SNIA DNSF | DATA, NETWORKING, & STORAGE

AI Storage: The Critical Role of Storage in Optimizing AI Training Workloads

Live Webinar

October 30, 2024

10:00 am PT / 1:00 pm ET

Today's Presenters

Jayanthi Ramakalanjiyam
Software Engineering Leader
Celestica

Ugur Kaynar, PhD
Technical Staff, Storage Technologist,
Chief Technology Office at Dell

The SNIA Community

200 Corporations, universities, startups, and individuals	**2,500** Active contributing members	**50,000** Worldwide IT end users and professionals

SNIA DNSF | DATA, NETWORKING & STORAGE

What We Do

Drive the awareness and adoption of a broad set of technologies, including:

- Storage Protocols (Block, File, Object)
- Traditional and software-defined storage
- Disaggregated, virtualized and hyperconverged
- AI, including storage and networking considerations
- Edge implementation opportunities and factors
- Storage and networking security
- Acceleration and offloads
- Programming frameworks
- Sustainability

How We Do It

By delivering:

- Expert webinars and podcasts
- White papers
- Articles in trade journals
- Blogs
- Social Media
- Presentations at industry events

www.snia.org/dnsf

SNIA Legal Notice

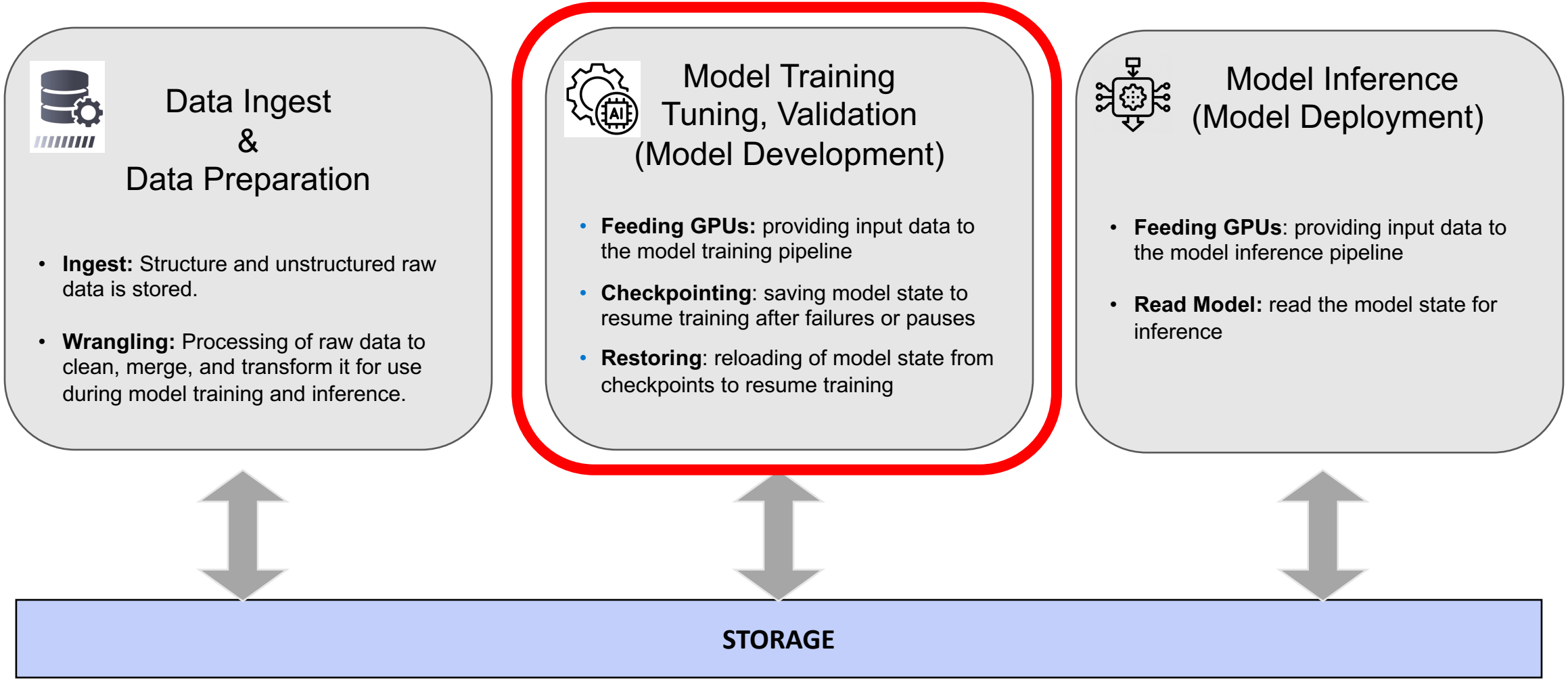
- The material contained in this presentation is copyrighted by SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Today's Agenda

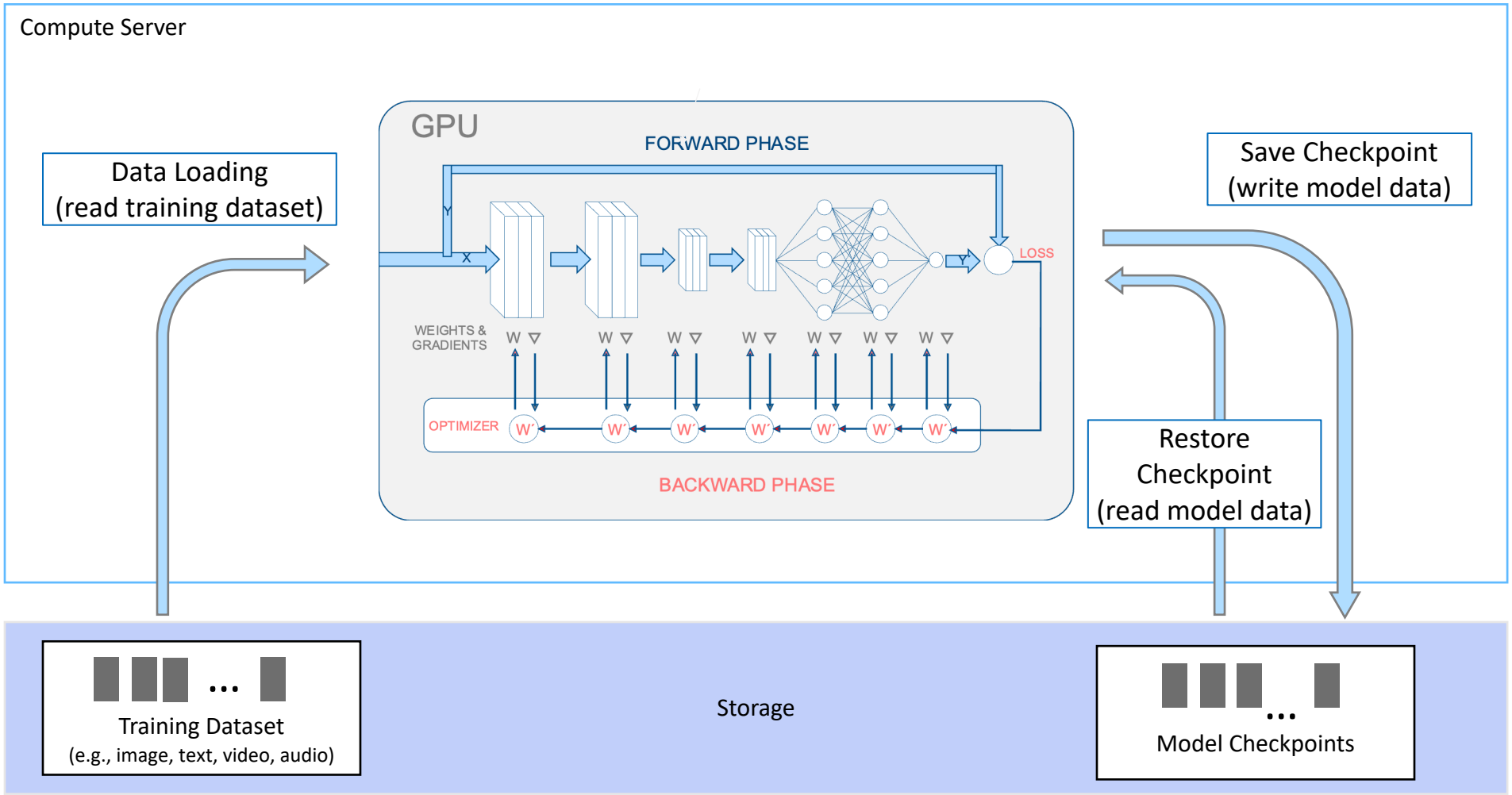
- Overview of AI Model Training
- Data Loading
- Checkpointing
- File and Object Based Storage
  - Storage Connectors

AI workloads interact with storage at every stages of the AI data pipeline.

