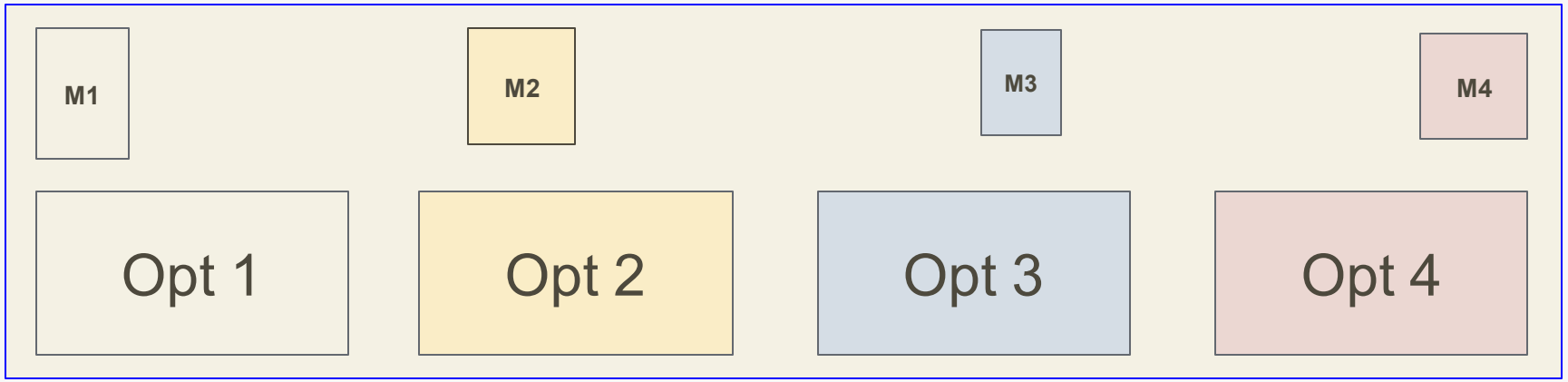
Loading the samples are very fast

Loading a microbatch (70ms): overlap with compute

Checkpointing

