Characterizing Machine Learning I/O with MLPerf Storage

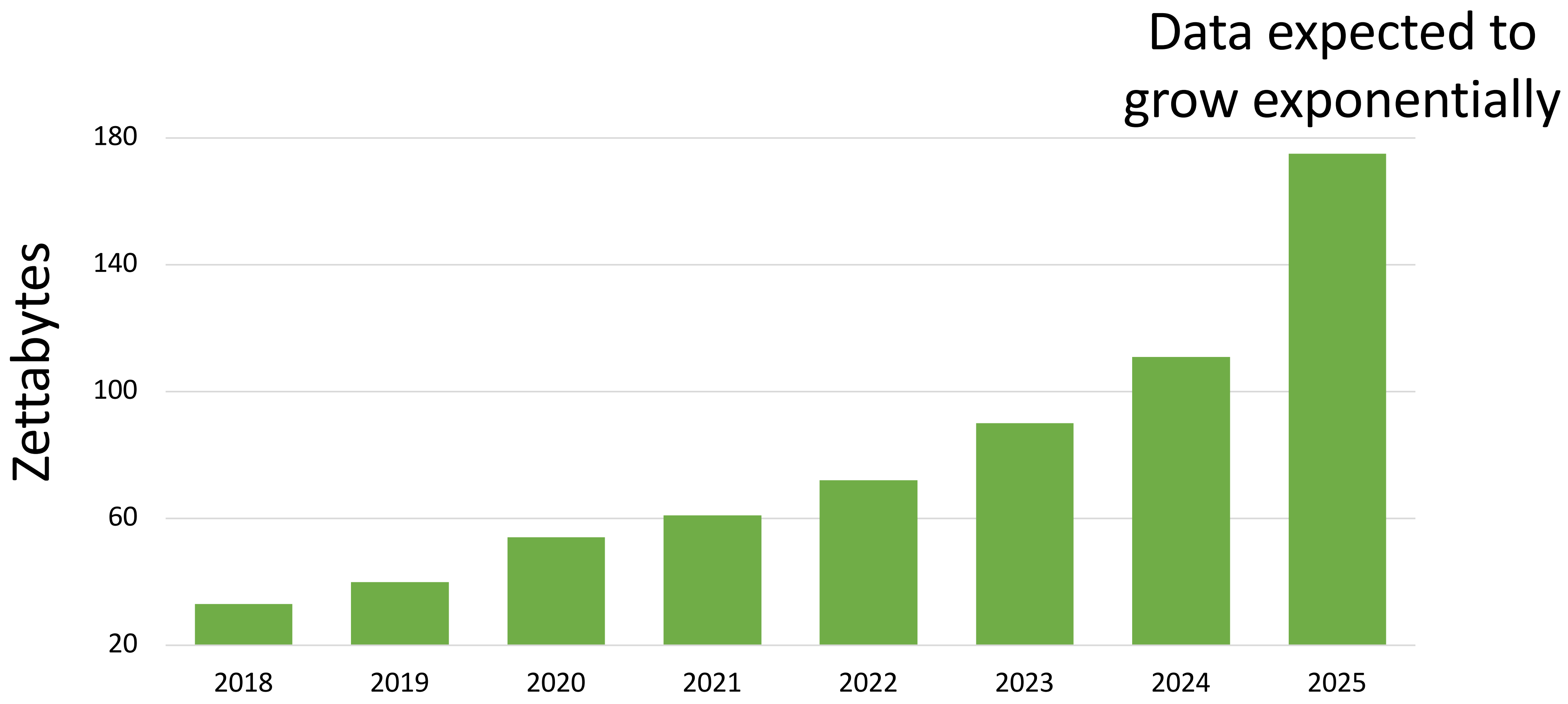
Oana Balmau

WoCC Keynote, December 12th, 2023

McGill

ML Commons

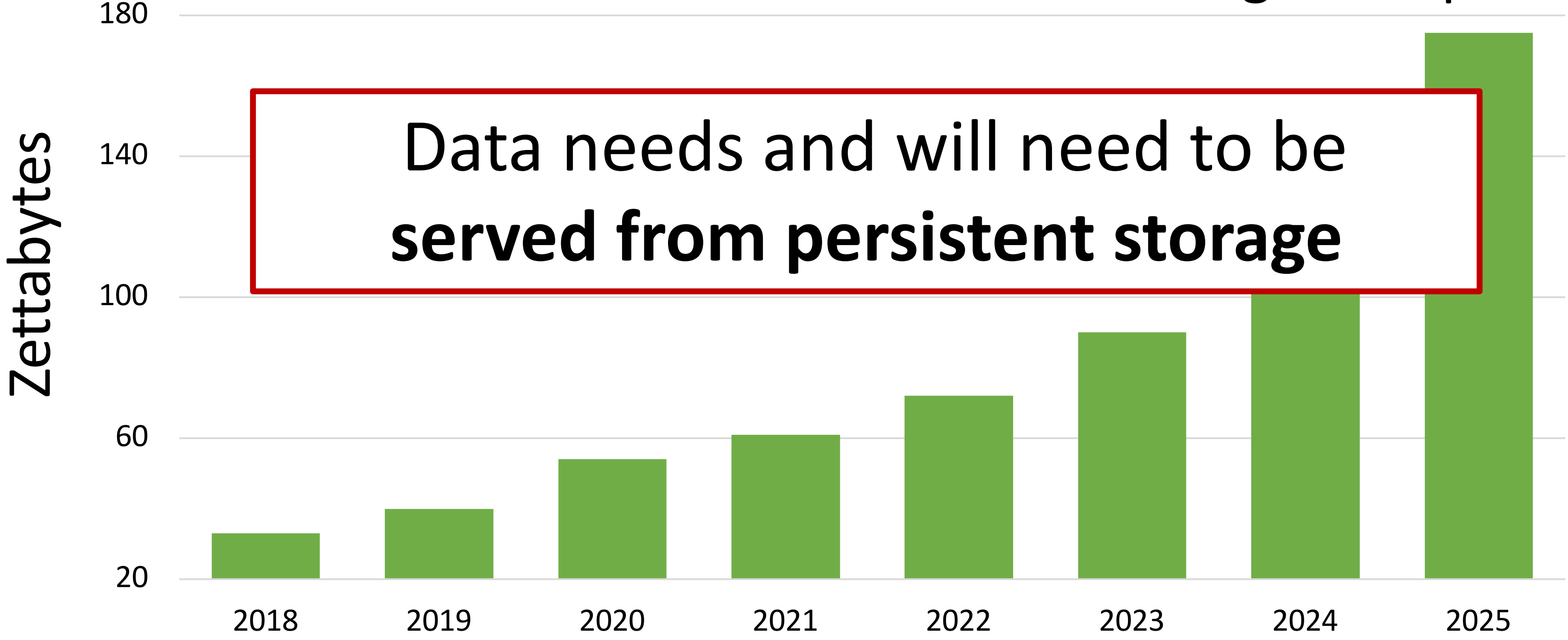
Humanity produces a lot of data



Source: IDC 2022

Humanity produces a lot of data

Data expected to
grow exponentially



Data needs and will need to be
served from persistent storage

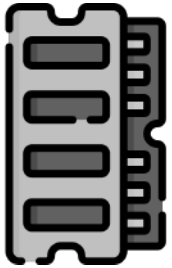
Source: IDC 2022

Data is the moving force of ML algorithms

… but in many projects the **storage decision is an afterthought**

Inefficient I/O can slow down ML Workloads

Dataset fits in system memory



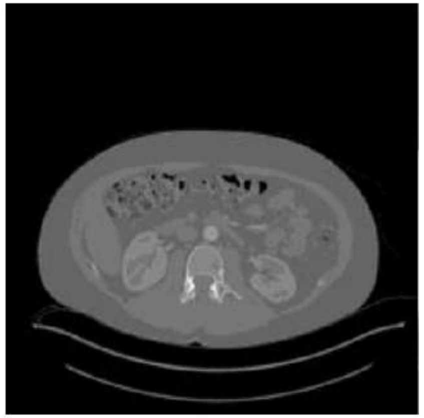
Dataset = 2x system memory

Training time increased by 3x

Example: Image Segmentation with 3D U-Net

An example of a coronal section of one of the training cases with its ground truth segmentation overlaid (kidney in red, tumor in blue).
Source https://arxiv.org/pdf/1912.01054.pdf

Medical image segmentation
2019 Kidney Tumor Segmentation Challenge (KiTS19)
CT scans from ~300 kidney tumor cases



Sample images from the KiTS19 dataset before (left) and after (right) preprocessing.
Source: https://arxiv.org/pdf/1908.02625.pdf

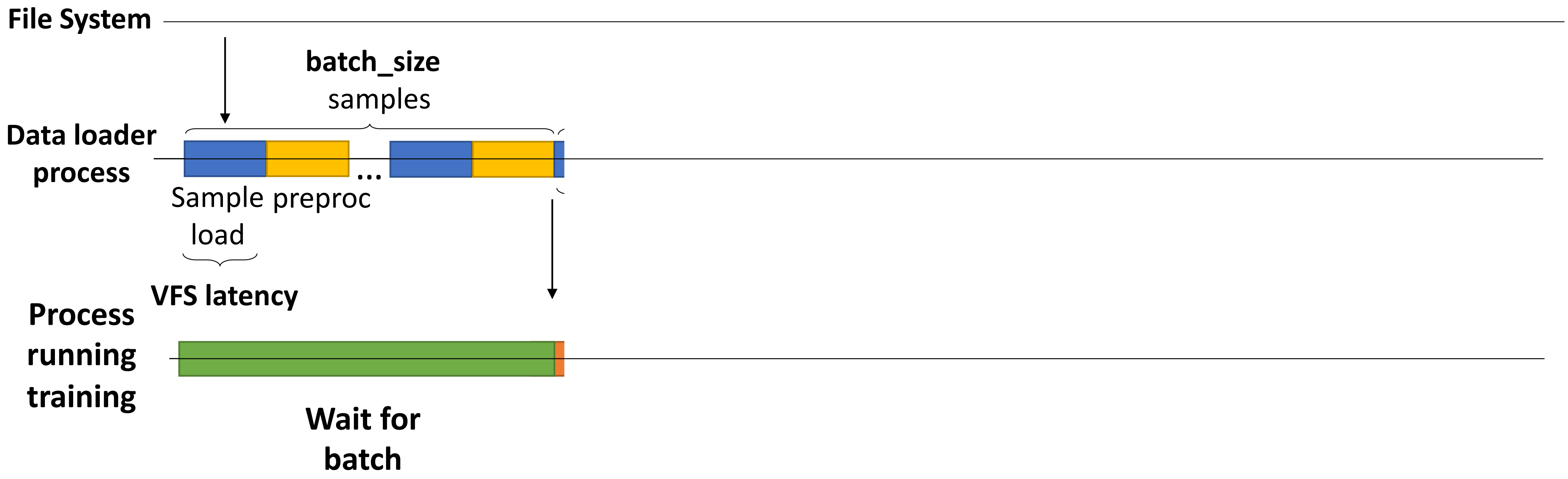
Example: Image Segmentation with 3D U-Net

