Analyzing and Mitigating Data Stalls in DNN Training

Jayashree Mohan, Amar Phanishayee, Ashish Raniwala, Vijay Chidambaram

Microsoft Research

TEXAS
The University of Texas at Austin

vmware

Deep Neural Networks (DNNs)

- Widely used for a variety of tasks

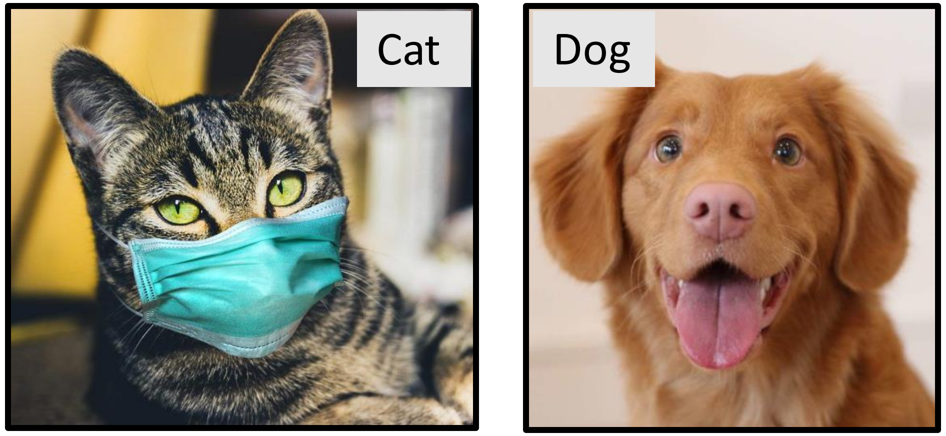
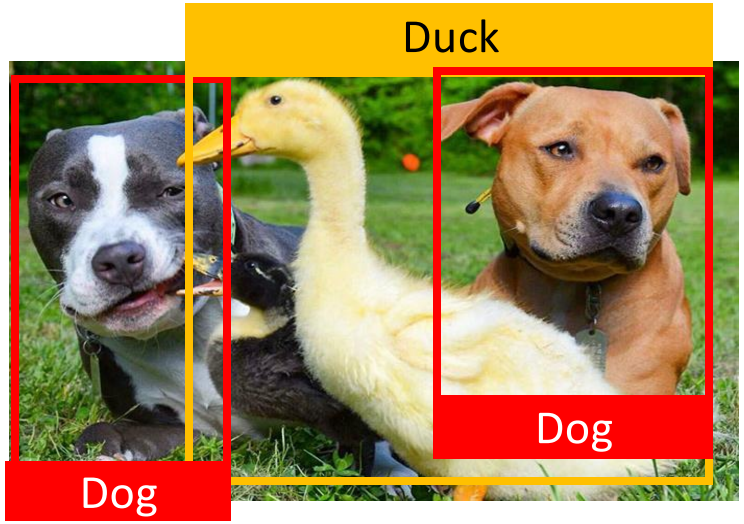
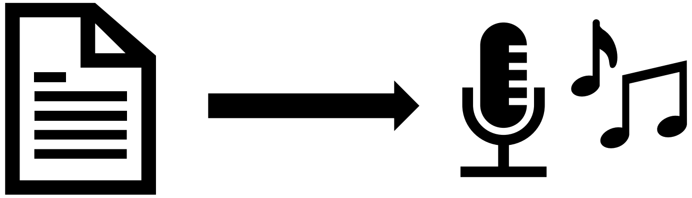