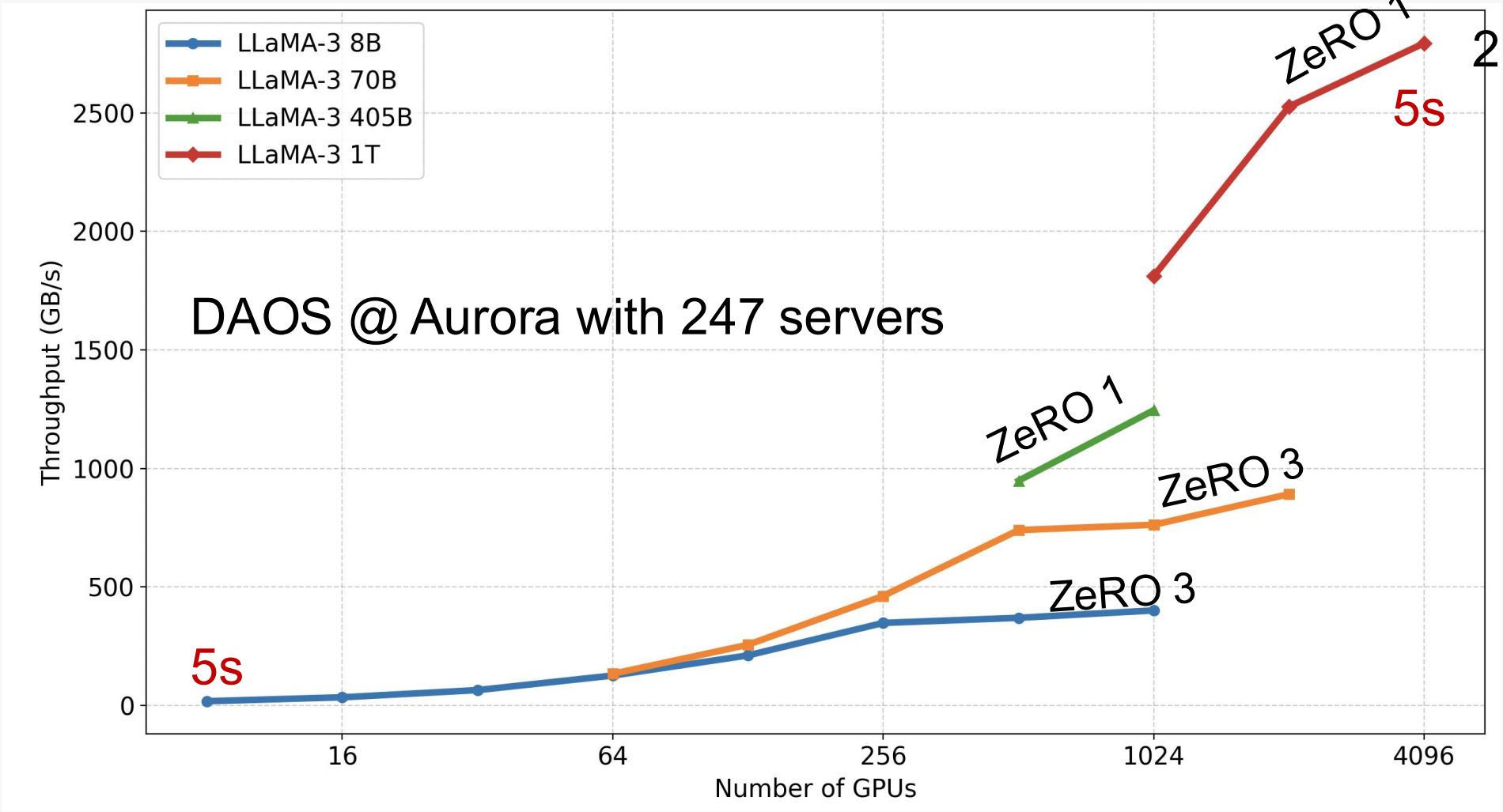


Zero 1 with PP: only optimizer is fully sharded; model are sharded within each DP group