File System

Data loader process

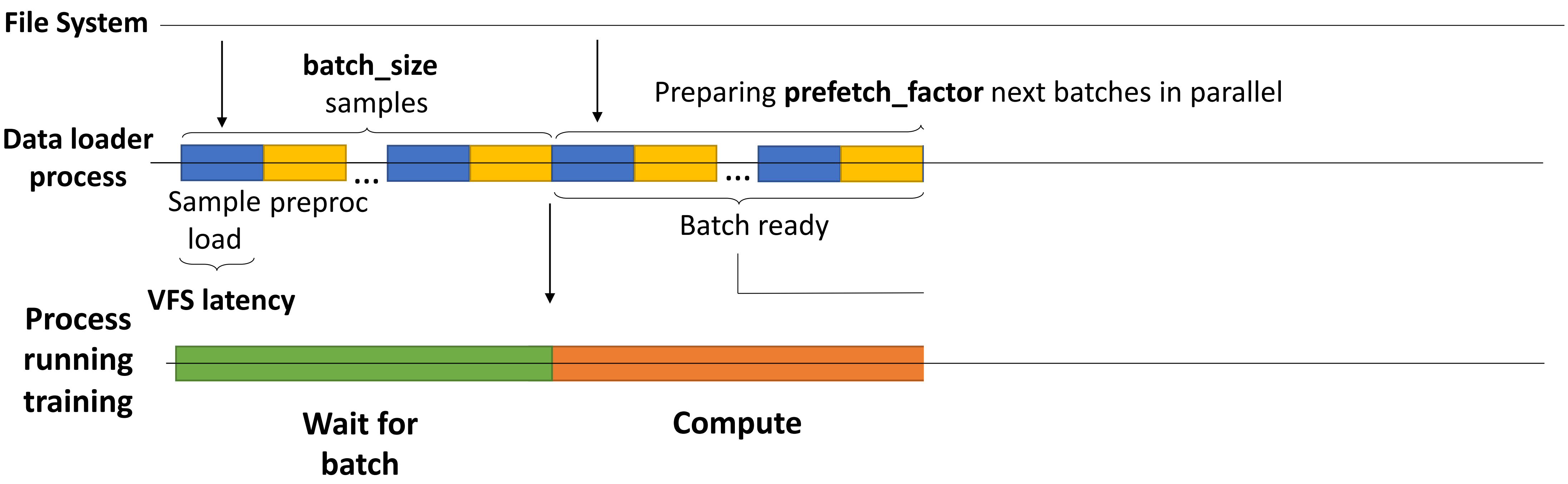
Process running training

Example: Image Segmentation with 3D U-Net

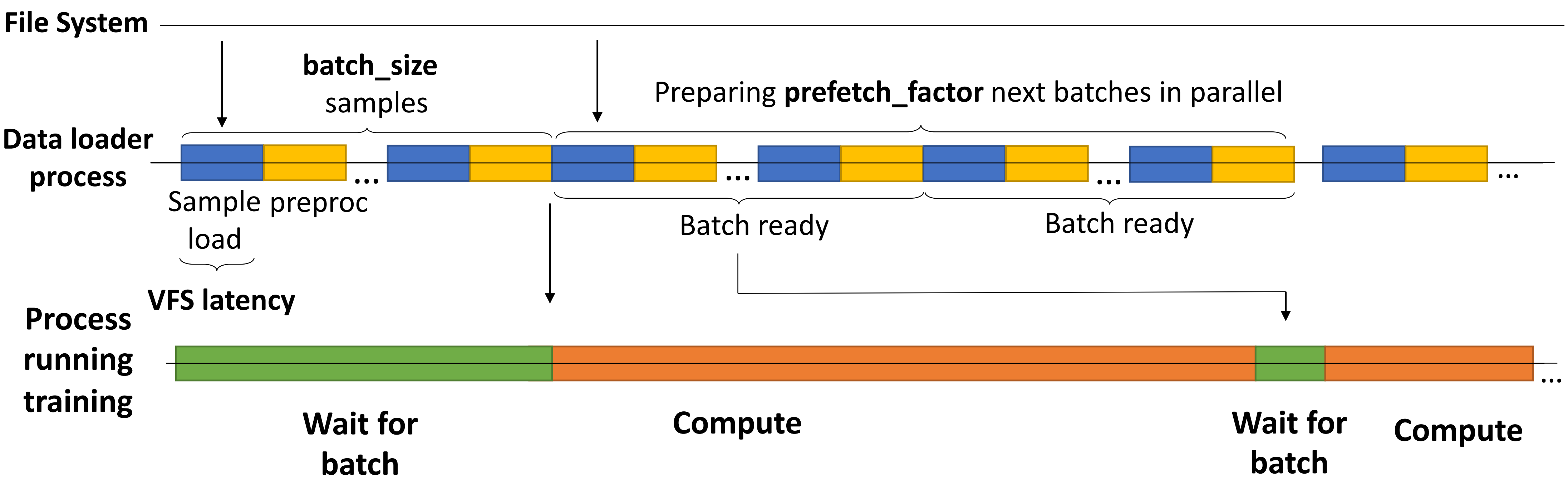


oana.balmau@mcgill.ca

Example: Image Segmentation with 3D U-Net



Example: Image Segmentation with 3D U-Net

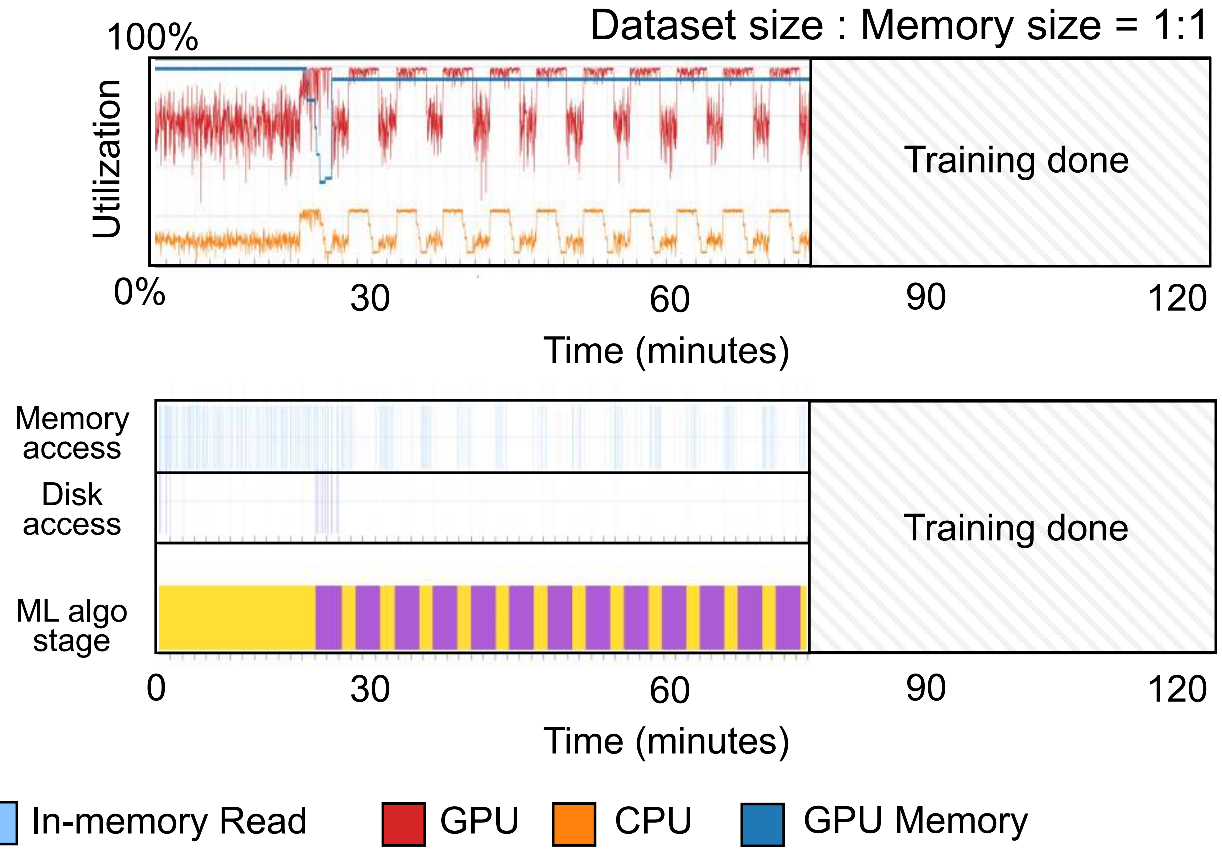


Inefficient I/O can slow down ML Workloads

Experiment setup

- DGX-1 server
  - 8 x V100 GPUs, 32GB GPU memory
  - 512GB DRAM
- Image segmentation workload:
  - Unet3D, Pytorch
  - MLPerf Training implementation
  - KiTS19 dataset

Dataset fits in system memory



Inefficient I/O can slow down ML Workloads

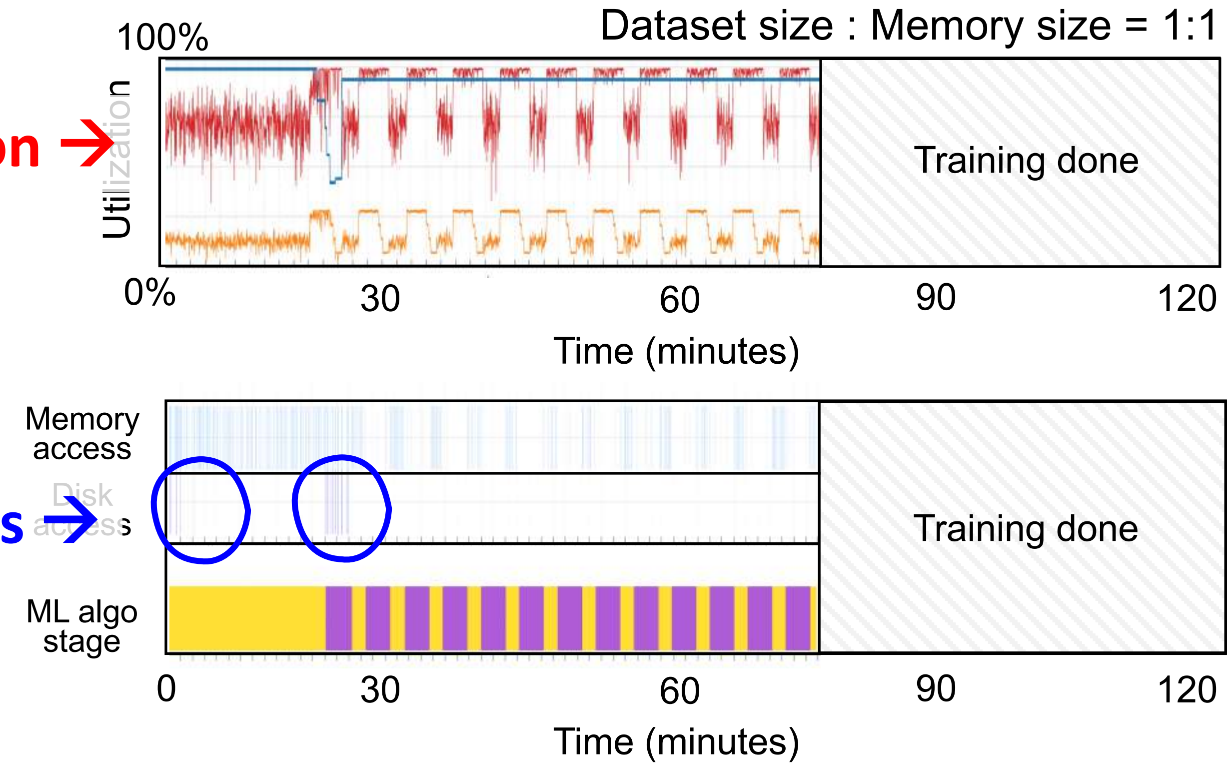
Experiment setup

- DGX-1 server
  - 8 x V100 GPUs, 32GB GPU memory
  - 512GB DRAM
- Image segmentation workload:
  - Unet3D, Pytorch
  - MLPerf Training implementation
  - KiTS19 dataset

High GPU utilization →

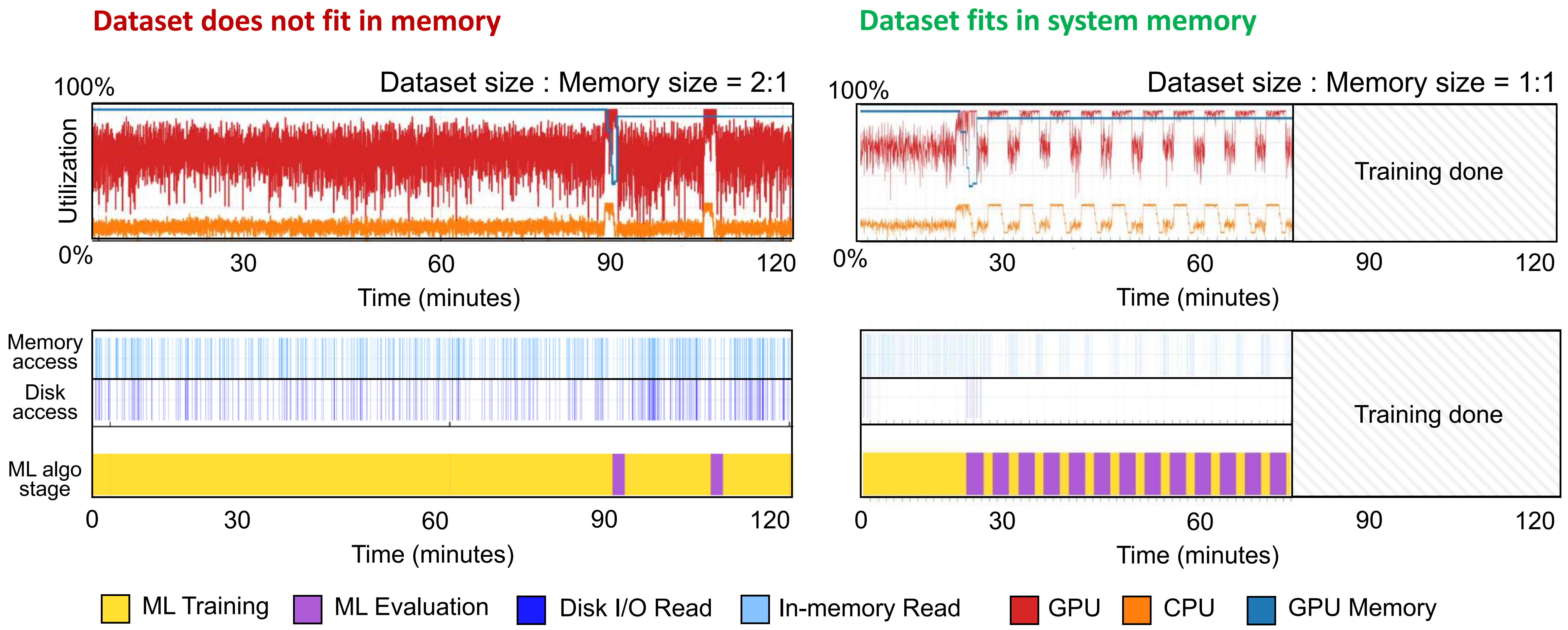
Little disk access →

Dataset fits in system memory

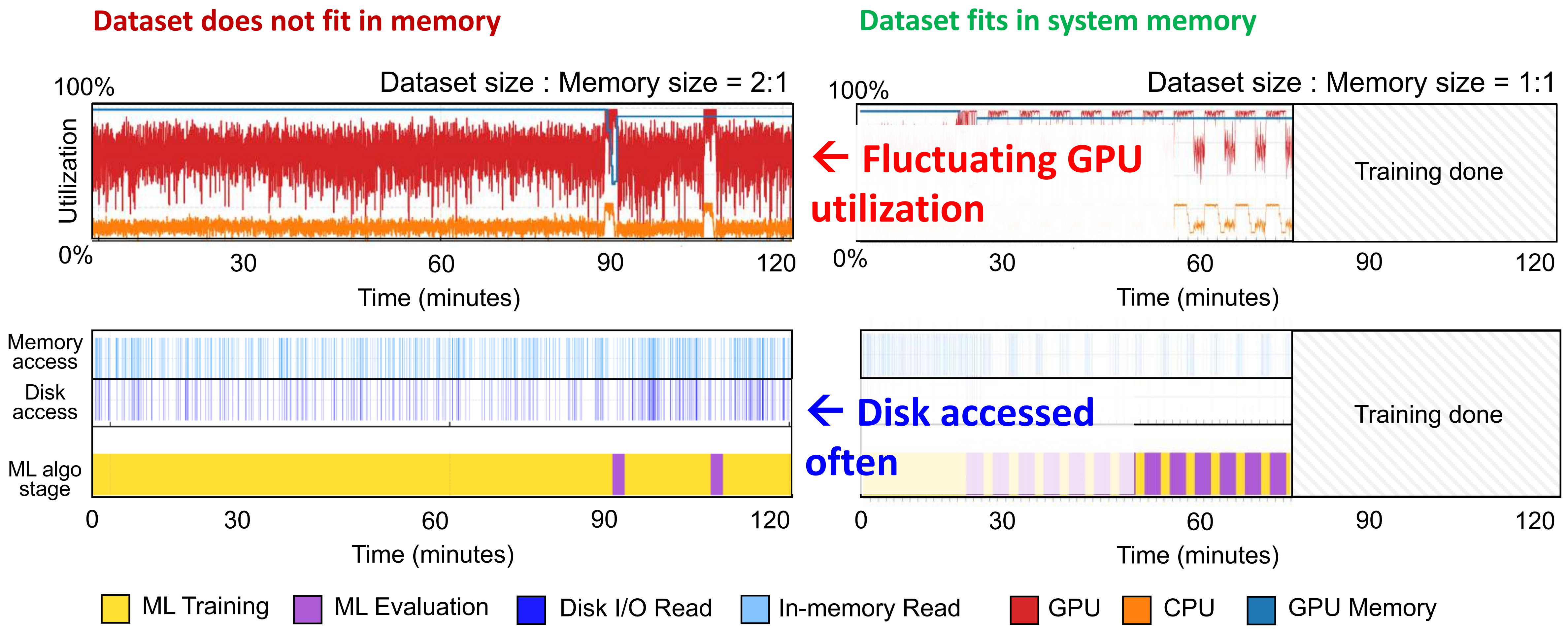


ML Training | ML Evaluation | Disk I/O Read | In-memory Read | GPU | CPU | GPU Memory

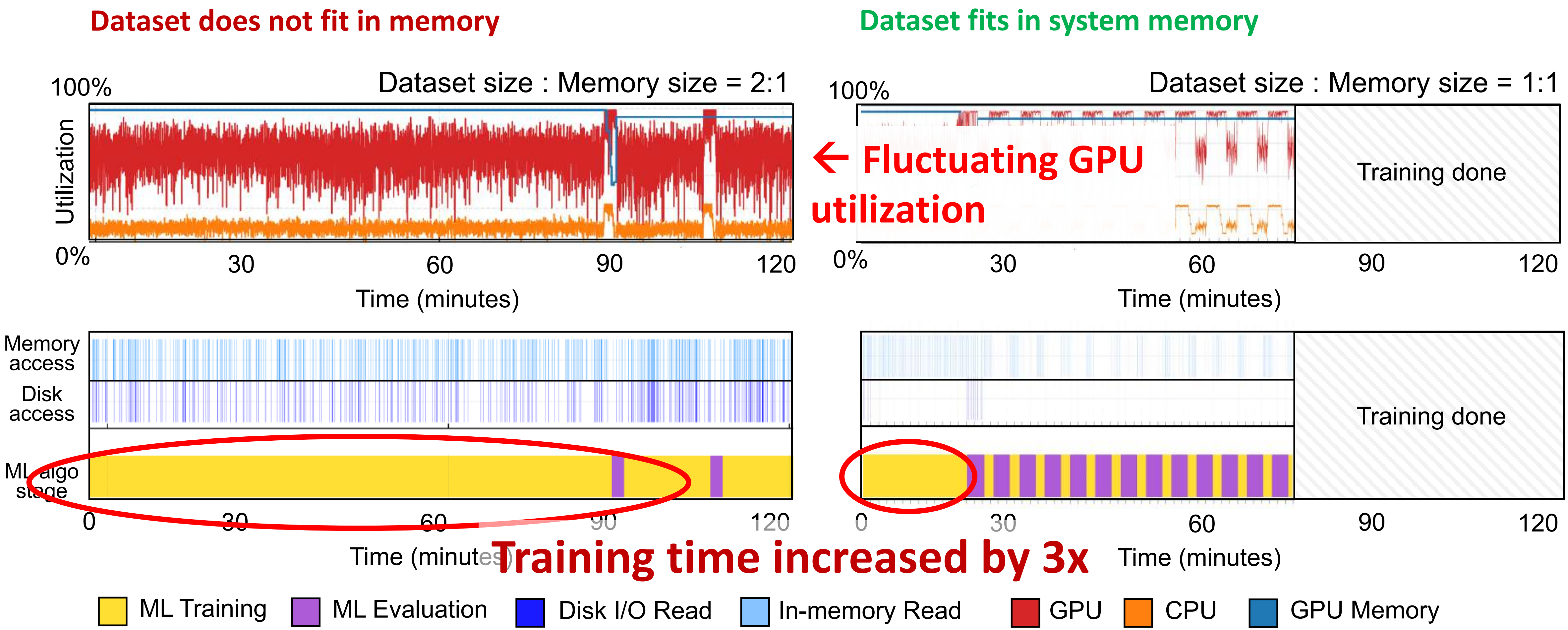
Inefficient I/O can slow down ML Workloads



Inefficient I/O can slow down ML Workloads



Inefficient I/O can slow down ML Workloads



Data is the moving force of ML algorithms

… but in many projects the **storage decision is an afterthought**

Why create an ML Storage benchmark?