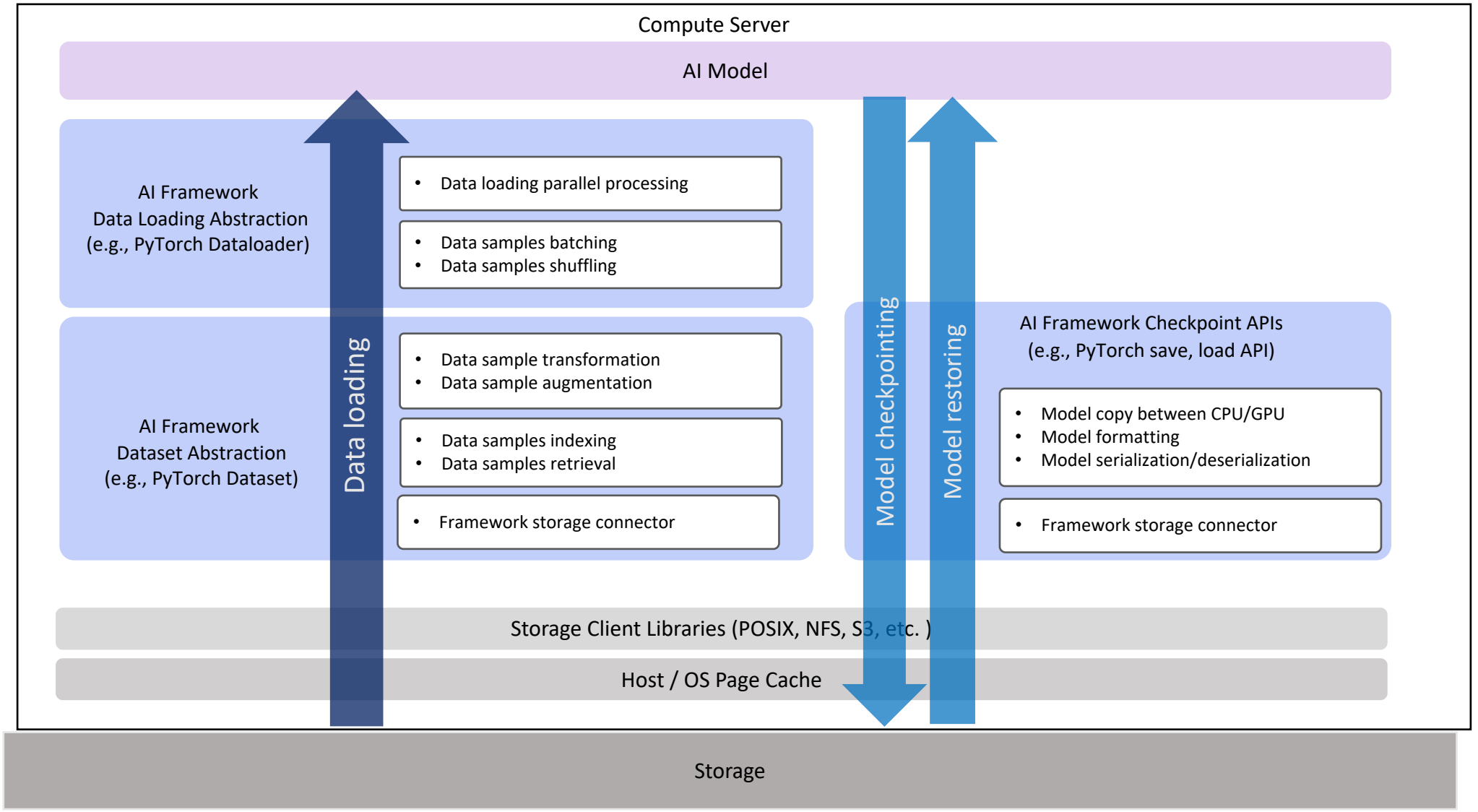


AI Training Workload Storage I/O

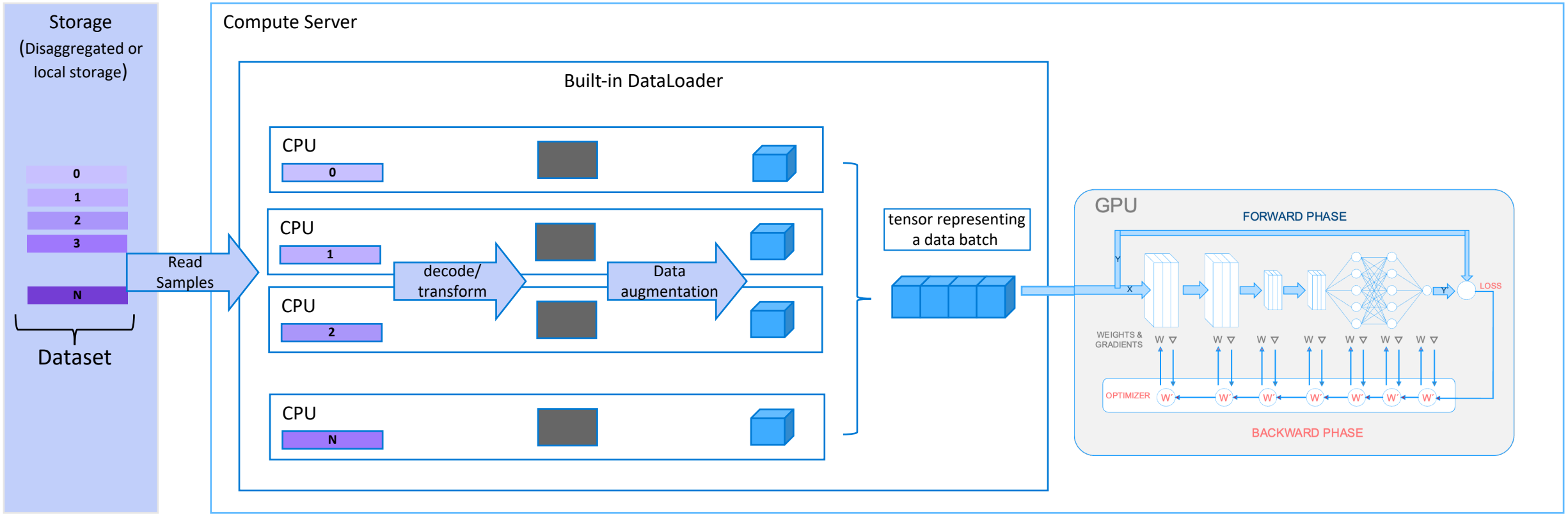
Prevent GPUs from idling on storage IO



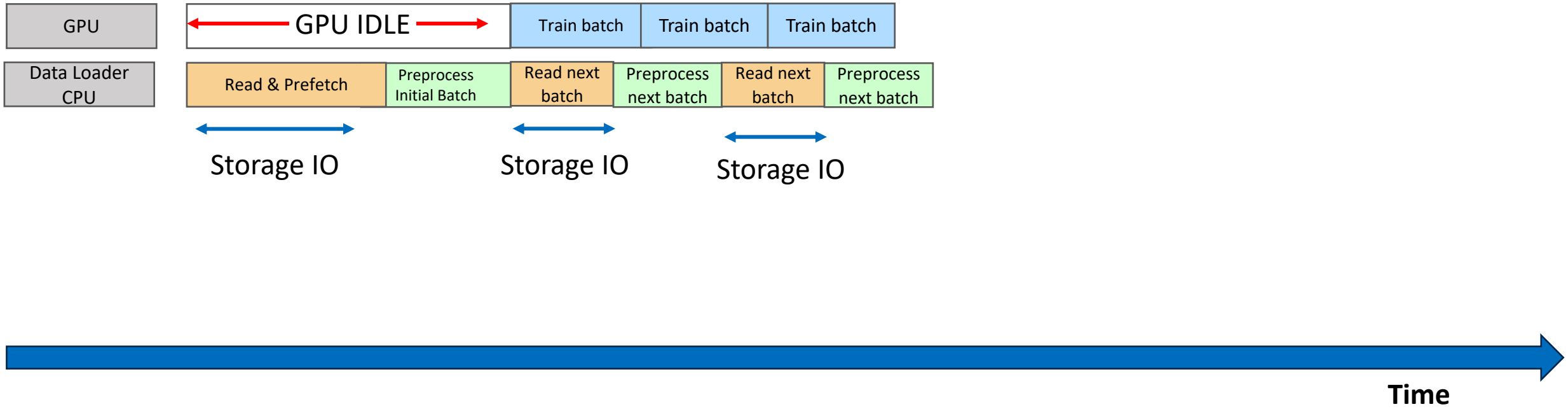
AI Framework Stack and Data Flow



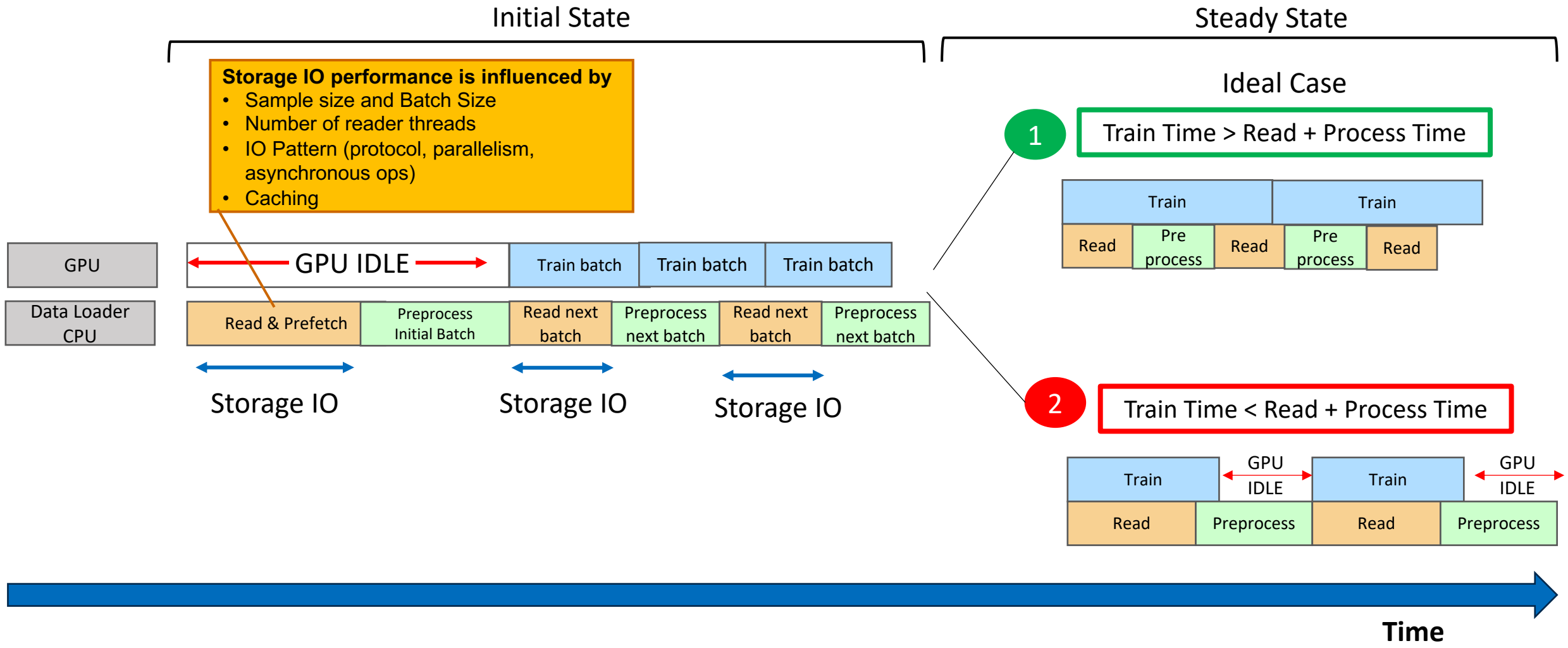
Dataloading involves storage IO and a pipeline of transformations



What causes GPUs to experience starvation during data loading?

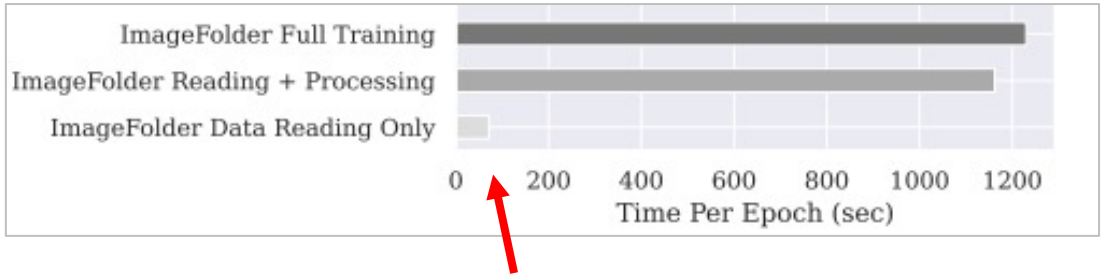


What causes GPUs to experience starvation during data loading?



Preprocessing can be very costly, when dealing with images, video, or audio files.

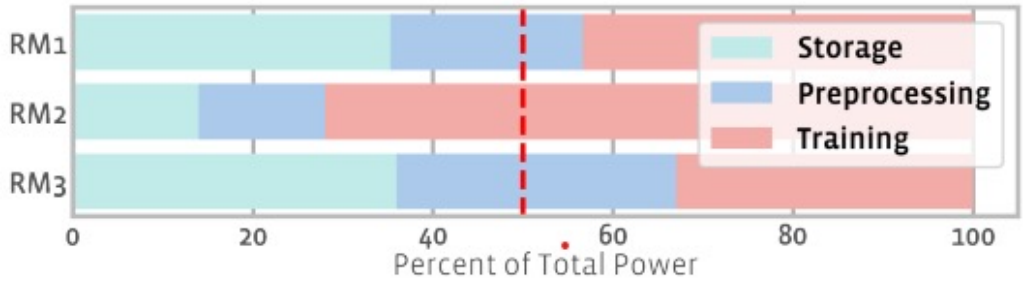
Time taken across stages in ImageNet training[2]



Data pre-processing is the major bottleneck of the training.

- ImageNet dataset stored in JPEG format.
- ImageFolder refers to the default PyTorch data loader used to load ImageNet.

Power consumption in Meta's datacenter for 3 different recommendation model[1]