Image Classification



Object detection

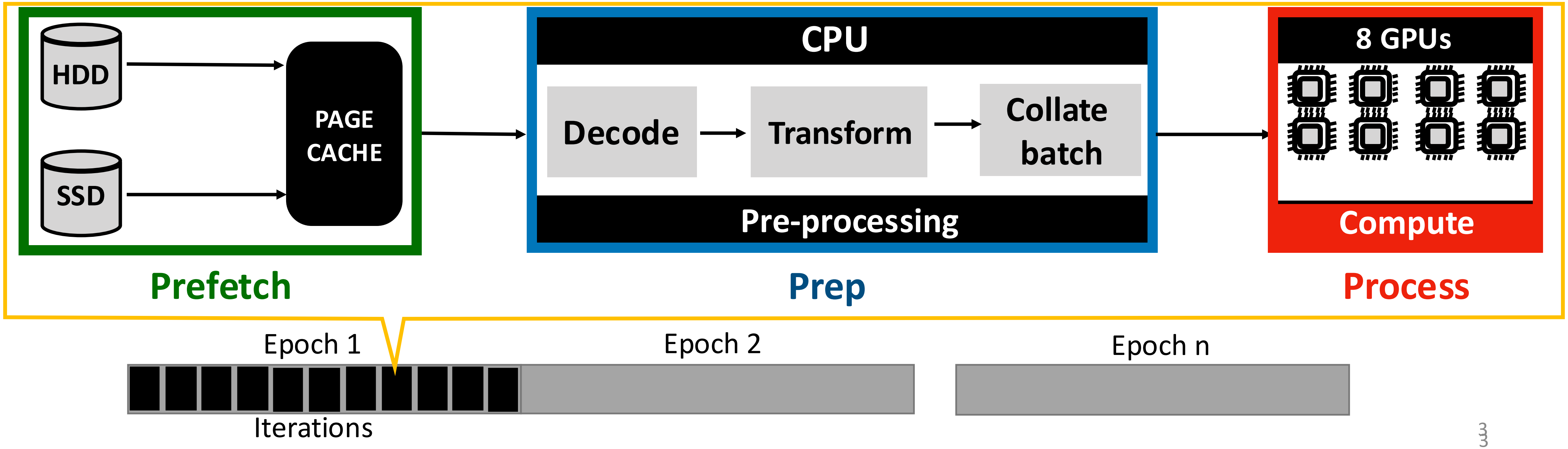
Language Translation



Text To Speech

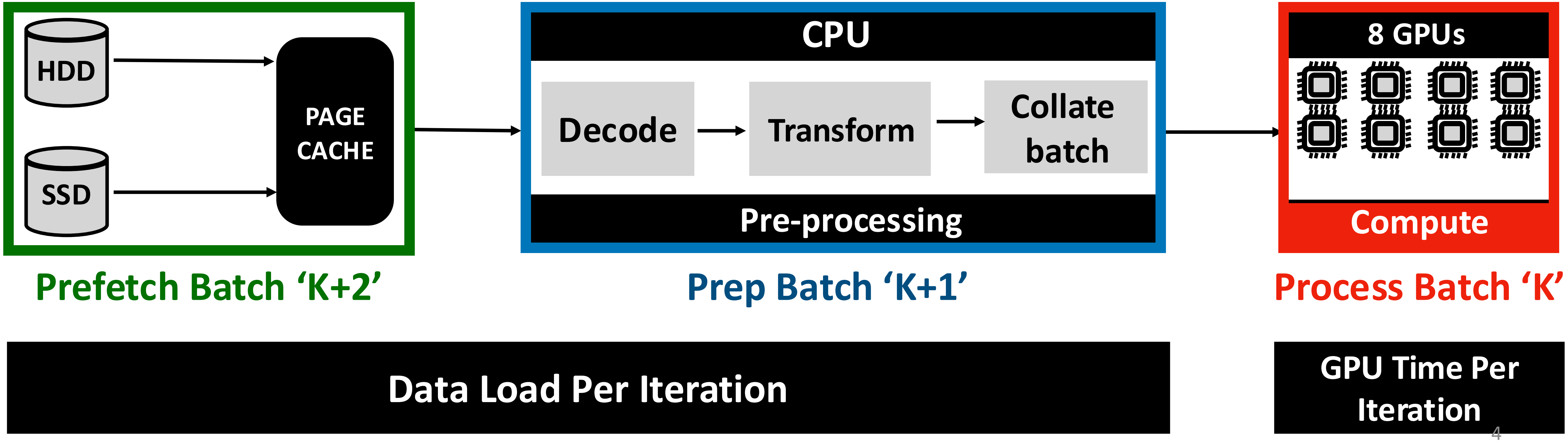
DNN Data Pipeline

- Training happens in epochs
- Each epoch processes the **entire dataset** in a **random order** with **random data augmentations**
- Each epoch is split into iterations (smaller minibatches of data)
- Fetched, pre-processed, and computed upon in a pipelined manner.