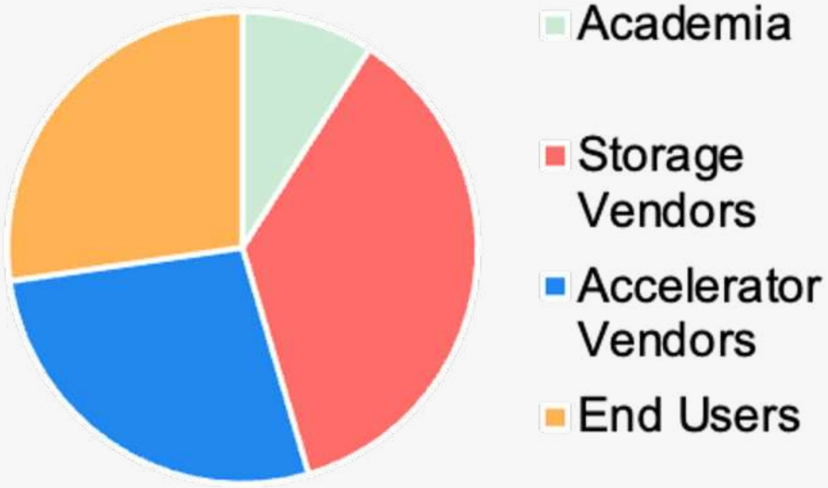
Why create an ML Storage benchmark?

- **Understand storage bottlenecks in ML workloads and propose optimizations**

- **Help AI/ML researchers and practitioners make an informed *storage* decision**

MLPerf Storage Working Group (132 members)

Who are we?



McGill NVIDIA intel Hewlett Packard Enterprise NUTANIX SAMSUNG WEKA PANASAS Micron Argonne NATIONAL LABORATORY Lawrence Livermore National Laboratory BERKELEY LAB Lawrence Berkeley National Laboratory tenstorrent HUAWEI NetApp Red Hat

Current ML/AI benchmarks

Many existing ML/AI benchmarks

DeepMind Lab

MLPerf

OpenAI

DLBT

PMLDB

DAWNBench

Current ML/AI benchmarks

- Focus on **end-to-end testing**

  → hard to isolate value of each component

- Insist on **training and inference** speed

  → tend to simplify storage

  → ignore pre-processing

- **Expensive accelerators** needed to run
- Require **extensive entry knowledge**

DeepMind Lab

MLPerf

OpenAI

DLBT

PMLDB

DAWNBench

Benchmark Vision

Existing benchmarks

Focus on **end-to-end testing**

Simplified storage setup

Expensive accelerators needed to run

Require **extensive entry knowledge**

Our work

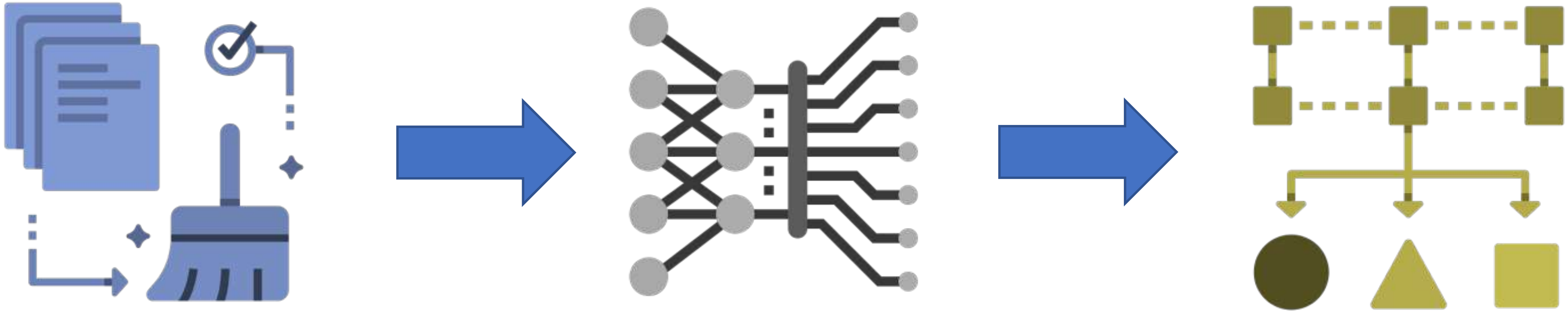
Focus on **storage impact in ML/AI**

Realistic **storage & pre-processing** settings

No accelerator required to run

Minimal AI/ML knowledge required

Stages of the ML Pipeline

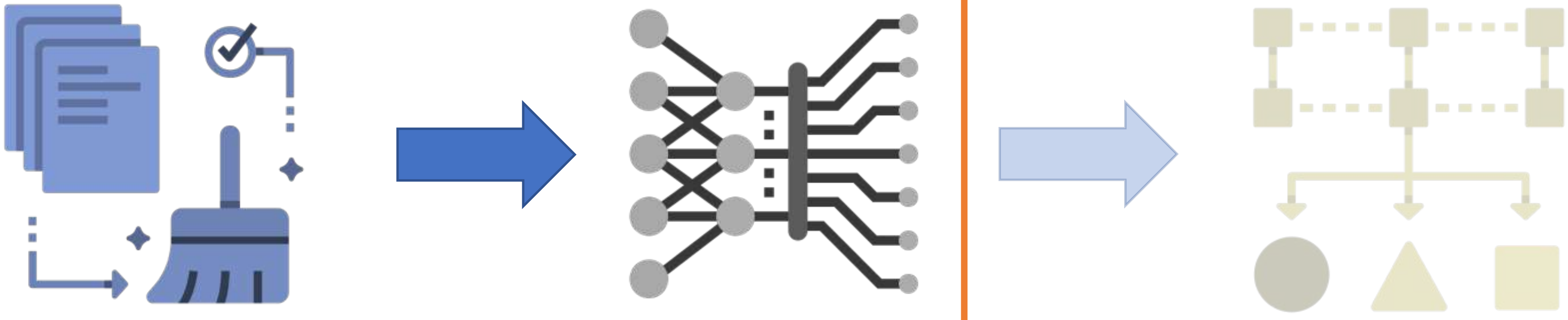


Data cleaning & pre-processing **Training** **Inference**

Stages of the ML Pipeline

I/O intensive [1,2] – Our focus

*As much as **50% of the Watts** can go into storage and data cleaning [2]*



Data cleaning & pre-processing

Training

Inference

*[1] Murray et al. **tf.data: A Machine Learning Data Processing Framework**, VLDB 21.*
*[2] Zhao et a. **Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training** ISCA 22.*

Data Pipeline in ML: Pre-processing

Storage resources

Disk

Memory

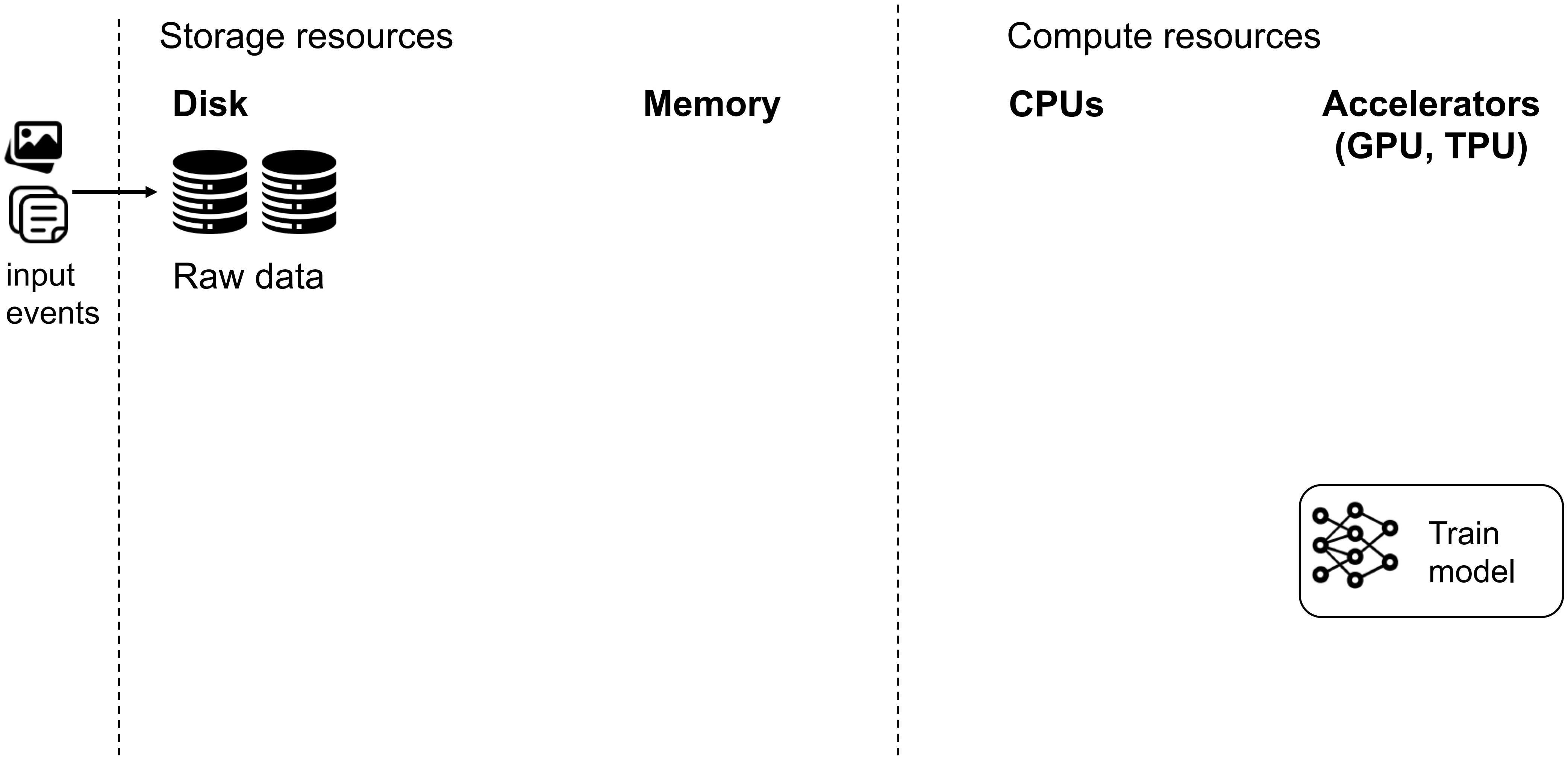
Compute resources

CPUs

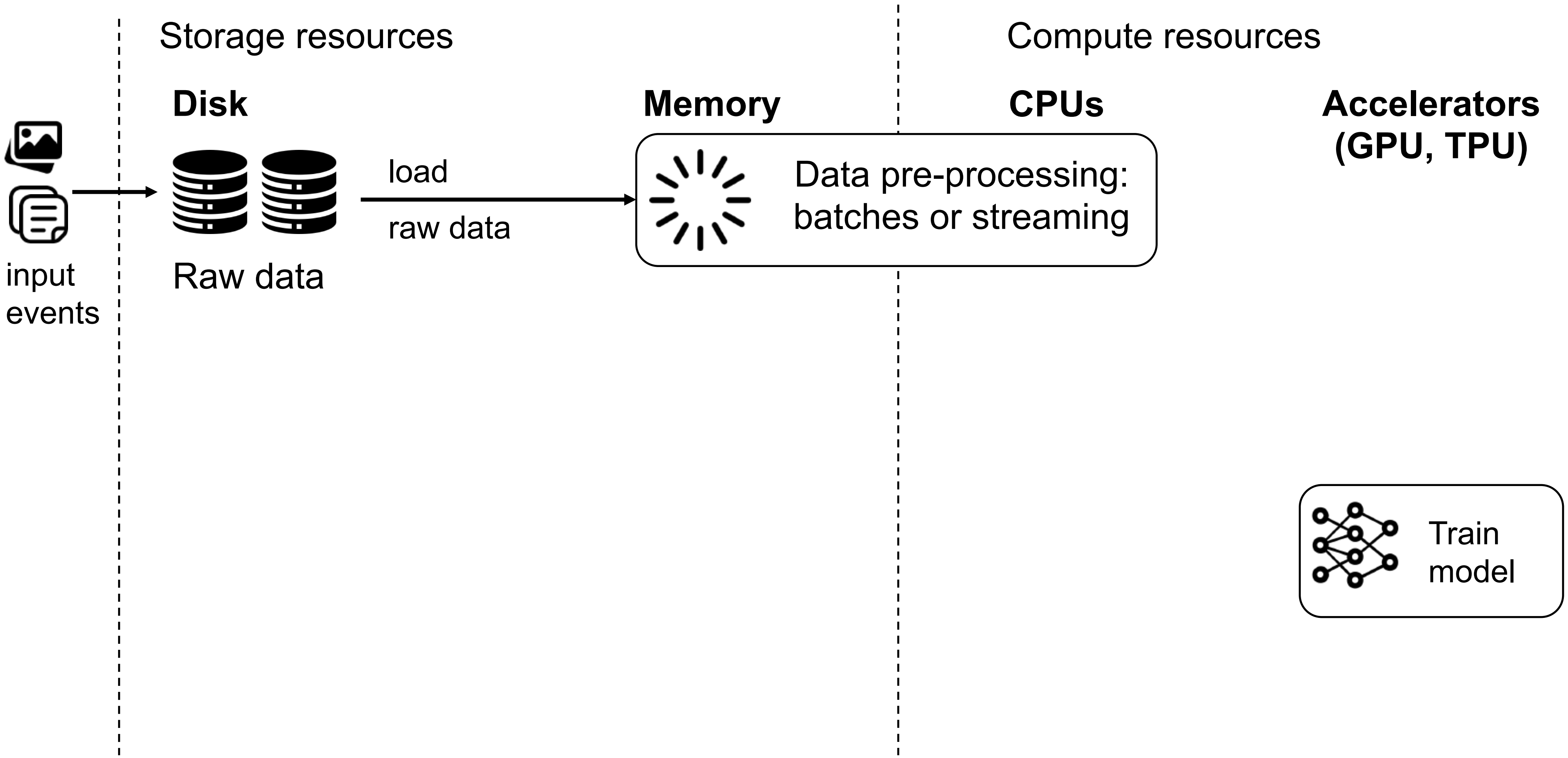
Accelerators (GPU, TPU)