Meta reports that 56% of GPU cycles were spent stalled waiting for training data, and the trainer's CPUs cannot preprocess data fast enough to serve the GPUs*.

Reference
[1] Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training (arxiv.org)
[2] FFCV: Accelerating Training by Removing Data Bottlenecks

* Storage and online processing pipeline consisting of offline data generation, dataset storage, and online preprocessing services.

Training workloads generate sequential and random read IO to storage systems

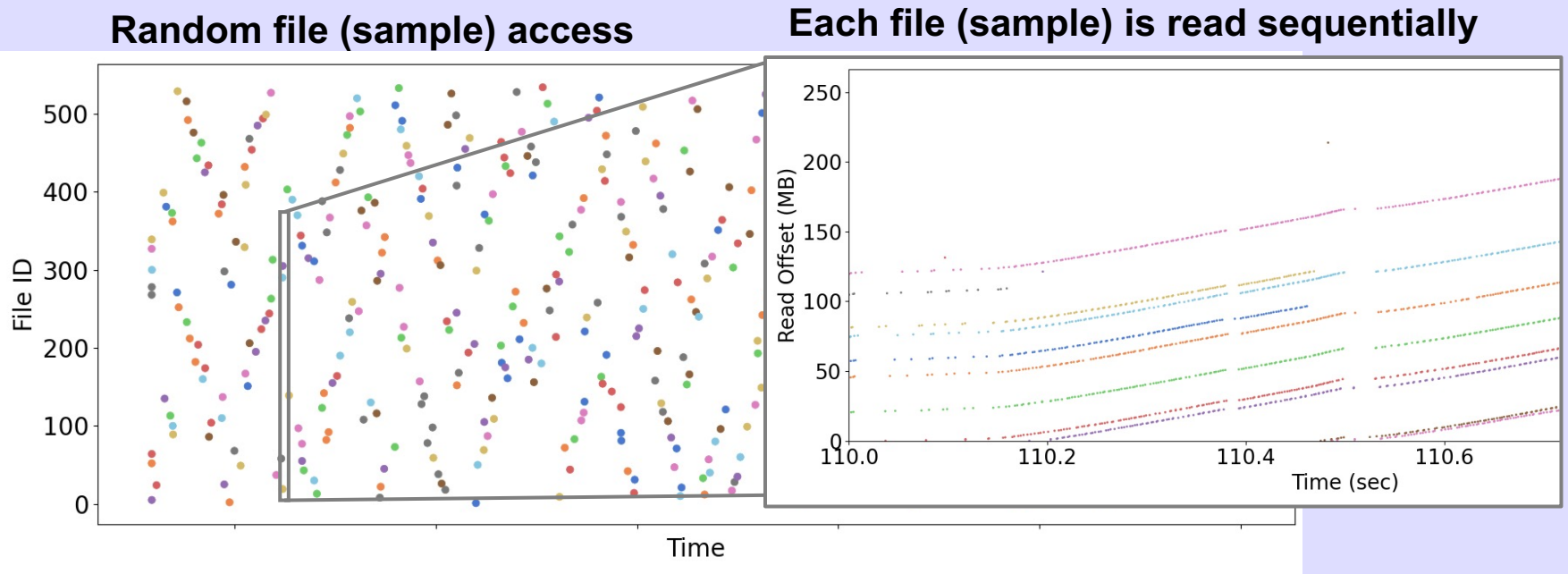
Per Epoch Data Access Pattern: Read Only

- Sequential IO → The entire sample is read
- Random IO → Samples can be retrieved randomly

Multiple Epoch Data Access Pattern: Repeated Access

- During each epoch, the model goes through the entire dataset once.
- Typically, training involves multiple epochs, which result in repeated reads of data samples.
- The data loader shuffles the data to randomize it, ensuring that samples are shuffled at the start of each epoch.