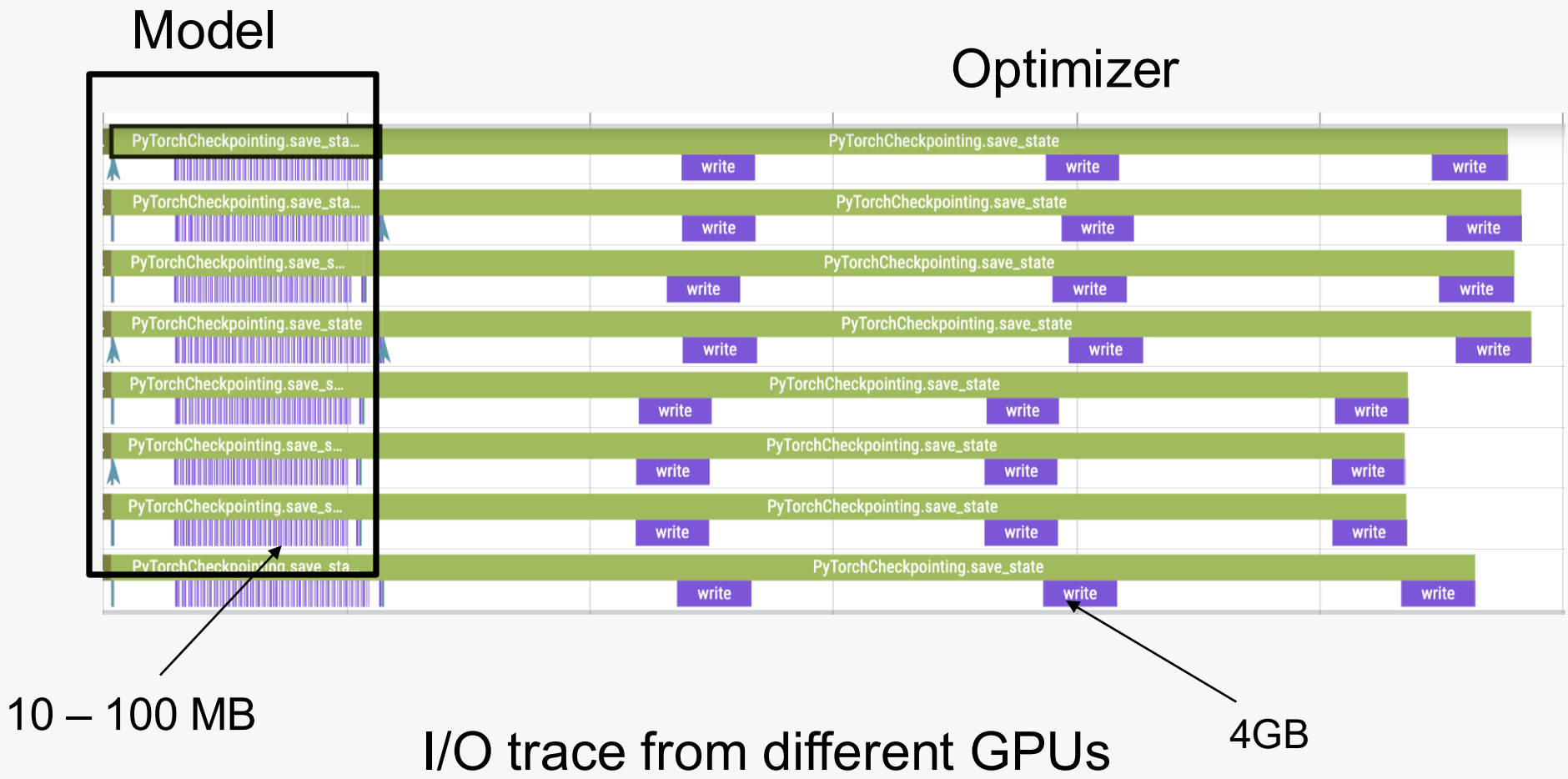
Zero 3, Both model and optimizers are fully sharded