Train model

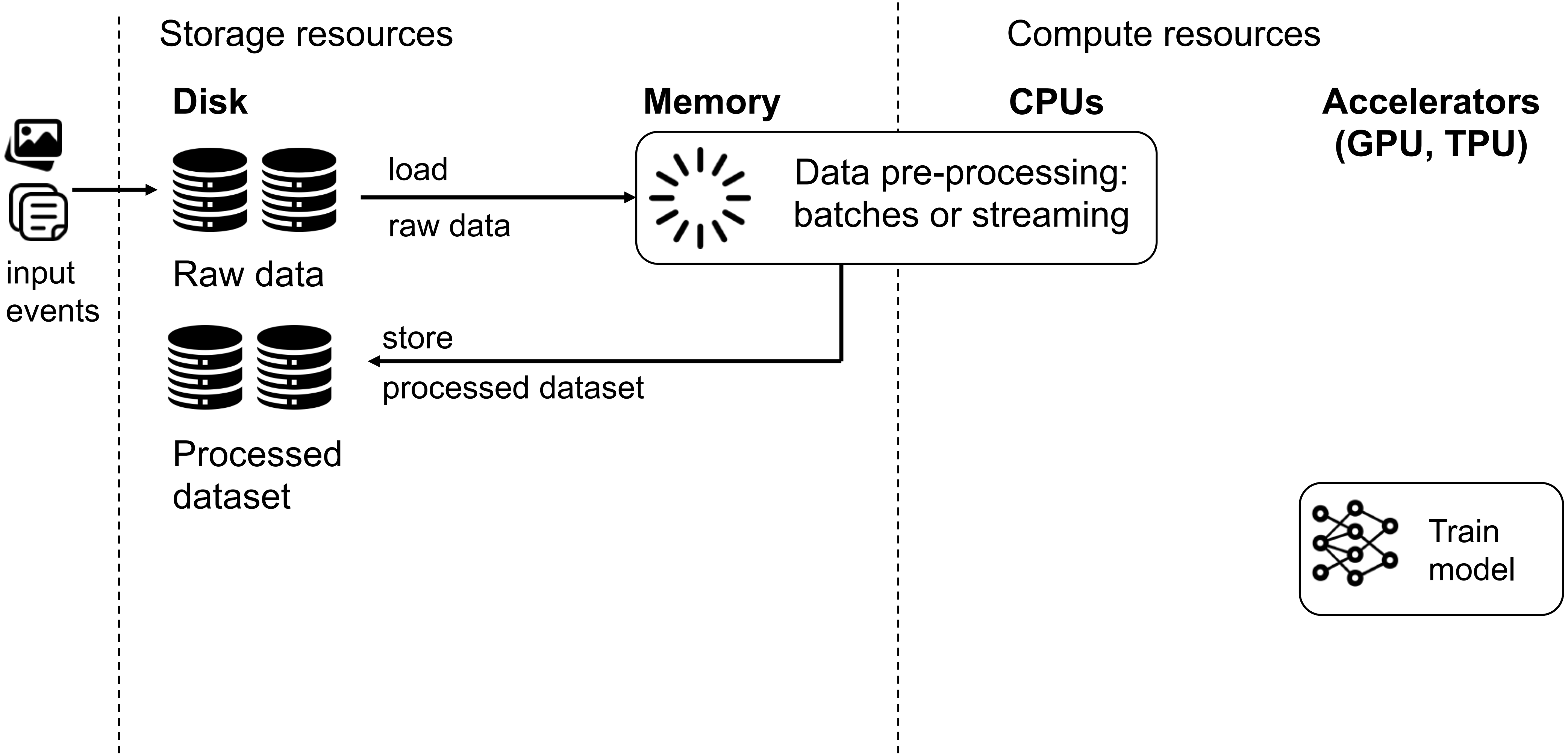
Data Pipeline in ML: Pre-processing



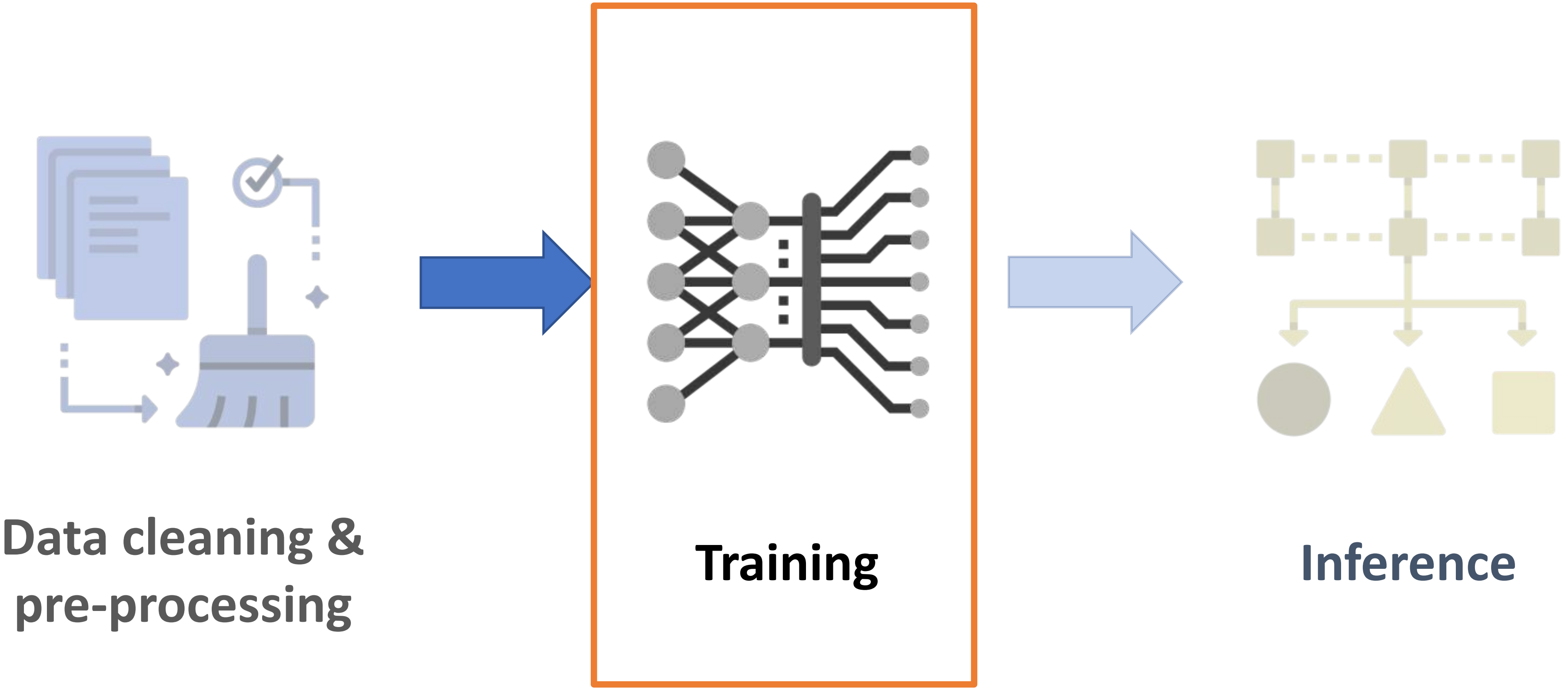
Data Pipeline in ML: Pre-processing



Data Pipeline in ML: Pre-processing



Stages of the ML Pipeline

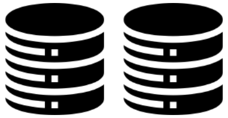


[1] Murray et al. ***tf.data: A Machine Learning Data Processing Framework****, VLDB 21.*
[2] Zhao et a. ***Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training*** *ISCA 22.*

Data pipeline in ML: Training

Storage resources

Disk



Cleaned
dataset

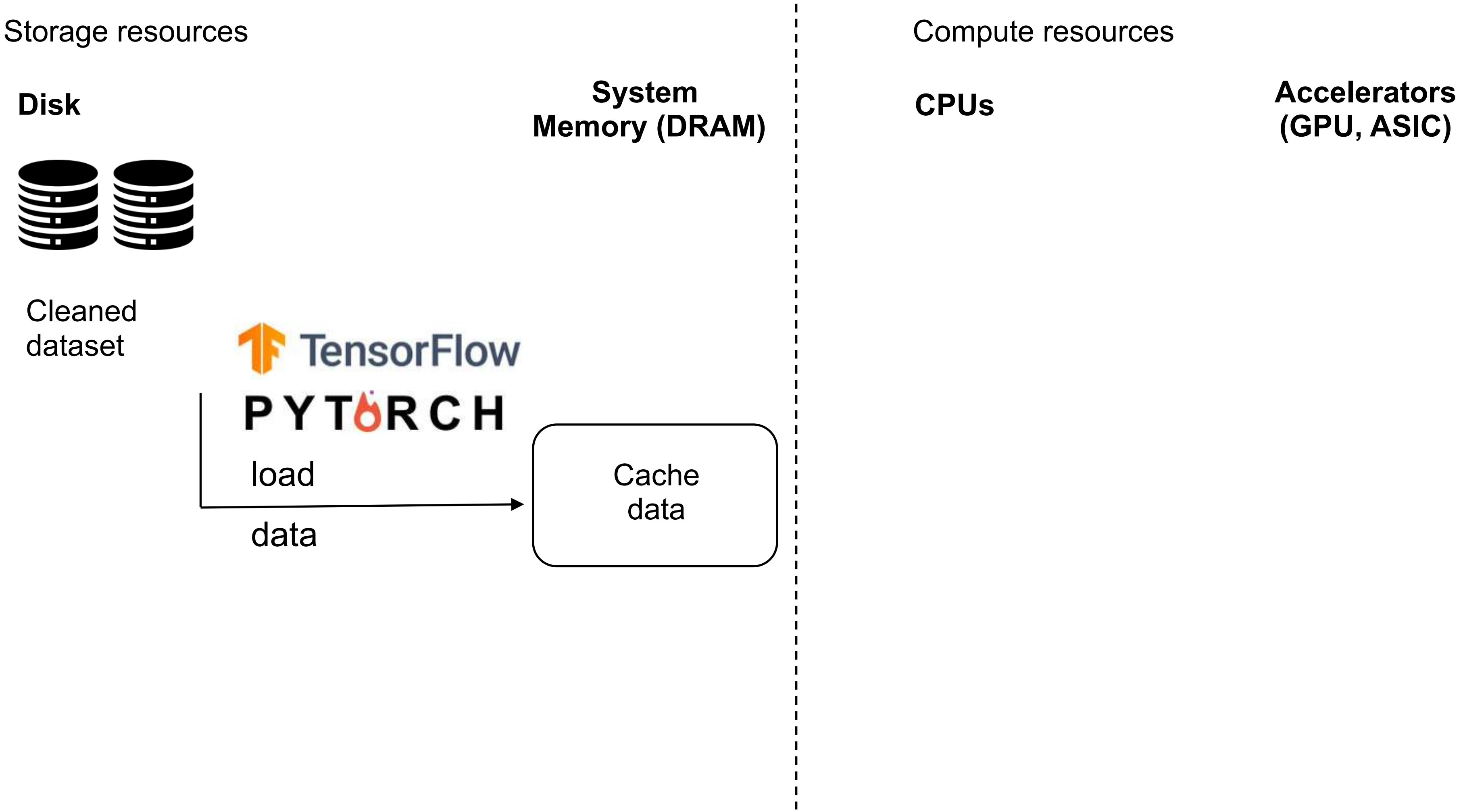
System Memory (DRAM)

Compute resources

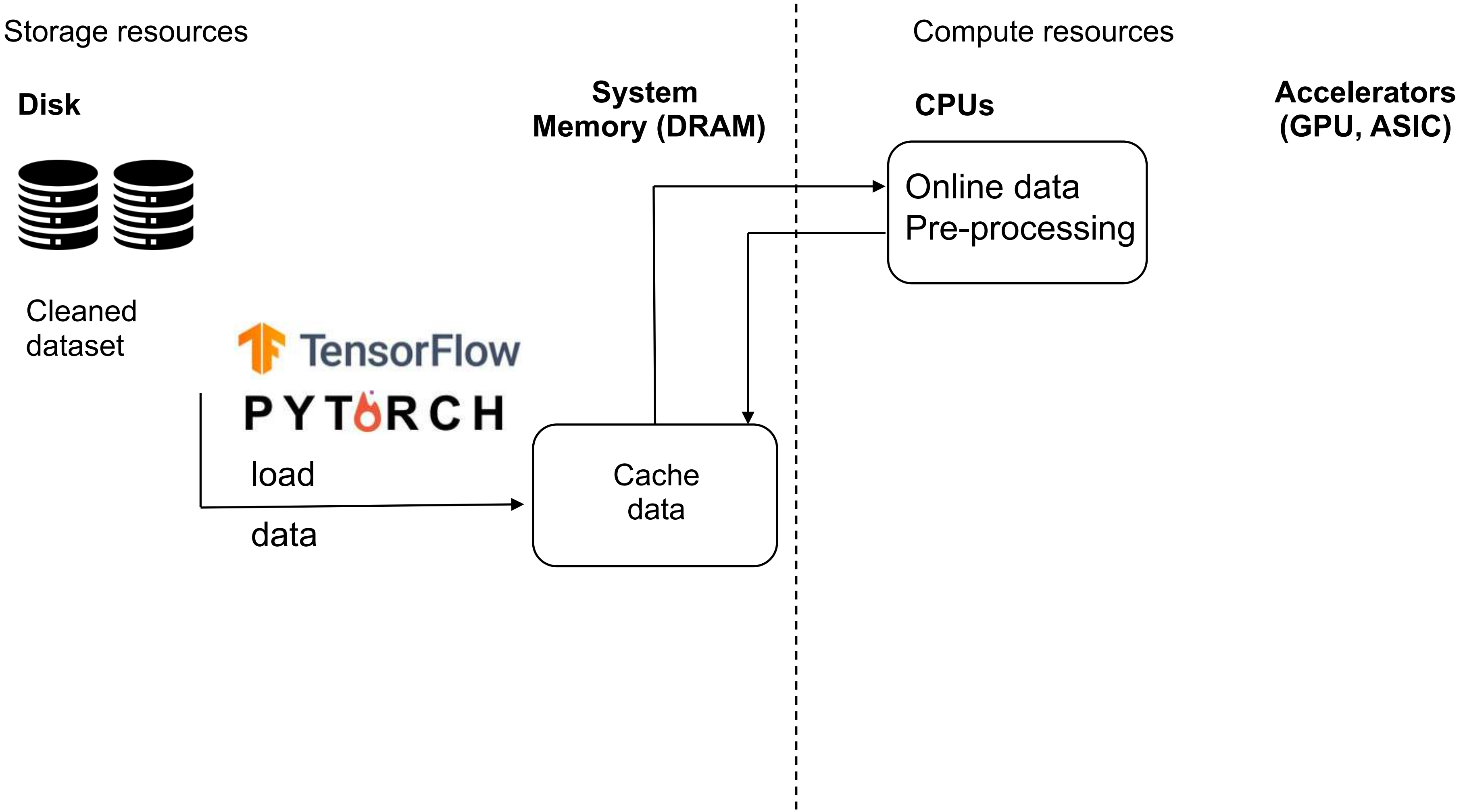
CPUs

Accelerators (GPU, ASIC)