- **3D-Unet** workload from MLPerf Storage Benchmark.
- Each sample stored in individual files.
- File size ~/150MB.



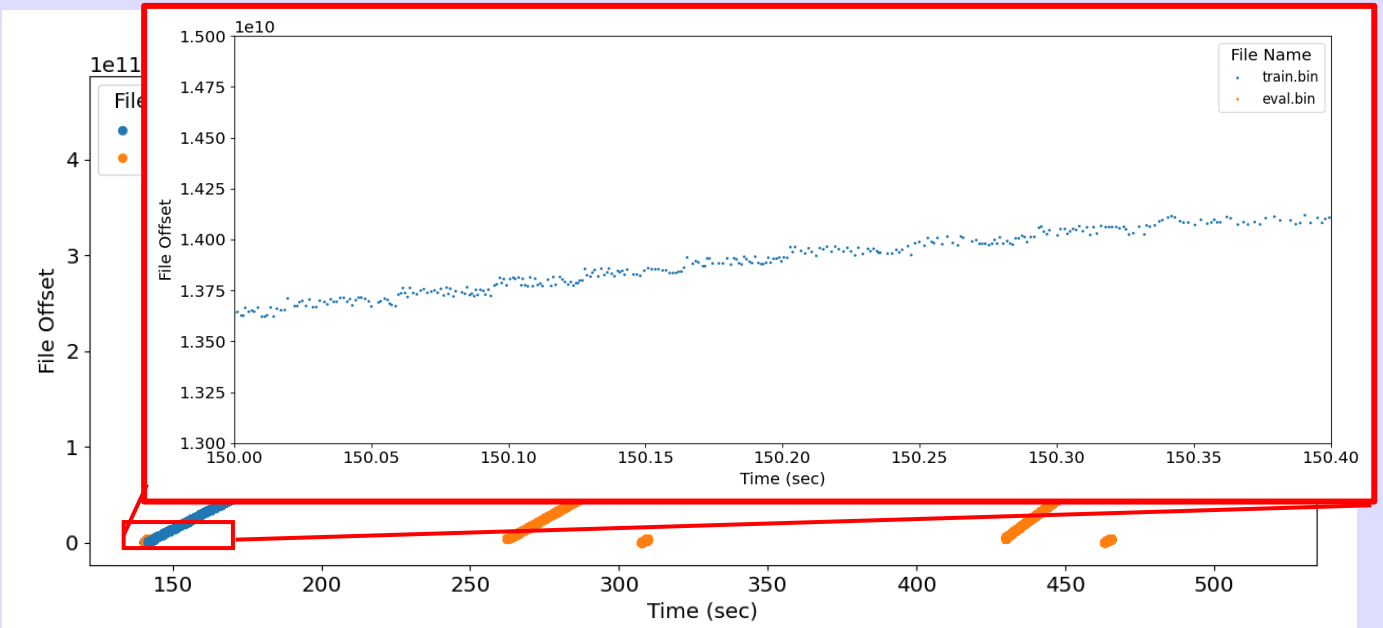
Training workloads generate sequential and random read IO to storage systems

Per Epoch Data Access Pattern: Read Only

- Sequential IO → The entire sample is read
- Random IO → Samples can be retrieved randomly

# of GPU	8x A100
batch size	16 per GPU
Input File	3.8 TB train dataset
Benchmark	MLPerf Training DLRMv2
Protocol	NFS

Workload access data randomly using a sliding window.



Model checkpointing is a periodic process to save current model state

- Checkpoint contains model weights (learned parameters), optimizer state and other states.

- Model checkpointing is done for various reasons:
  - Fault tolerance
  - Model debugging
  - Model evaluation

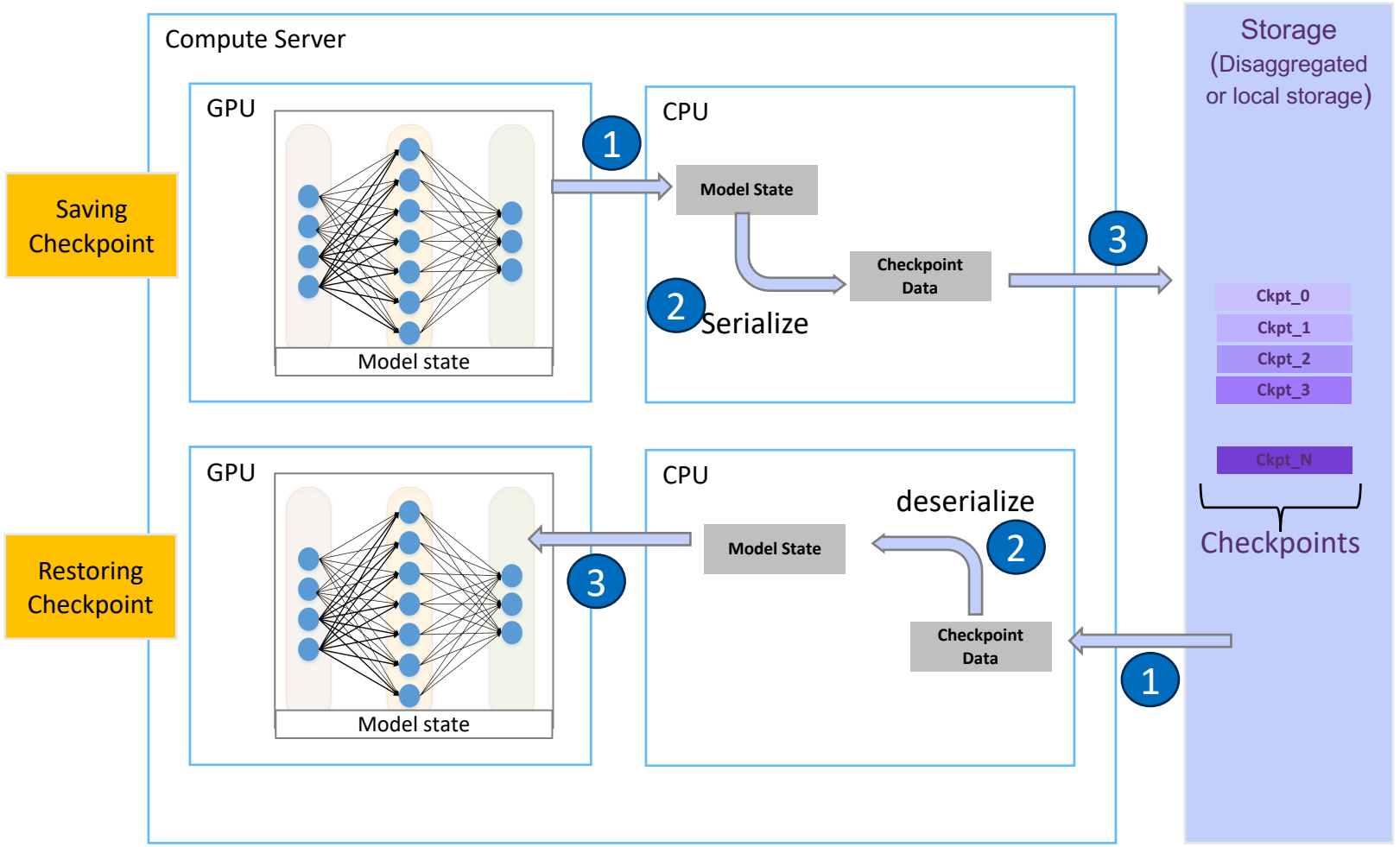
- Generally, checkpoints are retained for the duration of the training process and sometimes longer.
- Model can be restored to any previous versions, depending on failure reason, not just the most recent checkpoint.

The size of a checkpoint is based on the model size and is not influenced by the data size, GPU memory size or the number of GPUs.