Scaling of checkpointing for different models on DAOS

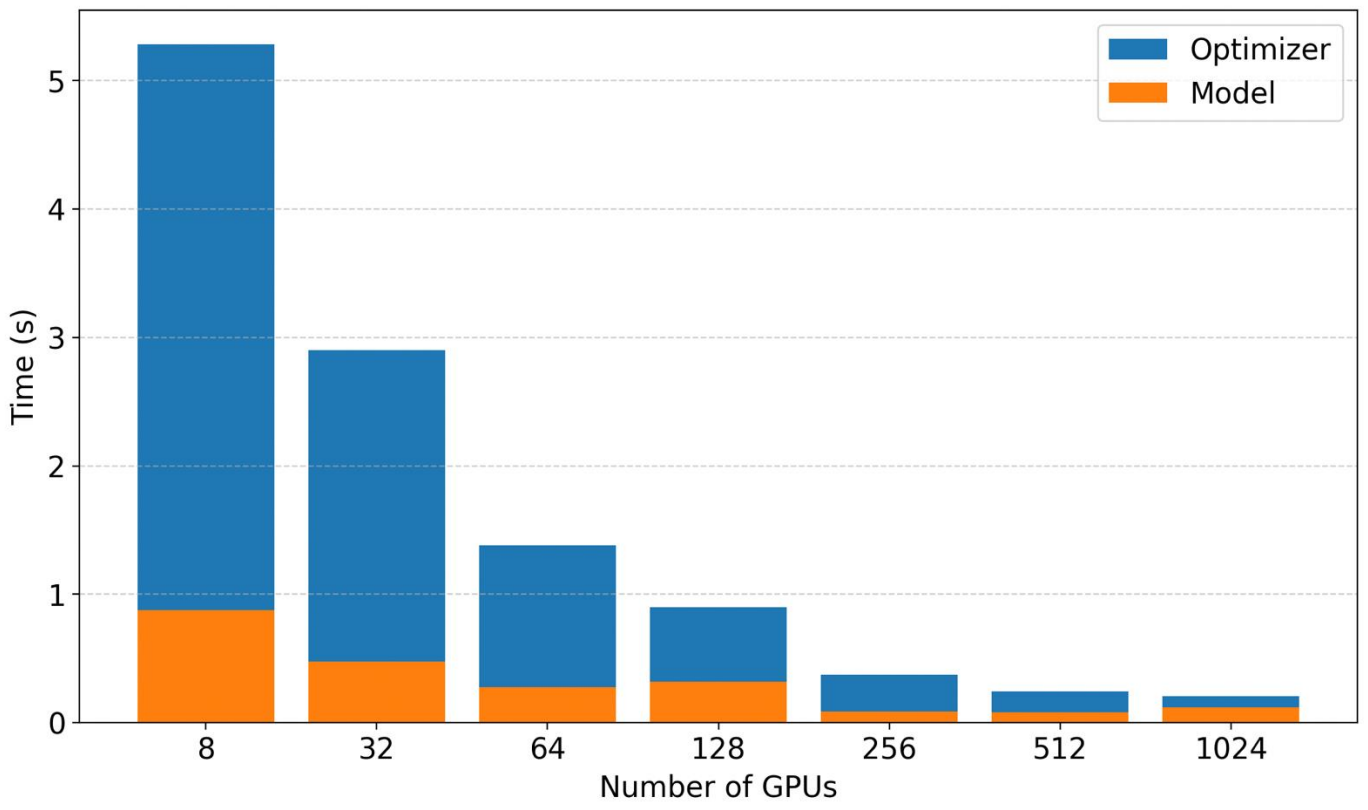


Throughput perfor

Llama3 8B checkpointing I/O pattern (8 GPU, ZeRO3)