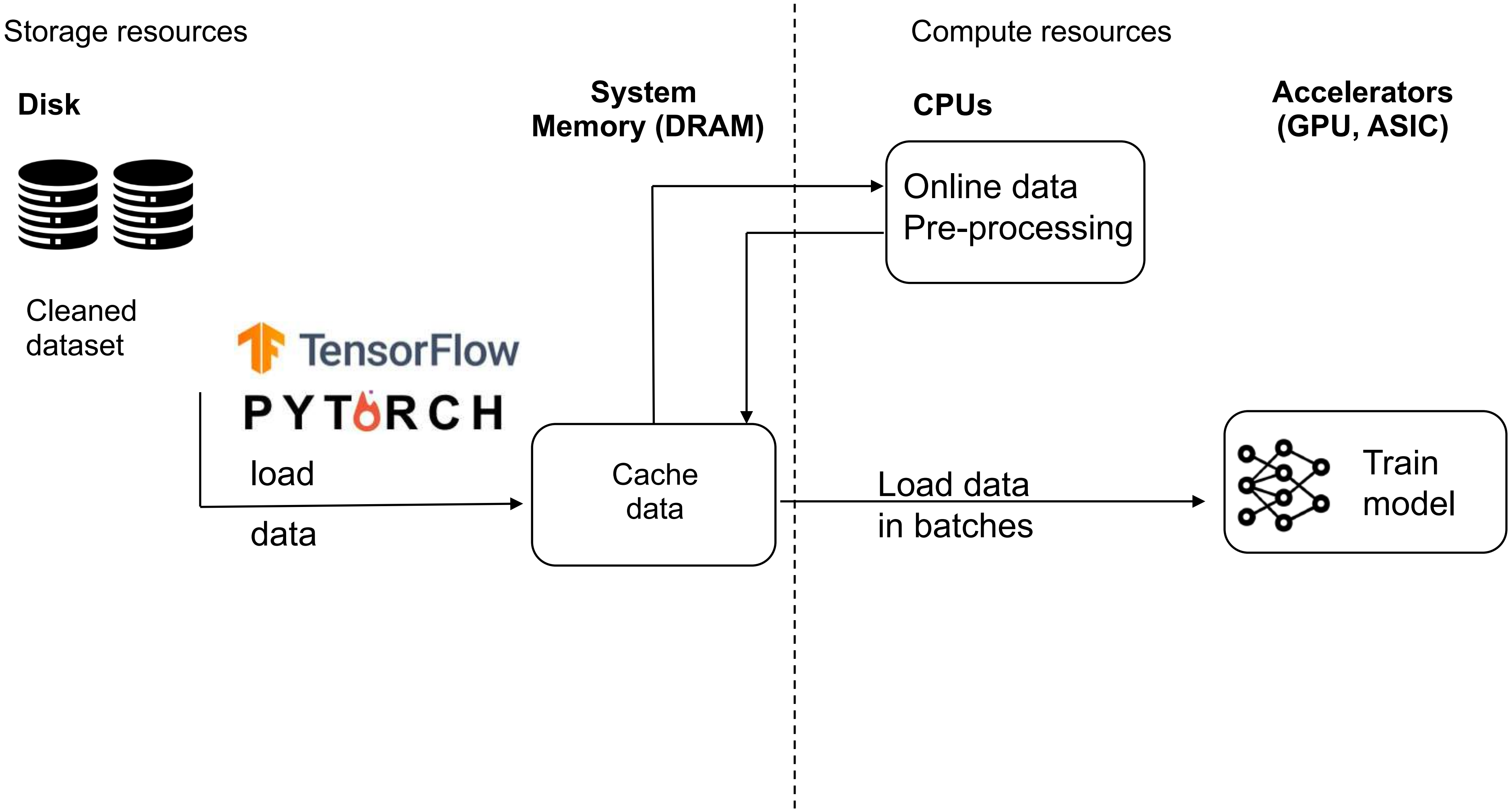
Data pipeline in ML: Training



Data pipeline in ML: Training



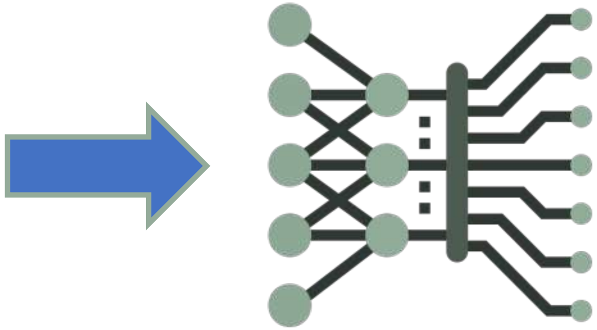
Data pipeline in ML: Training



MLPerf Storage

Data cleaning & pre-processing

MLPerf Storage V0.5



Training

Focus on **storage impact in ML/AI**

Realistic **storage** settings in

training phase

No accelerator required to run

Minimal AI/ML knowledge

MLPerf Storage – workloads

Workload	**Image segmentation**	**Natural language processing**	**Recommender Systems**
Model	3D U-Net	BERT	DLRM
Seed data	KiTS19 Set of images	Wikipedia 2020 Text	Criteo Terabyte Click logs
Framework	Pytorch	Tensorflow	Pytorch
I/O behavior	Random access inside many small files	Sequential access of small subset of files, streamed.	Random access inside one large file

https://github.com/mlcommons/storage

- **Single node**
- Many **simulated accelerators.**
- **Synthetic datasets** generated from real dataset seed.
- **Local storage**

MLPerf Storage – Benchmark metric

Must capture dynamics between storage and compute.

MLPerf Storage – Benchmark metric

Must capture dynamics between storage and compute.

Storage-centric metrics

- ✓ IOPS
- ✓ Latency
- ✓ Read/Write throughput
- ✓ Capacity

Compute-centric metrics

- ✓ Training time
- ✓ Trained model accuracy
- ✓ Accelerator utilization

☹Neither metric is enough to capture the storage-compute relationship

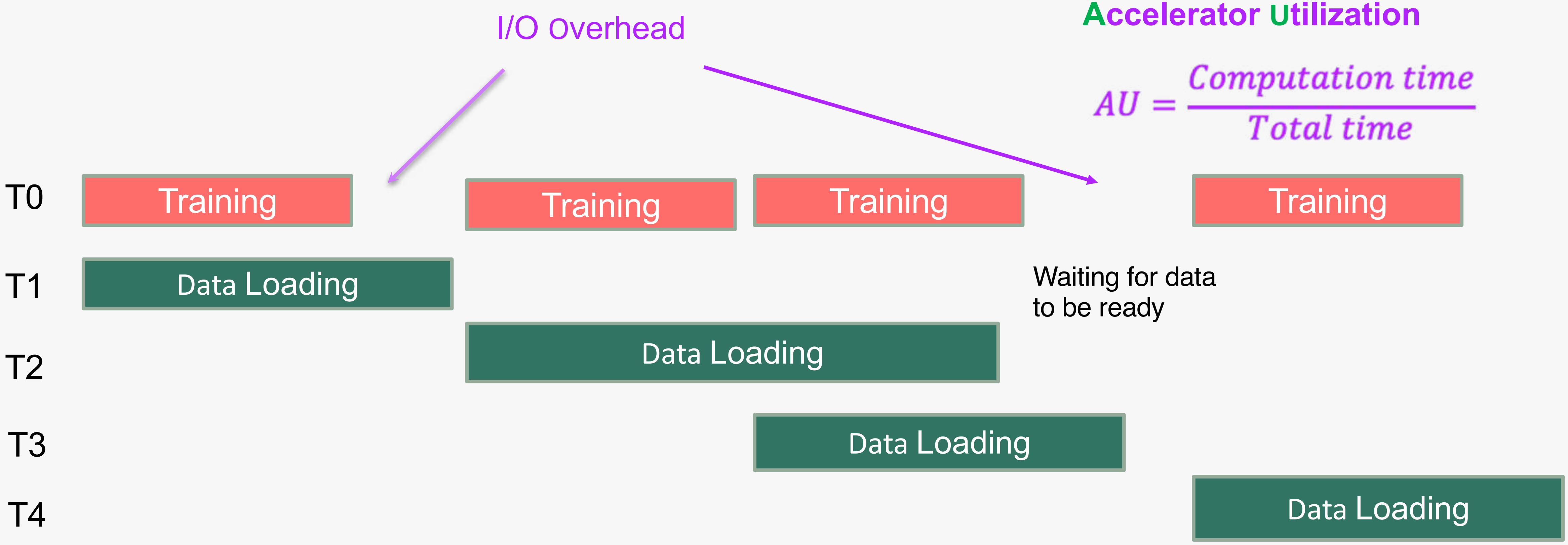
- Storage metrics too generic. Cannot capture dynamics of ML workloads.
- Compute-centric metrics too narrow (e.g., no notion of dataset size).

Proposed metric

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

Accelerator Utilization

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

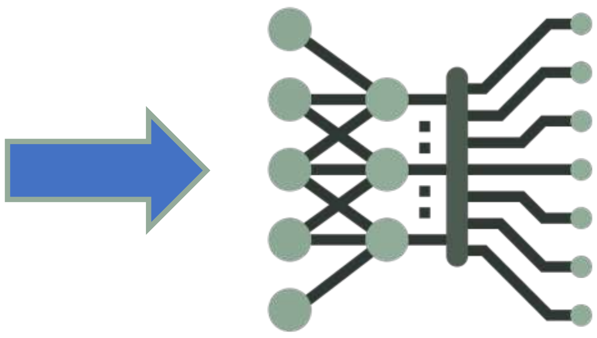


Goal of benchmark:
Maximize samples / second, given an **Accelerator Utilization > 90%** at a certain scale.

MLPerf Storage

Data cleaning & pre-processing

MLPerf Storage V1



Training

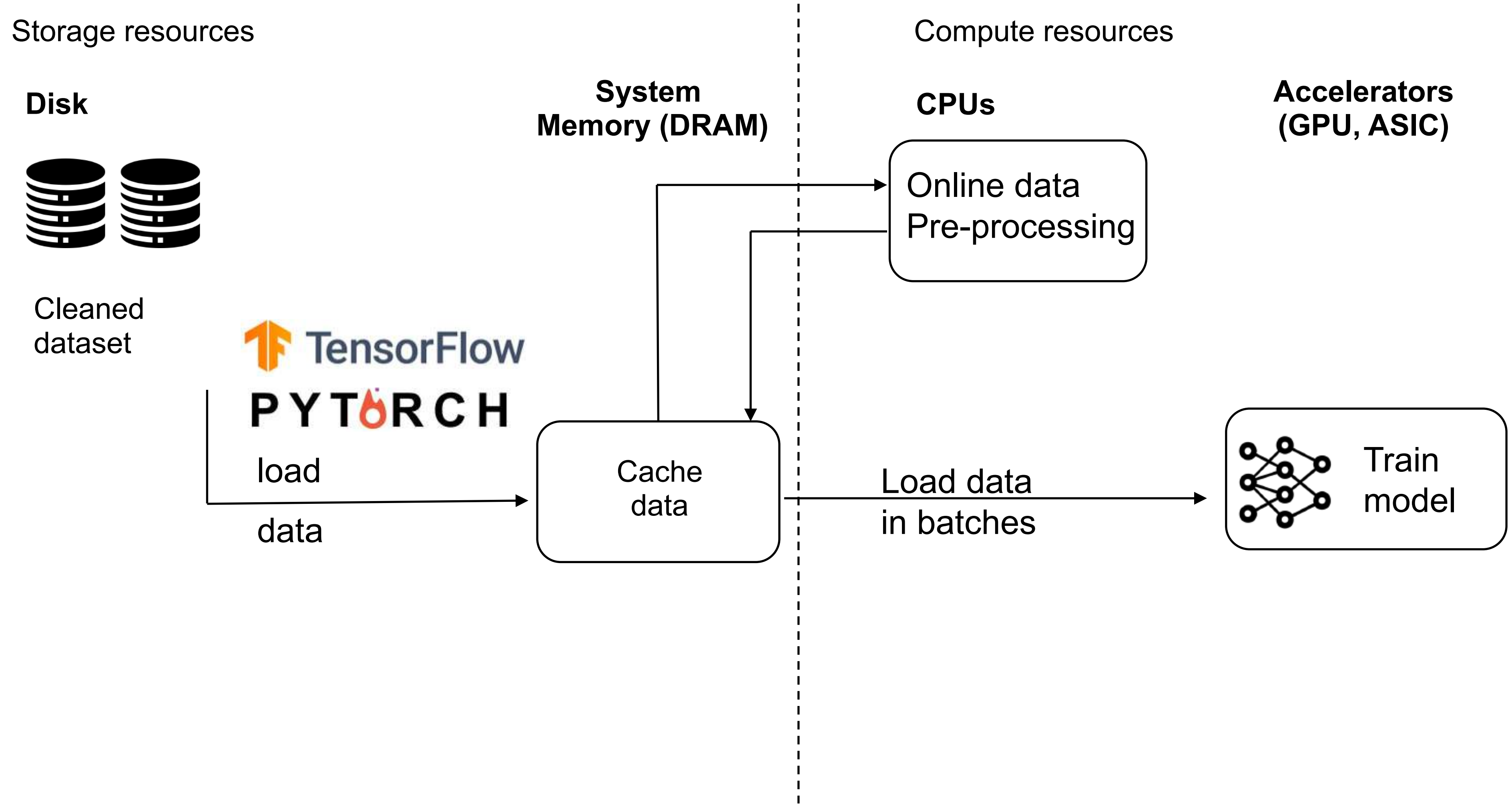
Focus on **storage impact in ML/AI**

Realistic **storage** settings in **training phase**

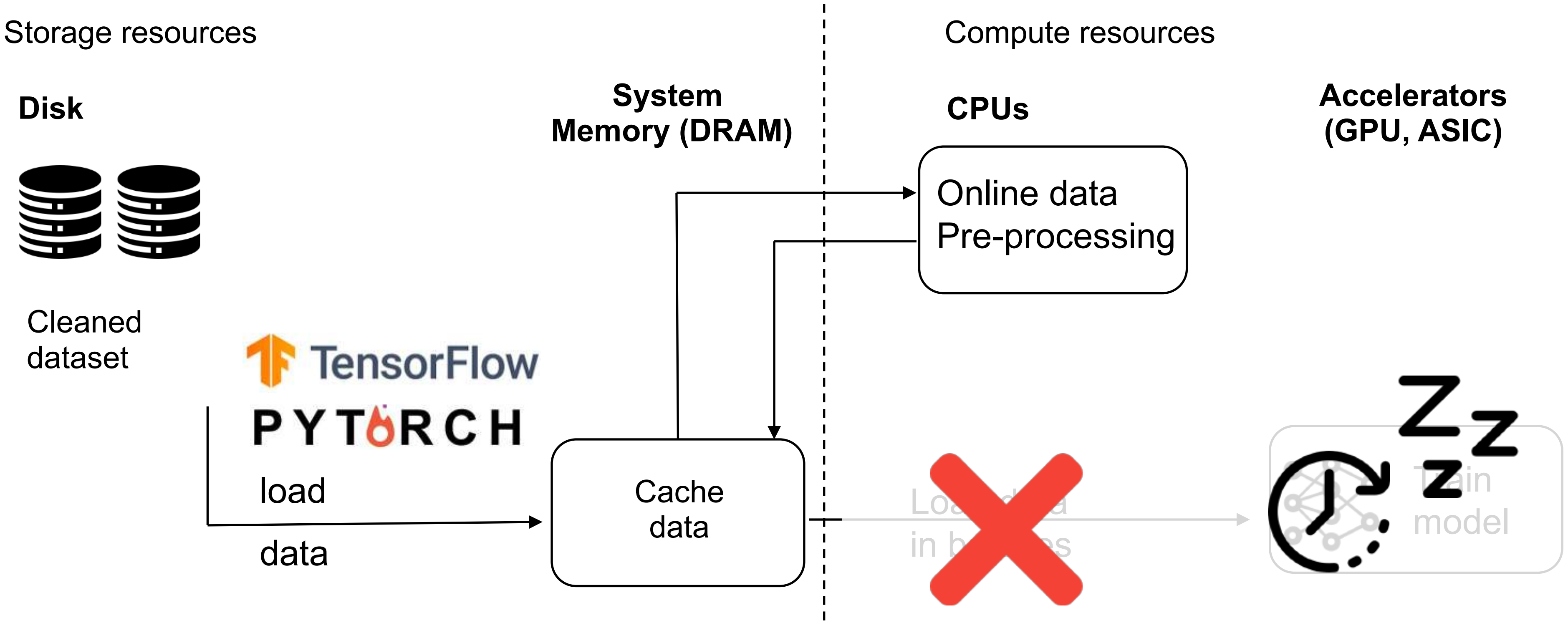
No accelerator required to run

Minimal AI/ML knowledge

Data pipeline in ML: Training



Data pipeline in MLPerf Storage benchmark

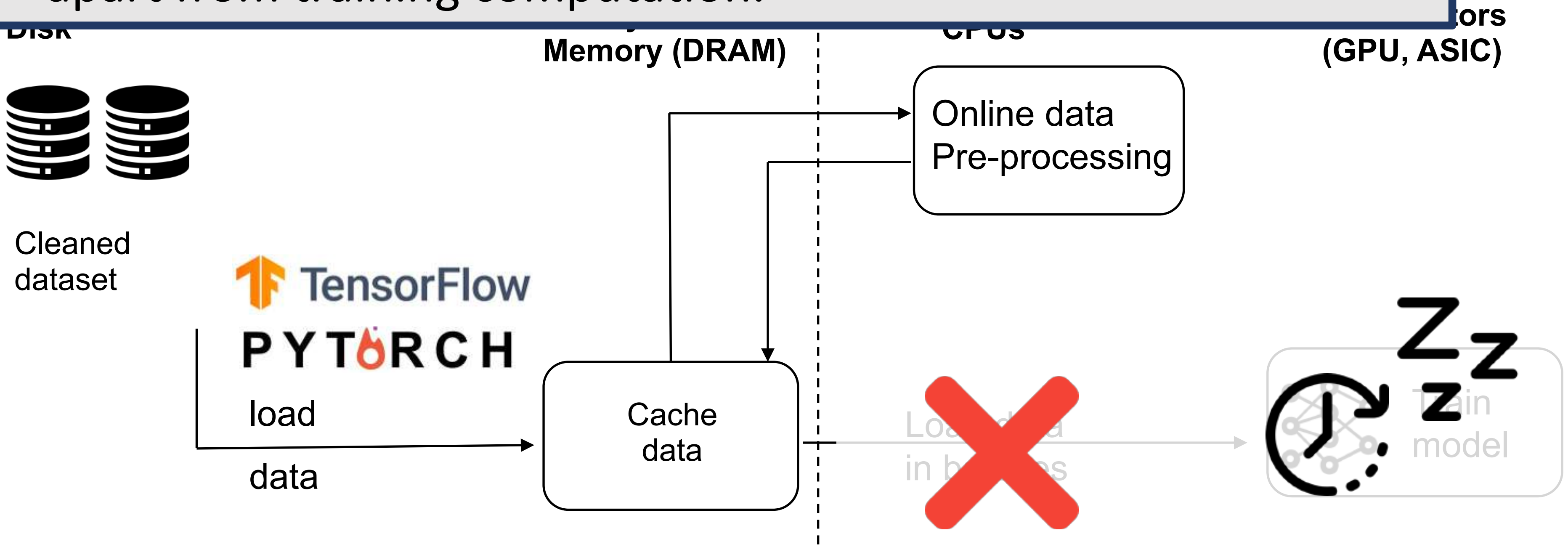


Benchmark is built as an extension of DLIO [1]

[1] H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.