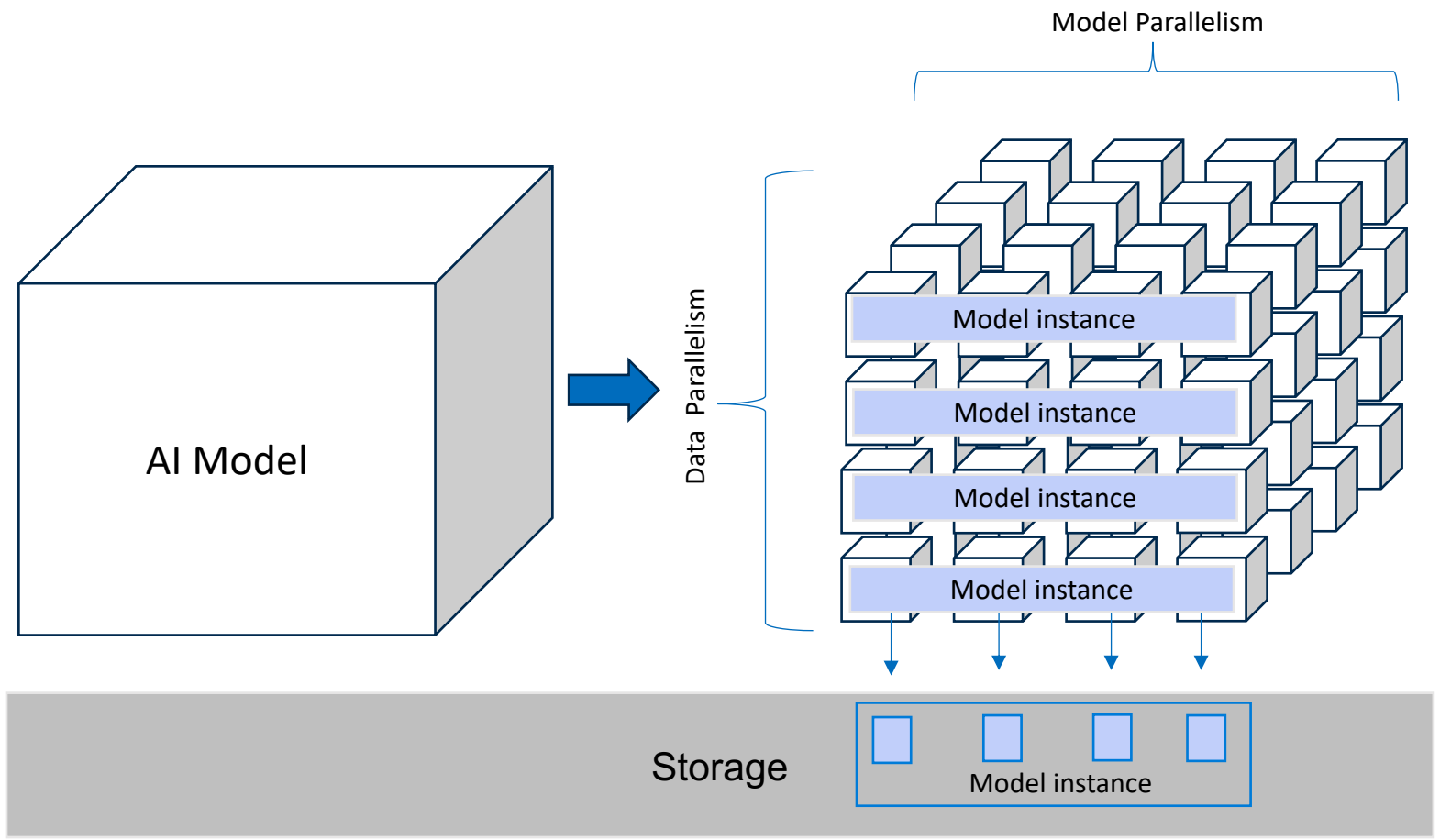
	Model Parameters (B)	Total Checkpoint Size (TB)
GPT3	175	2.4
Megatron-Turing NLG (MT-NLG)	530	7.4

Assumptions:
2 bytes per model parameter (BF16)
12 bytes per model parameter for optimizer and other state

The checkpointing process is expensive because training pauses during checkpointing.



Checkpoints may be saved as one or more files; depends on model parallelism and implementation



When using data parallelism;

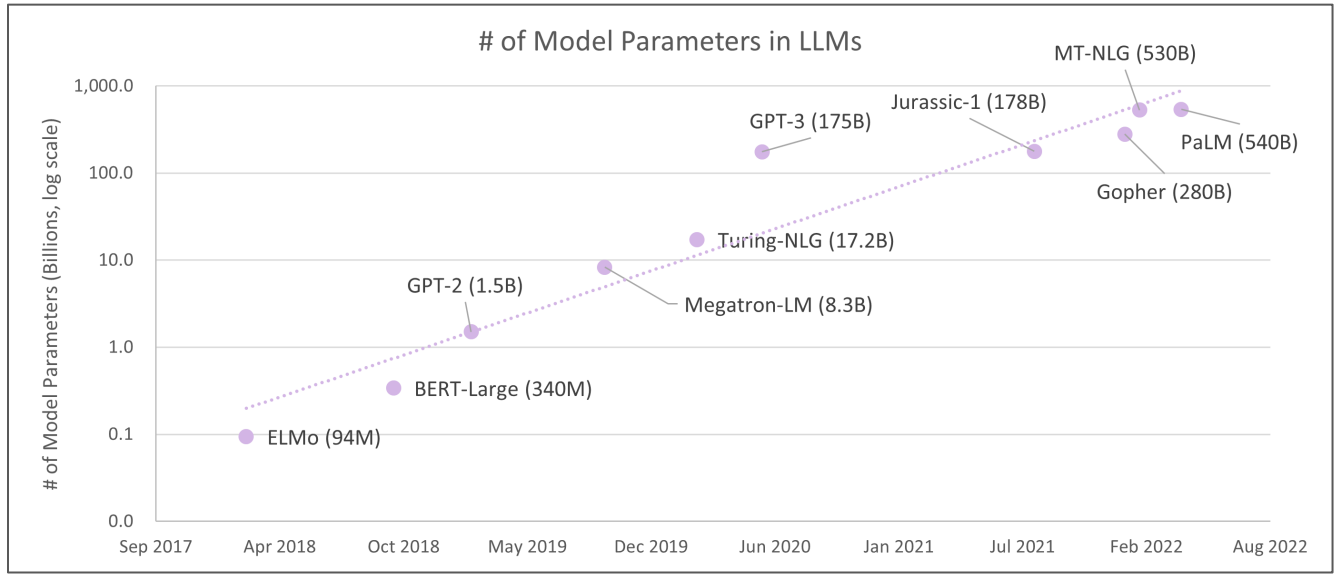
- Every GPU maintains the identical model state including model parameters and optimizer state.
- Single copy of model state needed to be written

Model Parallelism:

- Each GPU may write its portion of the model's parameters to the checkpoint.

Each checkpoint file is written sequentially to an independent file by a single thread.

Model sizes are increasing, and larger models result in larger checkpoint sizes



Large models → many GPUs → higher probability of failures

In Meta's datacenter, it was reported that checkpointing can slow down training by up to 43%[1].

Checkpointing related overheads in full recovery can consume an average of 12% of total training times[1].

- Alibaba Group reports the failure rates for LLM training tasks can skyrocket to 43.4%[2]

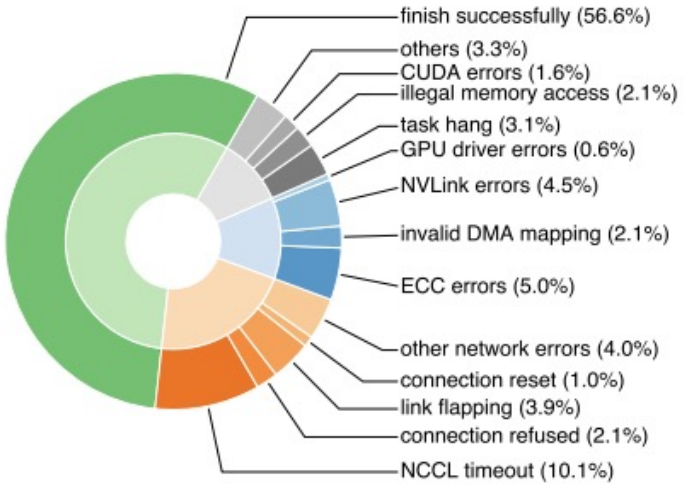


Figure 1: Distribution of task termination statistics.