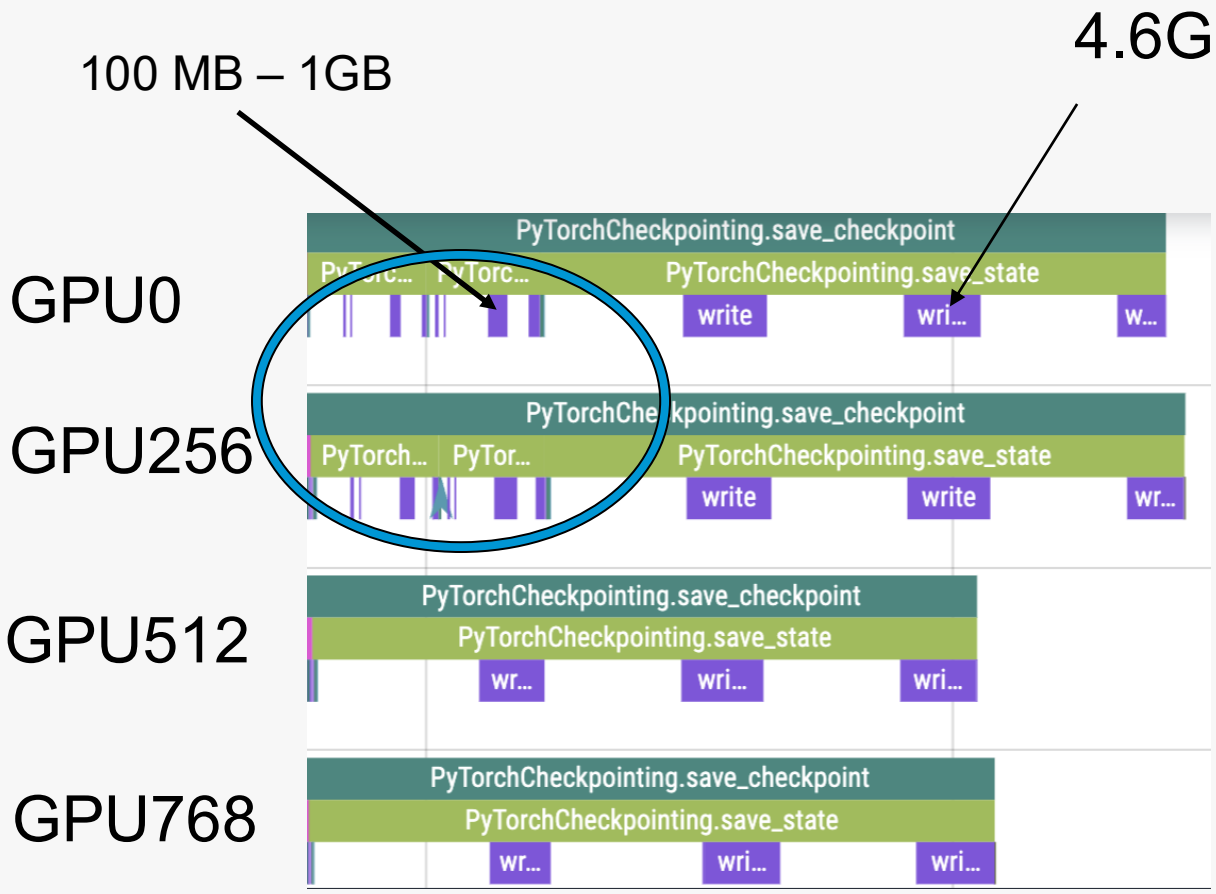


Llama3 8B checkpointing I/O scaling out



Checkpoint overhead at different scales: small I/O in model checkpointing cause poor scaling efficiency at large scale

Llama3 1T I/O pattern (1024 GPUs, ZeRO1, DP=2)



I/O trace from different GPUs

# GPUs	1024	2048	4096
Model	2.4	2.9	3.7
Optimizer	5.9	2.9	1.4
BW (TB/s)	1.81	2.53	2.79

As scale increases, optimizer checkpoint size shrinks while model checkpoint size remains fixed, making model checkpoints the dominant cost."

Future roadmap

- **New Workloads**: Graph Neural Networks (GNNs) and diffusion models introduce irregular graph traversals and very large intermediate states.
- **KV-cache persistence**: Efficient storage and management of key–value caches is essential for low-latency LLM serving.
- **Vector & Graph Data:** Vector databases and knowledge graphs drive demand for fast query, indexing, and retrieval.
- **GPU Initialized data loading:** Shifting preprocessing and staging to GPUs minimizes bottlenecks and sustains high utilization.

Takeaways

- **AI changes the storage game**: the I/O patterns of AI training are very different from traditional HPC, and storage design must adapt.
- **Benchmarks matter**: DLIO and MLPerf Storage give us a common way to measure and understand how storage impacts AI workloads.
- **Tools expose hidden gaps**: profiling with DFTracer shows that bottlenecks often live in the data pipeline, not just the storage hardware.
- **It takes a community**: with contributions from labs, vendors, and cloud providers, MLPerf Storage builds a shared view of what's needed to power AI at scale.

Acknowledgements

Collaborators involved in designing the benchmarks


Hariharan Devarajan
LLNL


Kaushik Velusamy
ANL


Zhong Zheng
UIC


Sergio Servantez
Northwestern U.

MLPerf Storage Working Group


David Kanter


Oana Balmau
McGill U.


Curtis Anderson
Panasas


Johnu George
Nutanix

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.