Data pipeline in MLPerf Storage benchmark

✓ Realistic storage settings: nothing changes in data pipeline, apart from training computation.

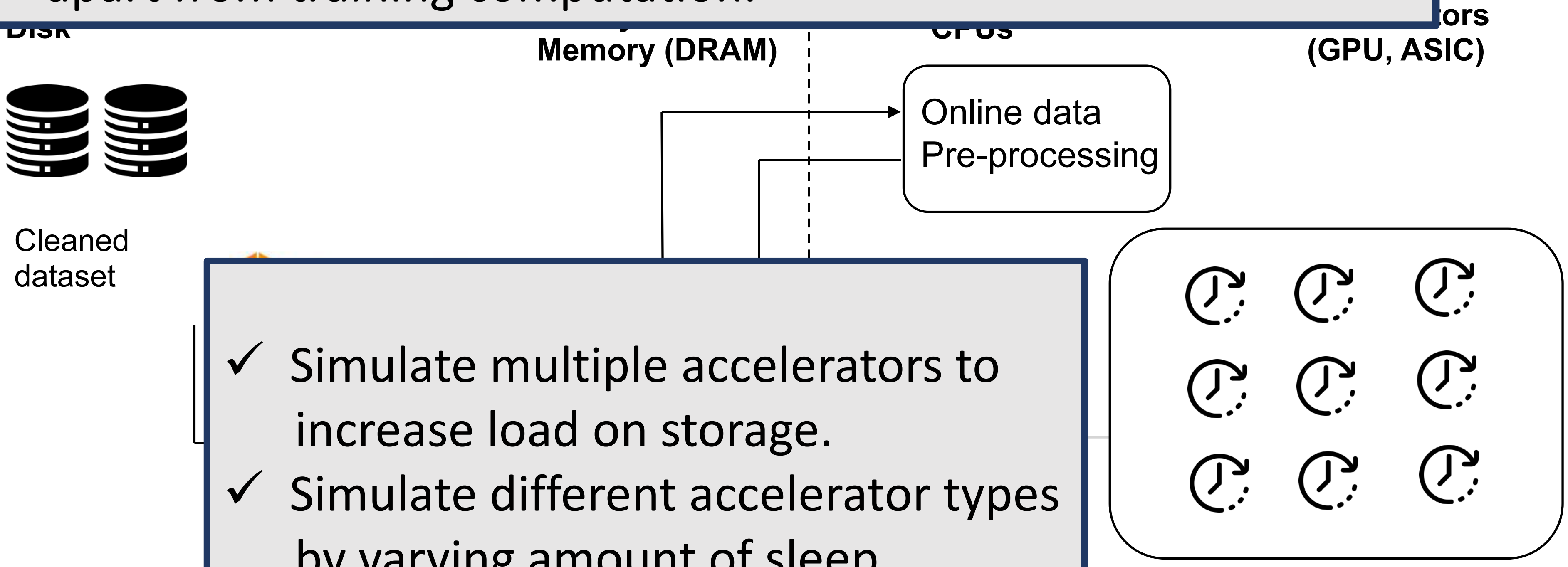


Benchmark is built as an extension of DLIO [1]

[1] H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.

Data pipeline in MLPerf Storage benchmark

✓ Realistic storage settings: nothing changes in data pipeline, apart from training computation.



✓ Simulate multiple accelerators to increase load on storage.

✓ Simulate different accelerator types by varying amount of sleep

Benchmark is built as an extension of DLIO [1]

[1] H. Devarajan, H. Zheng, et al. DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications, CCGrid '21.

Experimental Evaluation

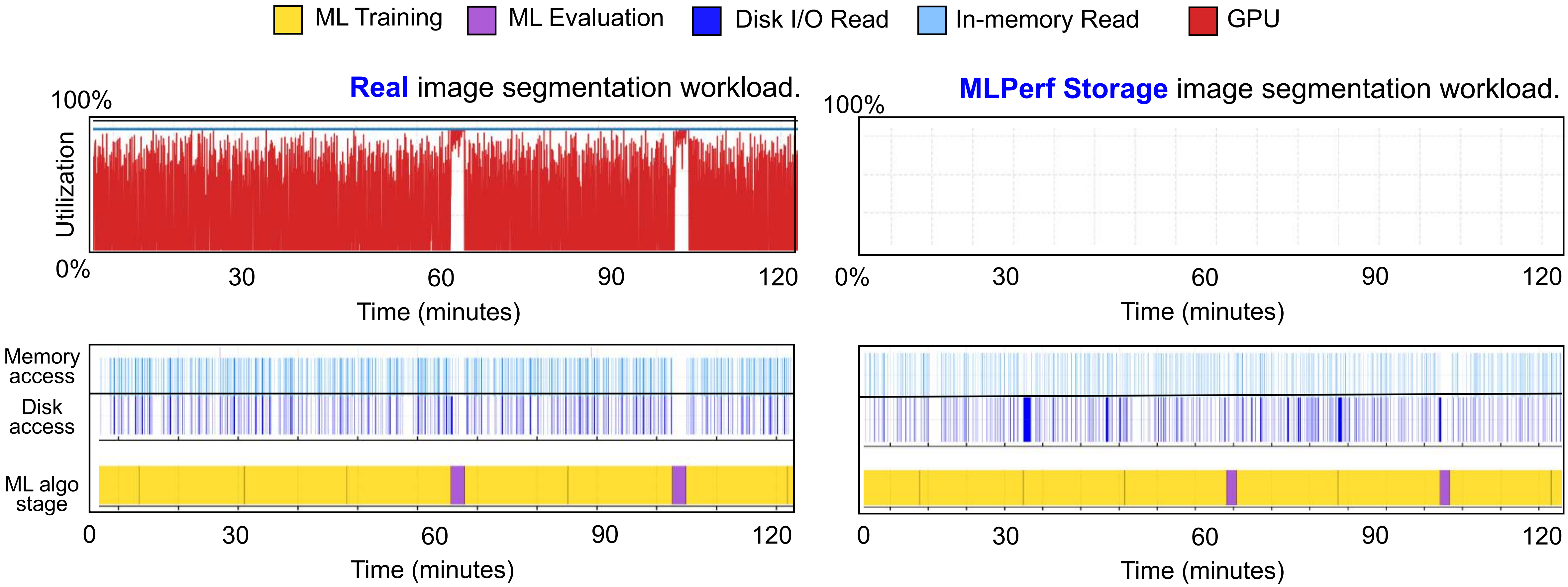
- DGX-1 server
  - 8 x V100 GPUs, 32GB GPU memory
  - 512GB DRAM

- Dataset size : Host memory size = 2:1

3D U-Net

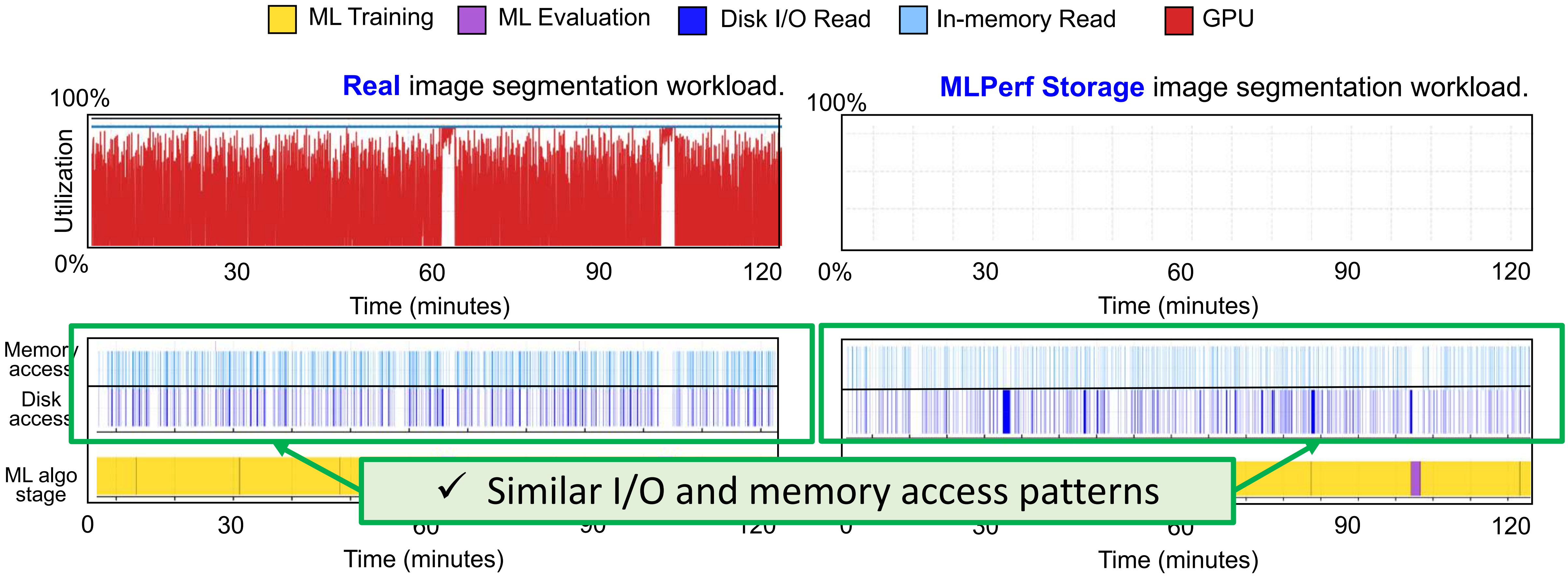
- Pytorch, KiTS19 dataset seed
- Small model, large data.
  - 100s MB per sample
- One sample per file.

Simulating training time does not impact I/O patterns



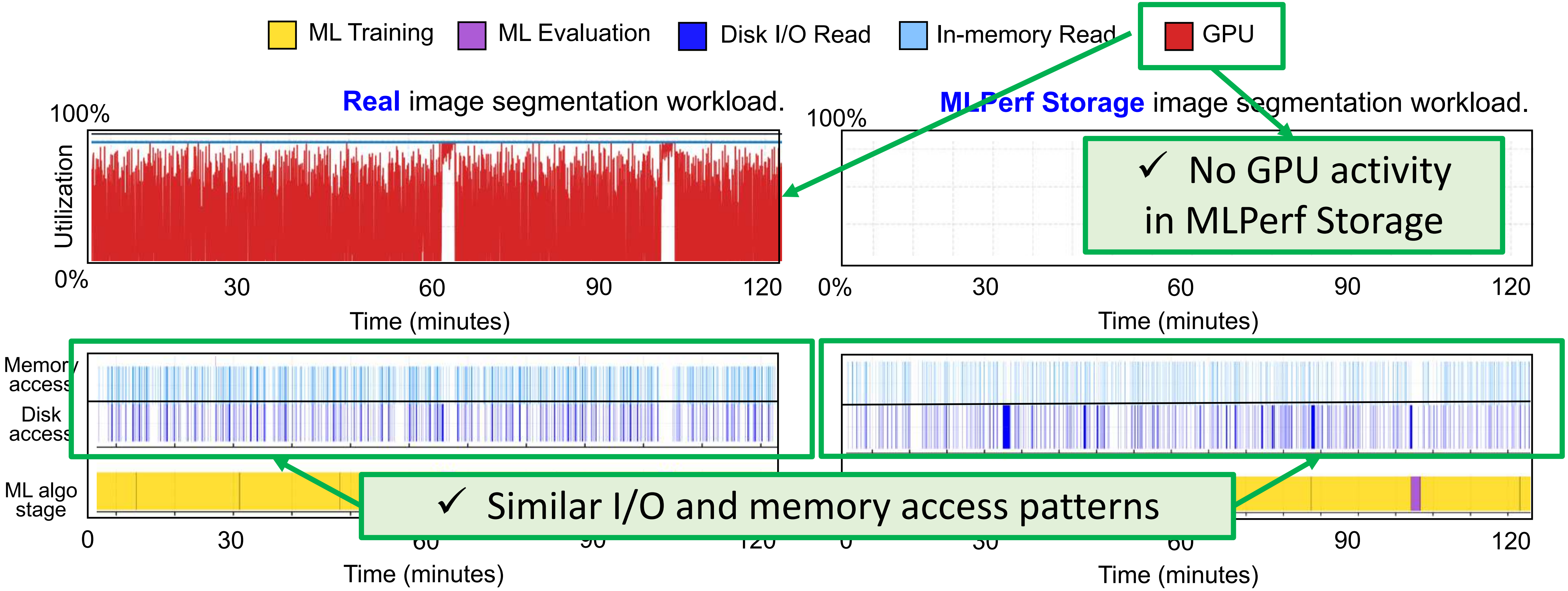
Experiment setup: DGX-1 with 8xV100 GPUs, 512GB DRAM. Dataset : KiTS19, Dataset size:Memory size ratio 2:1