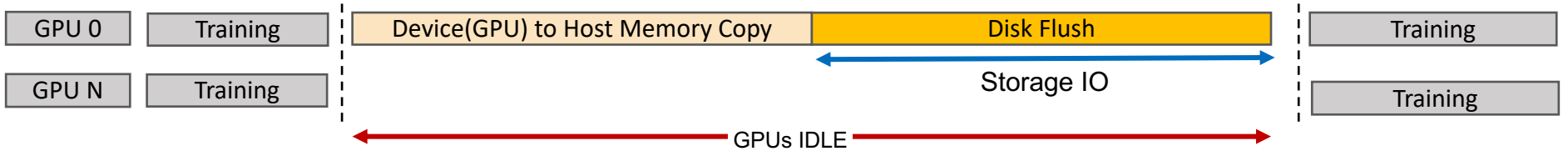
Literature indicates that hardware failures are very common:

- LLaMA3 pre-training also reports that 78% of the failures are hardware issues[3].

Reference
[1] Check-N-Run: a Checkpointing System for Training Deep Learning
[2] Unicron: Economizing Self-Healing LLM Training at Scale
[3] The Llama 3 Herd of Model

When do GPUs stall during checkpointing?

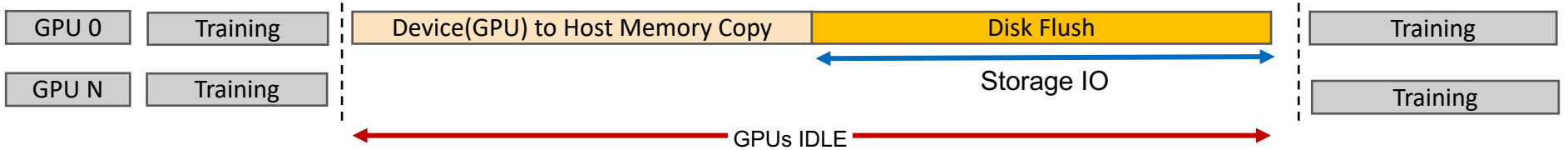
Persistent Checkpointing
`torch.save()`



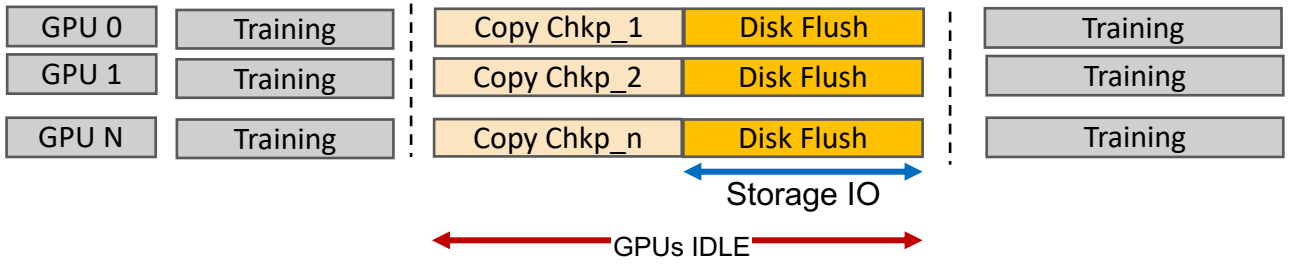
Reference
[1] pytorch/torchsnapshot: A performant, memory-efficient checkpointing library for PyTorch applications, designed with large, complex distributed workloads in mind.
[2] Gemini: Fast failure recovery in distributed training with in-memory checkpoints
[3] Check-N-Run: a Checkpointing System for Training Deep Learning Recommendation Models

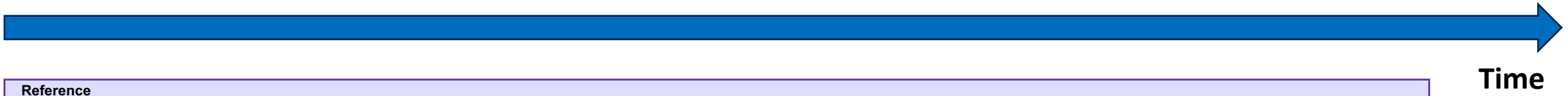
When do GPUs stall during checkpointing?

Persistent Checkpointing
`torch.save()`



Optimizations 1
Parallel Checkpoint Writes
(`torchsnapshot()`[1]
`Pytorch DCP`

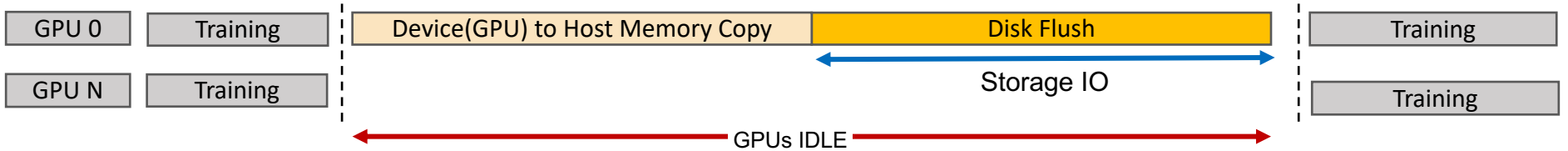




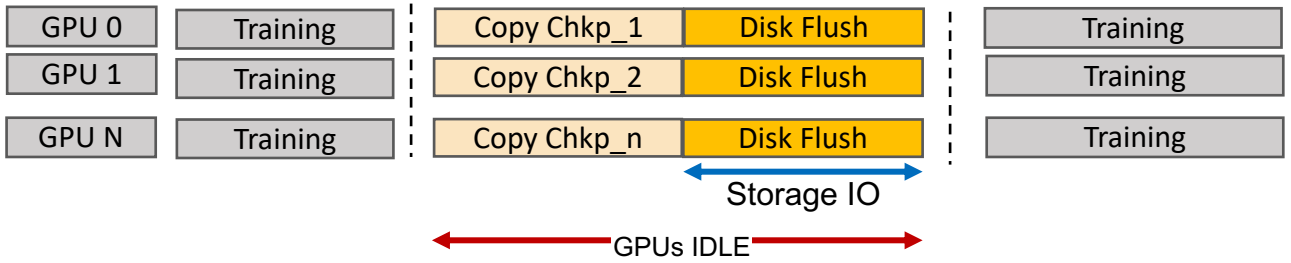
Reference
[1] pytorch/torchsnapshot: A performant, memory-efficient checkpointing library for PyTorch applications, designed with large, complex distributed workloads in mind.
[2] Gemini: Fast failure recovery in distributed training with in-memory checkpoints
[3] Check-N-Run: a Checkpointing System for Training Deep Learning Recommendation Models

When do GPUs stall during checkpointing?

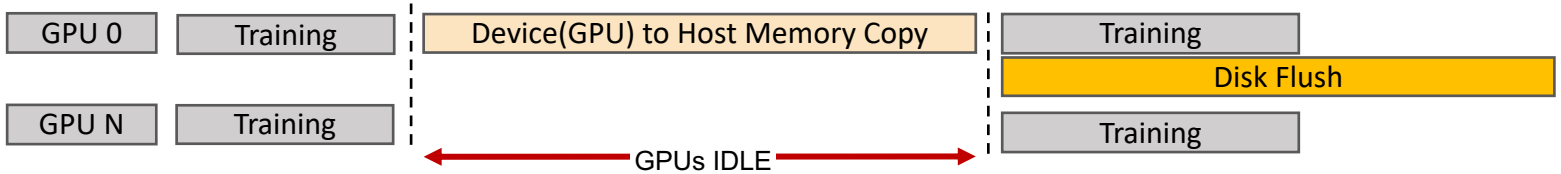
Persistent Checkpointing
`torch.save()`



Optimizations 1
Parallel Checkpoint Writes
`(torchsnapshot()`[1]
`Pytorch DCP`



Optimizations 2
In Memory Checkpointing
`Gemini`[2], `Meta`[3],
`torchsnapshot()`
`Pytorch DCP`



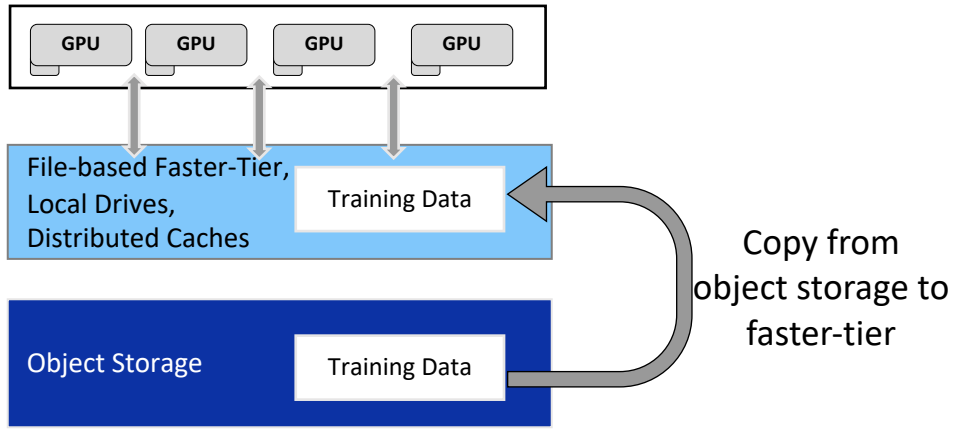
Reference

[1] pytorch/torchsnapshot: A performant, memory-efficient checkpointing library for PyTorch applications, designed with large, complex distributed workloads in mind.
[2] Gemini: Fast failure recovery in distributed training with in-memory checkpoints
[3] Check-N-Run: a Checkpointing System for Training Deep Learning Recommendation Models