Simulating training time does not impact I/O patterns



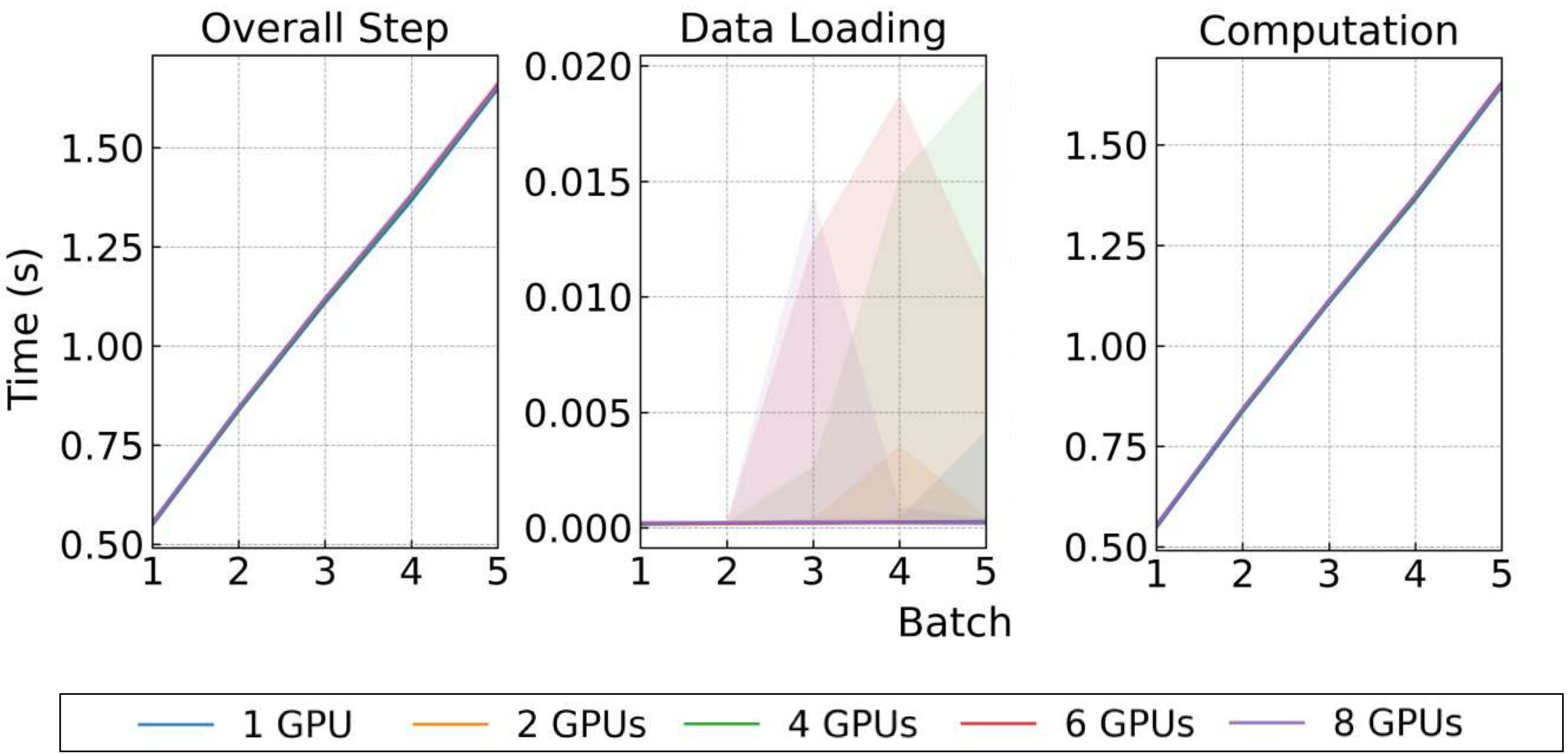
Experiment setup: DGX-1 with 8xV100 GPUs, 512GB DRAM. Dataset : KiTS19, Dataset size:Memory size ratio 2:1

Simulating training time does not impact I/O patterns



Experiment setup: DGX-1 with 8xV100 GPUs, 512GB DRAM. Dataset : KiTS19, Dataset size:Memory size ratio 2:1

3D U-Net: Step Breakdown

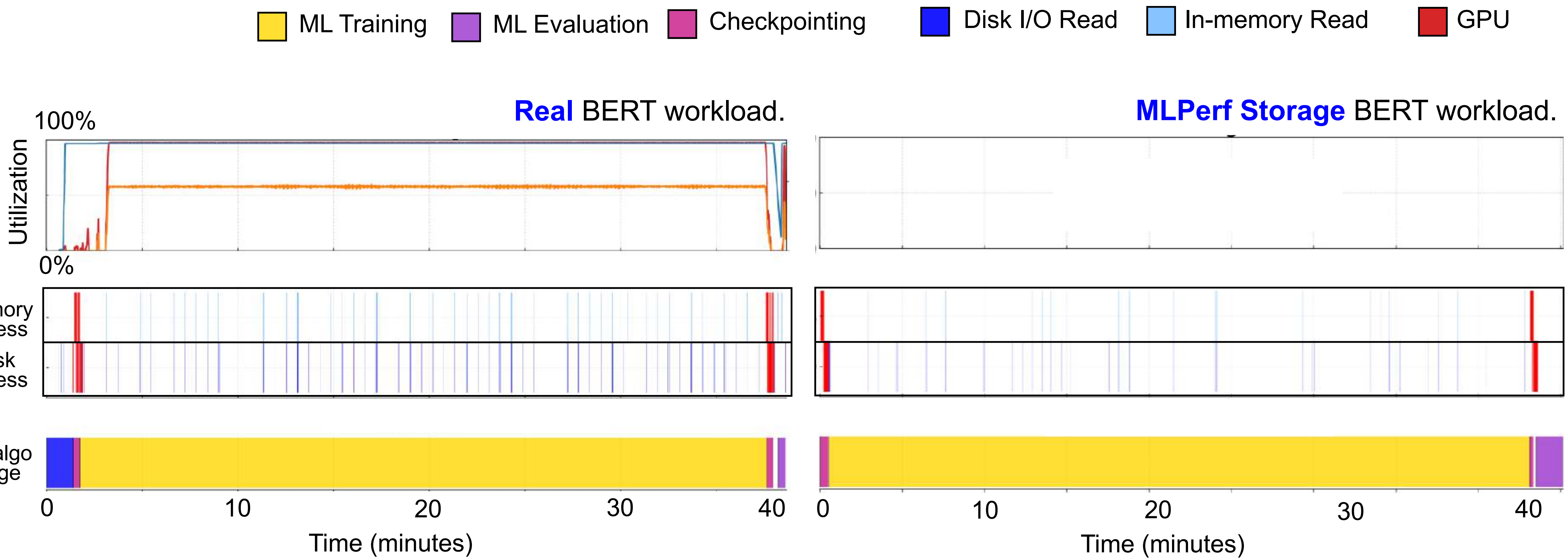


Natural Language Processing: BERT

- Tensorflow implementation, Wikipedia dataset seed
- Large model, small data.
    - Model takes up most of GPU memory
- Many small samples per file
    - ~300K samples per file
- Sequential access inside the files
    - Prefetching

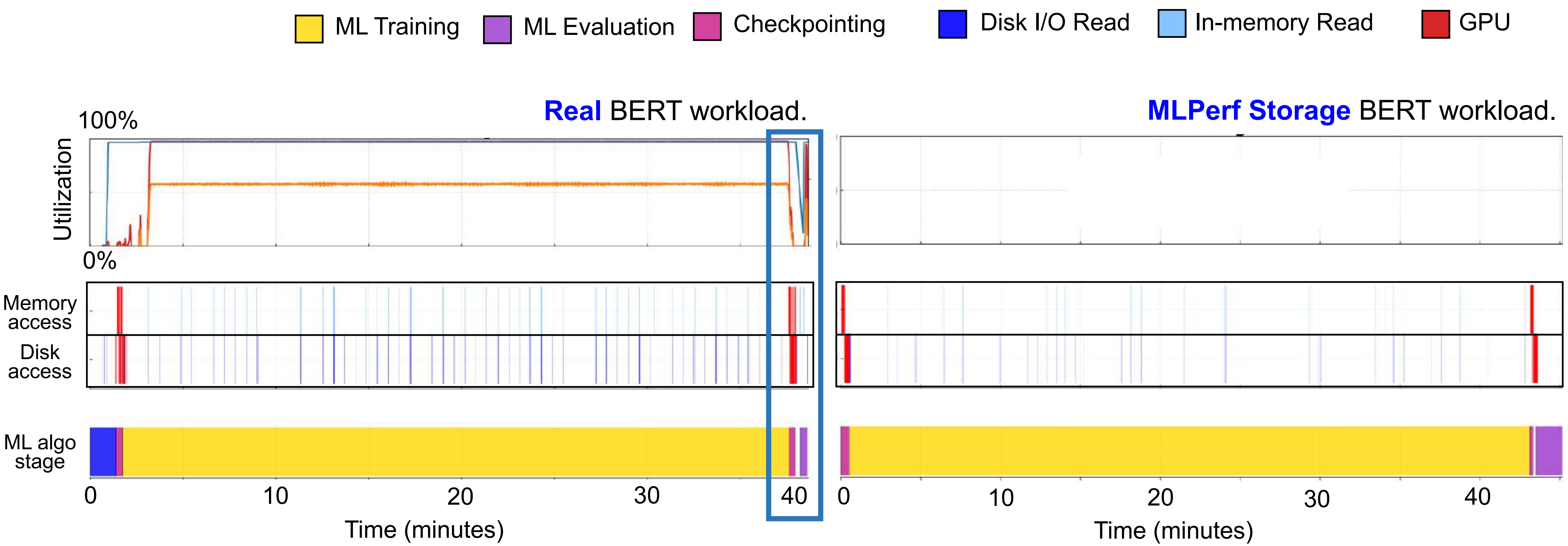
oana.balmau@mcgill.ca

BERT



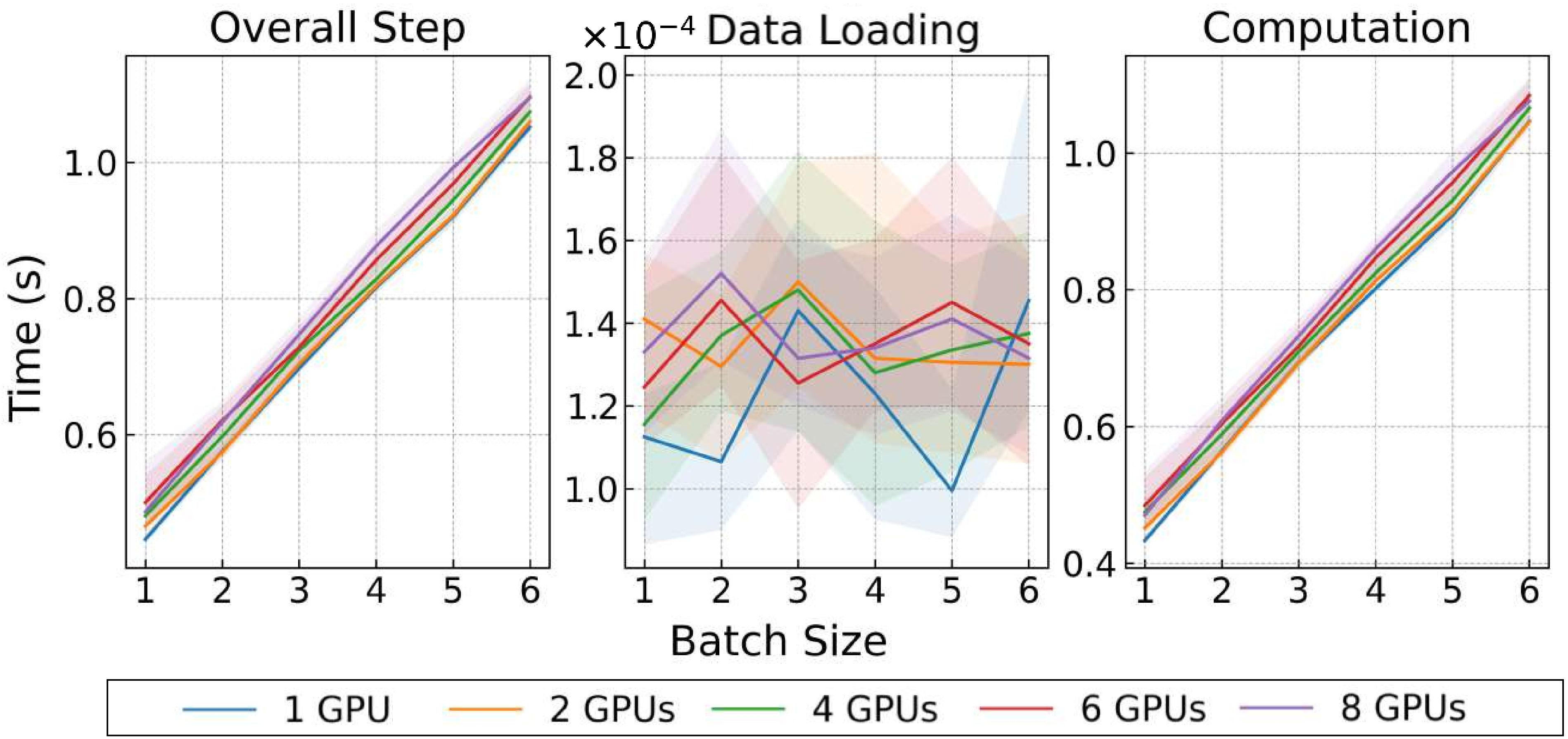
ML Training
ML Evaluation
Checkpointing
Disk I/O Read
In-memory Read
GPU
Real BERT workload.
MLPerf Storage BERT workload.
100%
0%
Utilization
Memory access
Disk access
ML algo stage
0
10
20
30
40
Time (minutes)