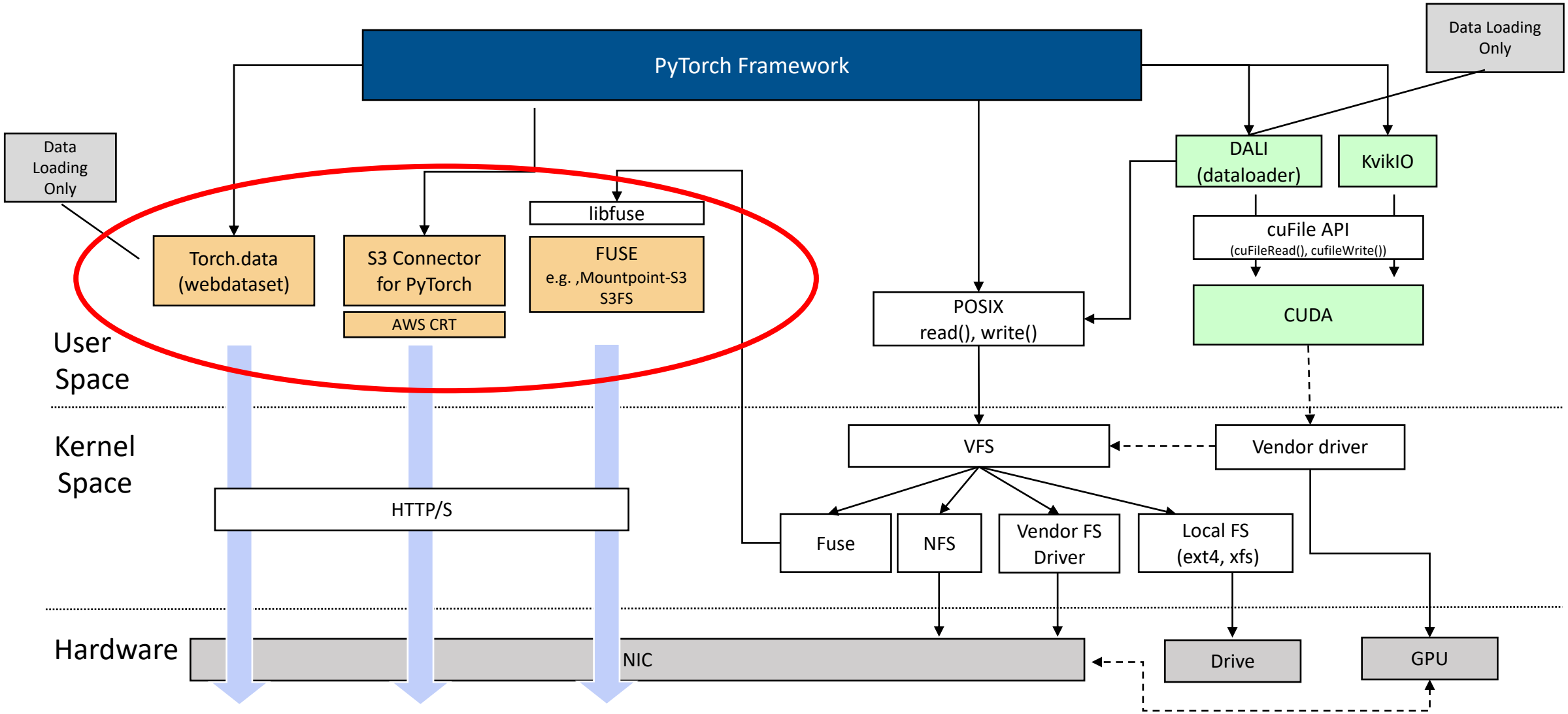
Majority of the AI training workloads today uses file-based storage solutions

- Today, most existing AI frameworks supports both file and object storage for data access for **data loading,** and **checkpointing.**
- File Storage:
  - Typically, file-based storage systems are used for AI model trainings workloads to store and access training data for model training to ensure expensive GPUs are utilized.
  - PyTorch and other AI frameworks designed for file-like access and expect to work on files.

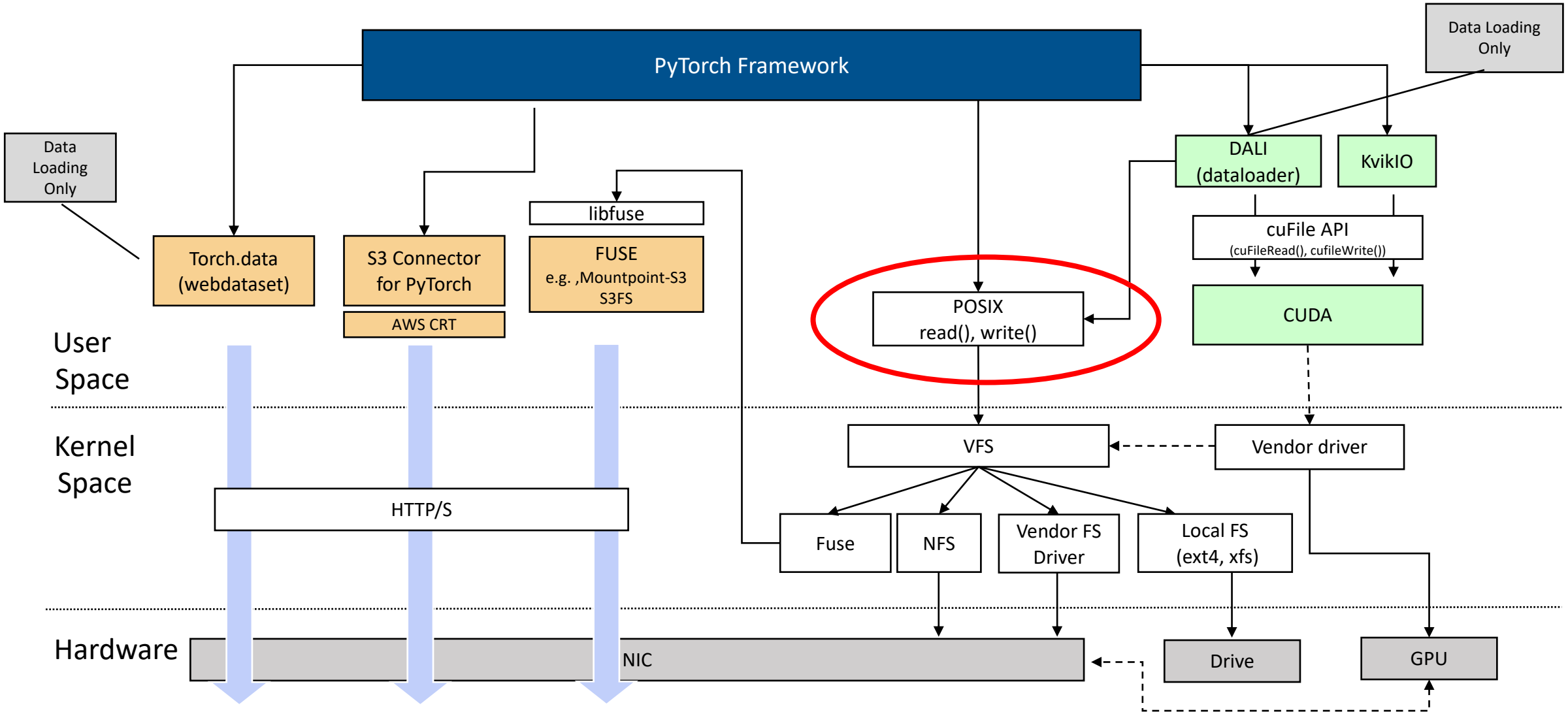
- Object Storage
  - For AI training workloads, the high and unpredictable latencies of object storage prevent it from being directly consumed by AI Frameworks.
  - The S3 APIs used during AI training workloads are quite limited.



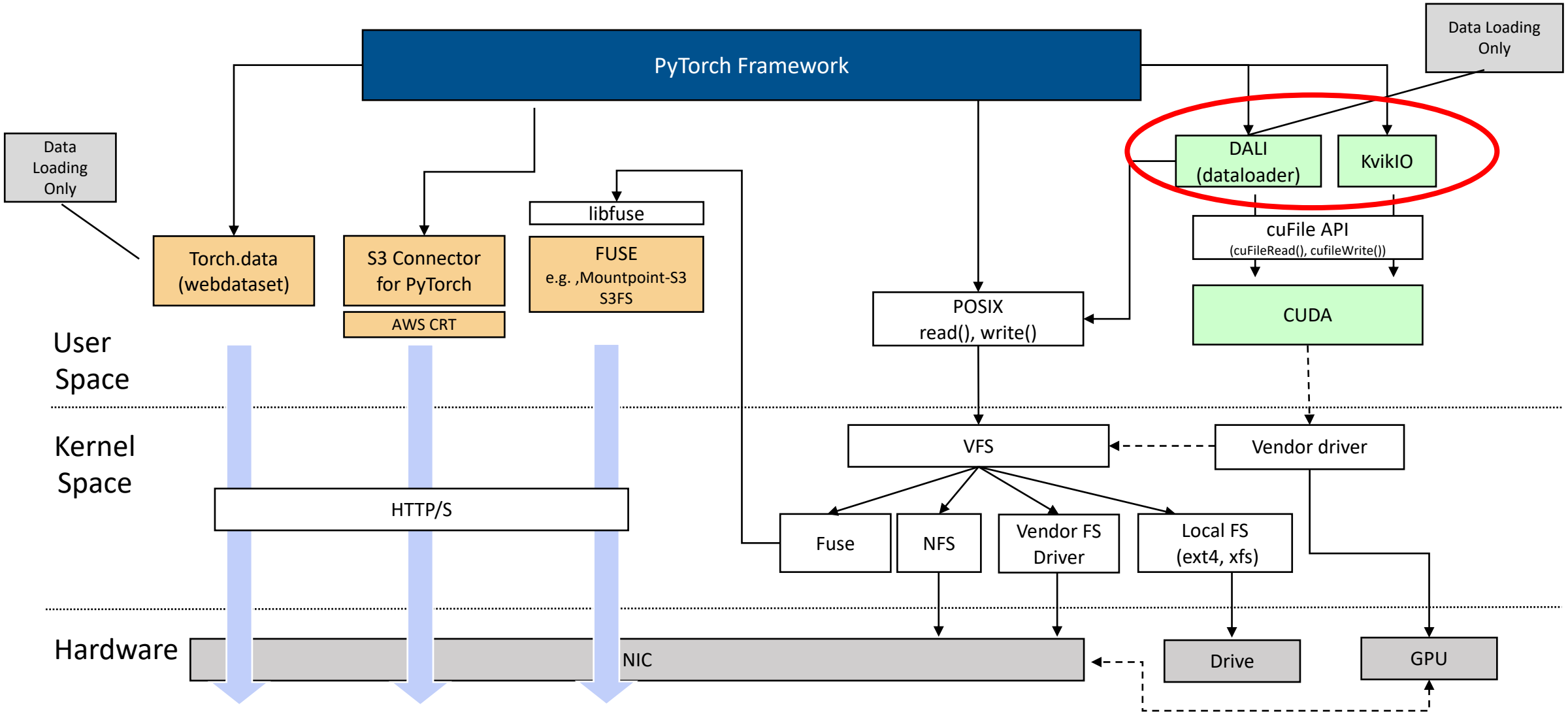
A Storage connector for AI is a specialized tool or library that enables AI frameworks to access storage systems for reading and writing the data



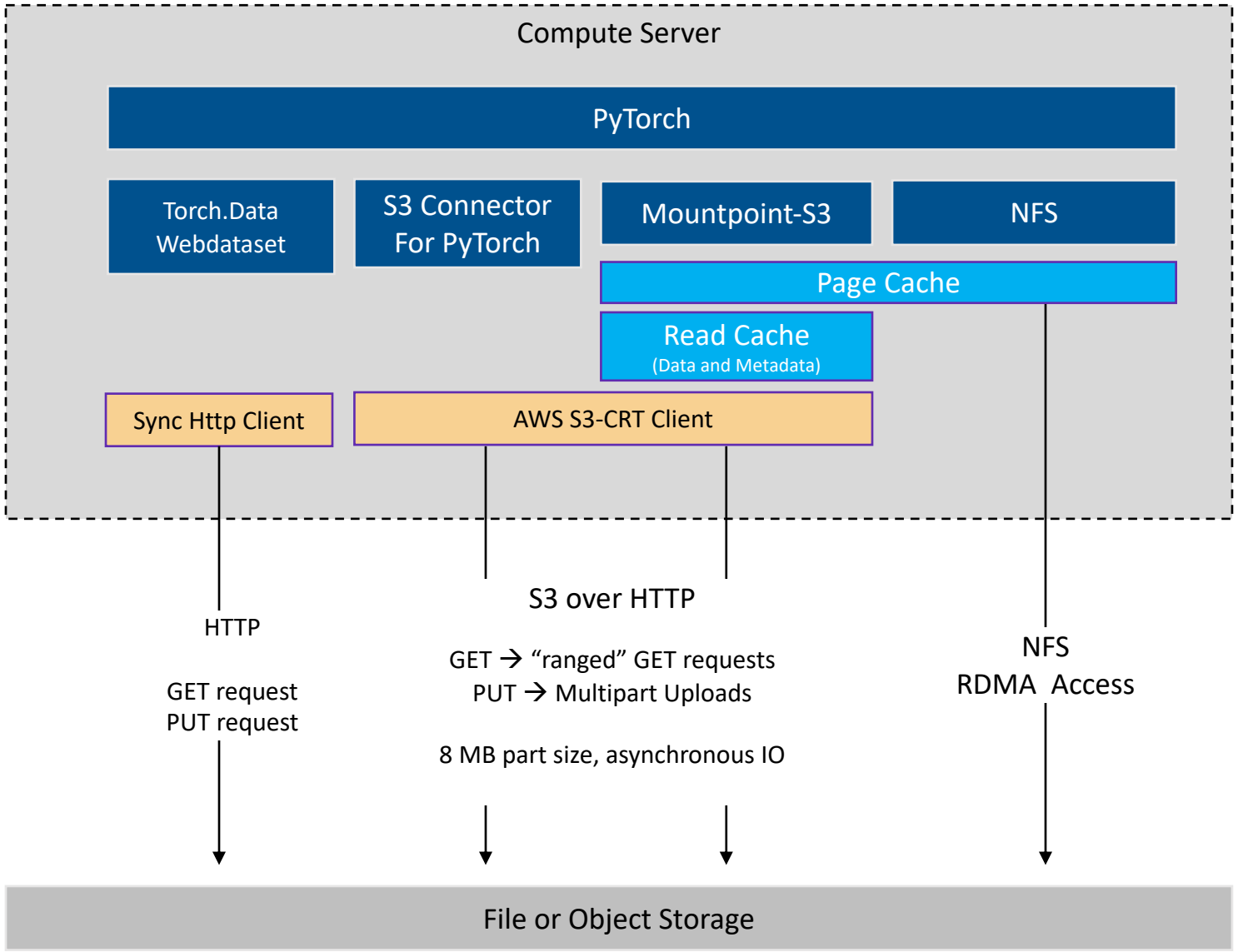
A Storage connector for AI is a specialized tool or library that enables AI frameworks to access storage systems for reading and writing the data



A Storage connector for AI is a specialized tool or library that enables AI frameworks to access storage systems for reading and writing the data



High-level comparison of different storage connectors



- Cache:
  - File-based storage benefits from the OS page-cache.
  - S3 connectors need to implement additional cache to speed up repeated access.
- IO Access Pattern:
  - AWS CRT based S3 Client solutions leveraging asynchronous and parallel I/O
- Protocol:
  - Object protocols have higher latency.
    - IAM authentication and authorizations are very expensive.
  - File-based storage solutions benefit from RDMA technology.

Summary

- Data Loading:
  - Data loading phase consists of storage IO and data transformations
  - IO Access Patterns depend on the model and dataset
  - The storage system must provide high throughput and low latency to ensure that data is fed to the GPUs as quickly as possible
- Checkpointing:
  - Large models need high read and write bandwidth to save and restore checkpoints efficiently.
  - Checkpoint files can be saved as one or more files, and each checkpoint file is written by a single writer.
  - Write accumulated checkpoint storage can be significant for large models and long runs.
- File and Object Storage:
  - AI frameworks expect file-like interfaces to access the storage.
  - However, in recent years, there has been a noticeable increase in support for accessing object storage solutions.

Q&A

After this Webinar

- Please rate this webinar and provide us with your feedback
- This webinar and a copy of the slides are available at the SNIA Educational Library https://www.snia.org/educational-library
- A Q&A from this webinar, including answers to questions we couldn't get to today, will be posted on our blog at https://sniansfblog.org/
- Follow us on X @SNIADNSF

Thank You