BERT



Checkpointing
is a bottleneck
GPU → 0% utilization during checkpoint

BERT: Step Breakdown

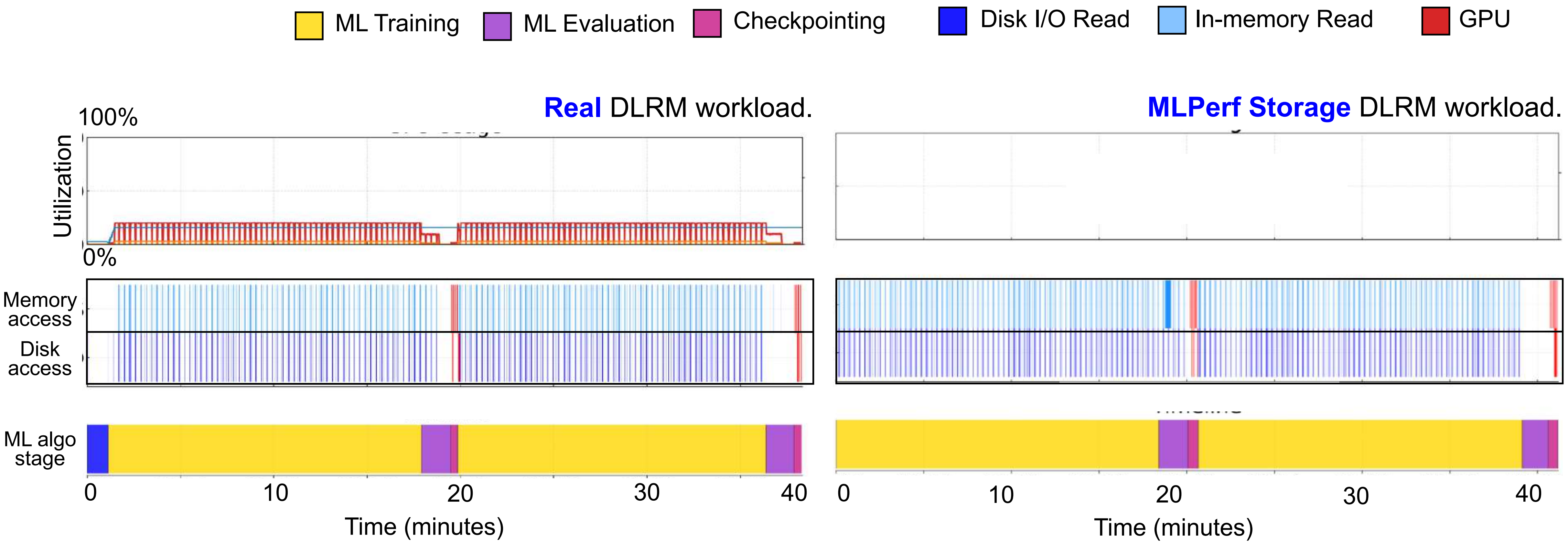


oana.balmau@mcgill.ca

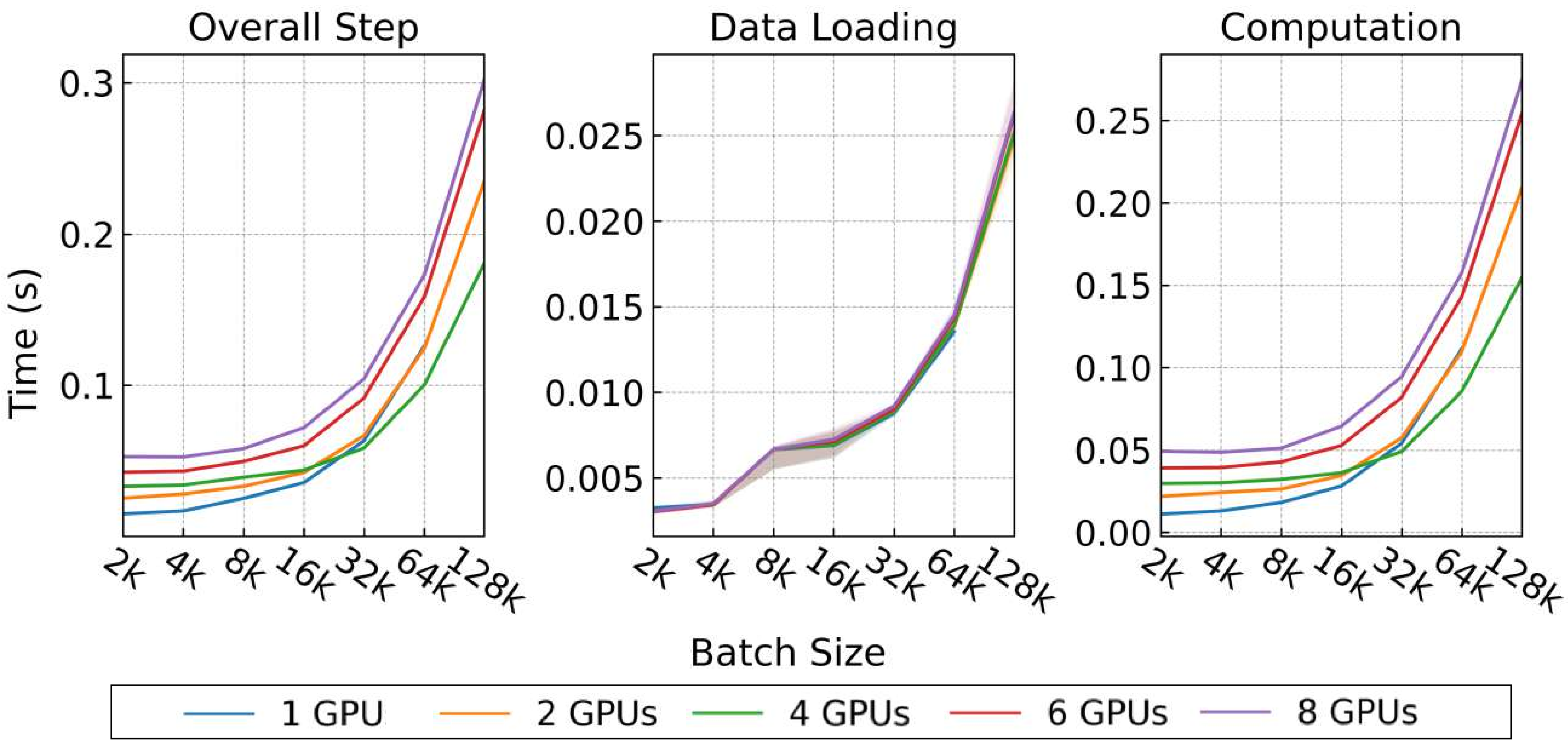
DLRM

- PyTorch implementation, Criteo dataset seed
- Large model, Large data.
  - Model and data parallelism
- Many small random accesses inside a large file

DLRM


ML Training
ML Evaluation
Checkpointing
Disk I/O Read
In-memory Read
GPU
Real DLRM workload.
MLPerf Storage DLRM workload.
100%
Utilization
0%
Memory access
Disk access
ML algo stage
0
10
20
30
40
Time (minutes)

DLRM: Step Breakdown



Lessons learned so far

- Storage on its own is not the main issue.

- Studying I/O patterns revealed opportunities for improvement:
  - In the data loaders (3D U-Net)
  - In the checkpointing (BERT)
  - In the algorithm (DLRM)

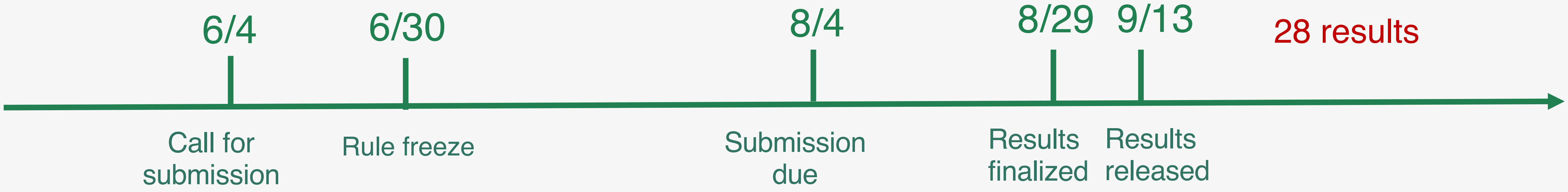
MLPerf Storage v0.5 results overview

Workload selected

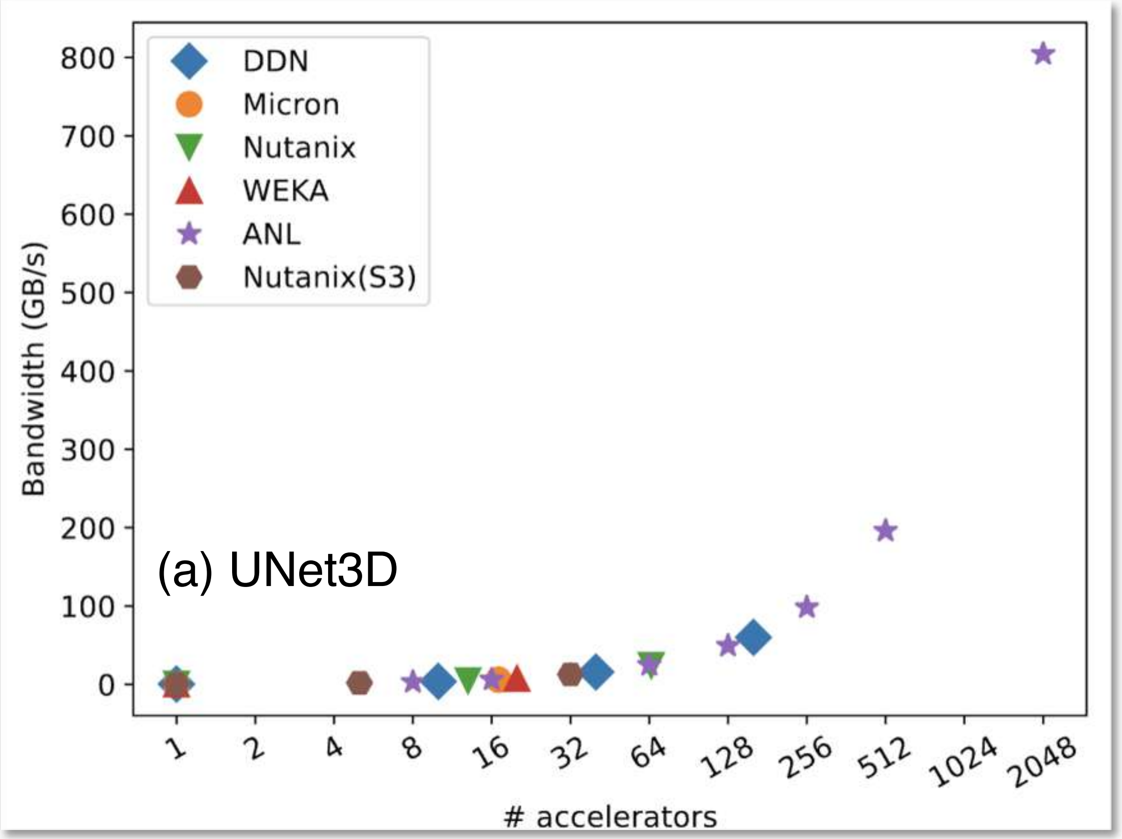
Workload	Image segmentation	Natural language processing
Model	UNet3D	BERT
Seed data	KiTS19 Set of images	Wikipedia 2020 Text
Sample size	~146 MB	~2.5 KB
Framework	Pytorch	Tensorflow
I/O behavior	Randomly select and read a file	Sequential access a subset of files, streamed

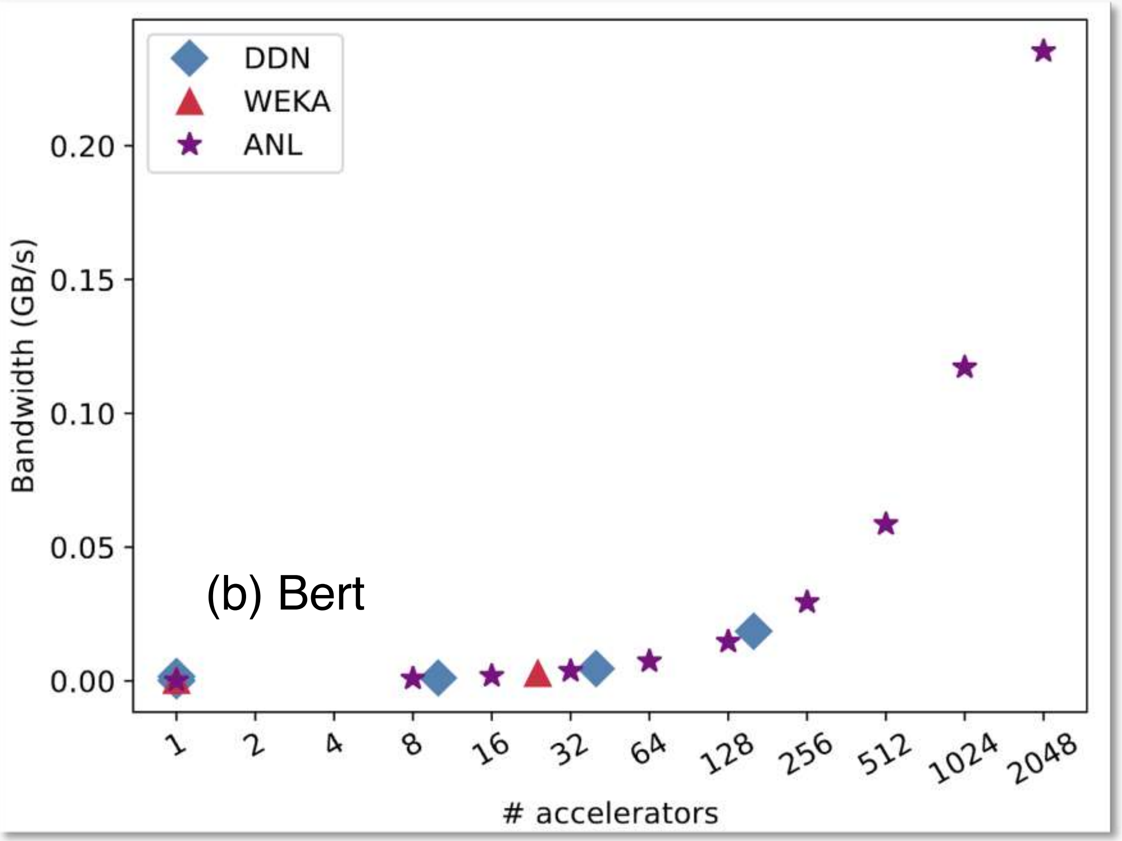
Timeline

5 submitters
28 results



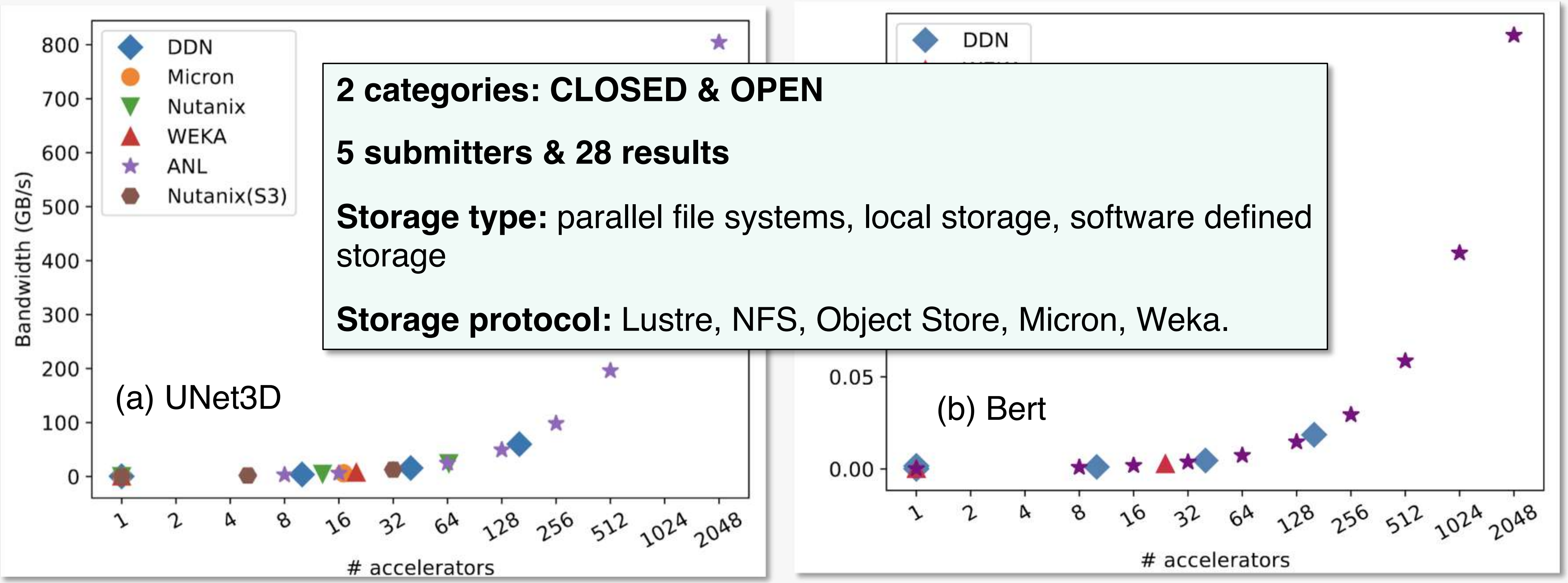
MLPerf Storage v0.5 results overview





Scatter plots of the results from the submitters: (a) UNet3D and (b) Bert. UNet3D is I/O intensive workload and Bert is compute intensive

MLPerf Storage v0.5 results overview



2 categories: CLOSED & OPEN

5 submitters & 28 results

Storage type: parallel file systems, local storage, software defined storage

Storage protocol: Lustre, NFS, Object Store, Micron, Weka.

Scatter plots of the results from the submitters: (a) UNet3D and (b) Bert. UNet3D is I/O intensive workload and Bert is compute intensive

Next Steps in MLPerf Storage

Collect **processing times** for different accelerator types: A100, H100.

Benchmark competition round 2: *https://github.com/mlcommons/storage*

I/O in distributed training

New workloads (LLM, text-to-image, HPC)

Workload collocation

Extend benchmark with **ML pre-processing phase.**

oana.balmau@mcgill.ca

McGill DISCS Lab

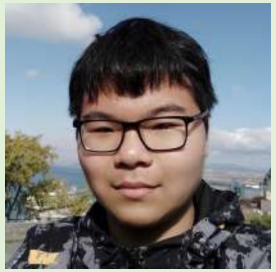
discslab.cs.mcgill.ca
gitlab.cs.mcgill.ca/discs-lab

Postdoctoral Researcher

Dr. Stella Bitchebe

PhD Candidates:

Nelson Bore



Jiaxuan Chen

Shubham Vashisth

Pritish Mishra

Rahma Nouaji

Masters Students

Zachary Doucet

Aayush Kapur

Aidan Goldfarb

Ruoyu Deng

Key Takeaways – MLPerf Storage

MLPerf Storage is a new benchmark

Realistic **storage** settings

No accelerators required to run

Follow MLPerf Storage repository for updates:

https://github.com/mlcommons/storage

Get involved
https://mlcommons.org/working-groups/benchmarks/storage/

Share your thoughts
Email oana.balmau@mcgill.ca

Thanks to all working group co-chairs!

Curtis Anderson
Panasas

Huihuo Zheng
Argonne National Labs

Johnu George,
Nutanix