DNN Data Pipeline

- Training happens in epochs
- Each epoch processes the **entire dataset** in a **random order** with **random data augmentations**
- Each epoch is split into iterations (smaller minibatches of data)
- Fetched, pre-processed, and computed upon in a pipelined manner.



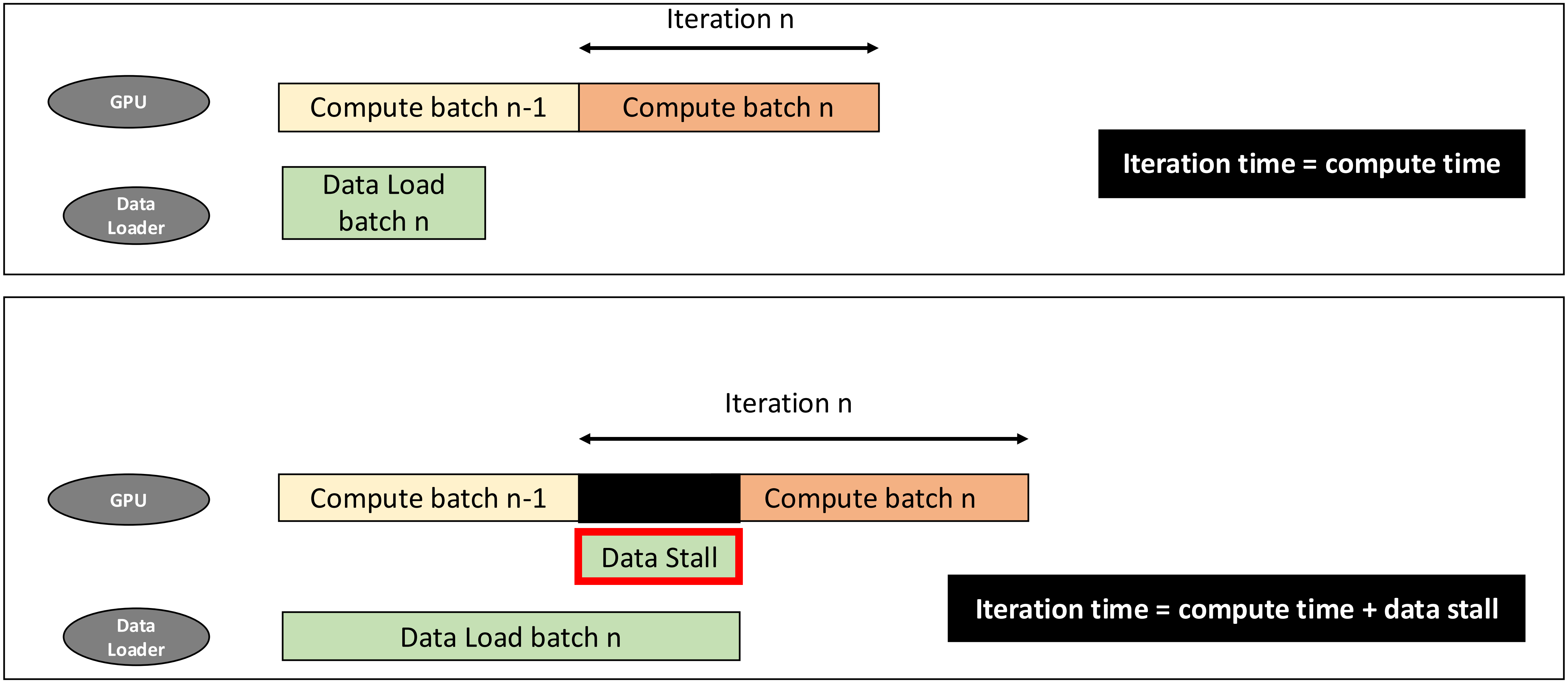
Analyzing and Mitigating Data Stalls

Analyze the impact of the ingest pipeline (**storage, memory and CPU**) on DNN training in a variety of training scenarios and propose solutions to mitigate data stalls

Outline

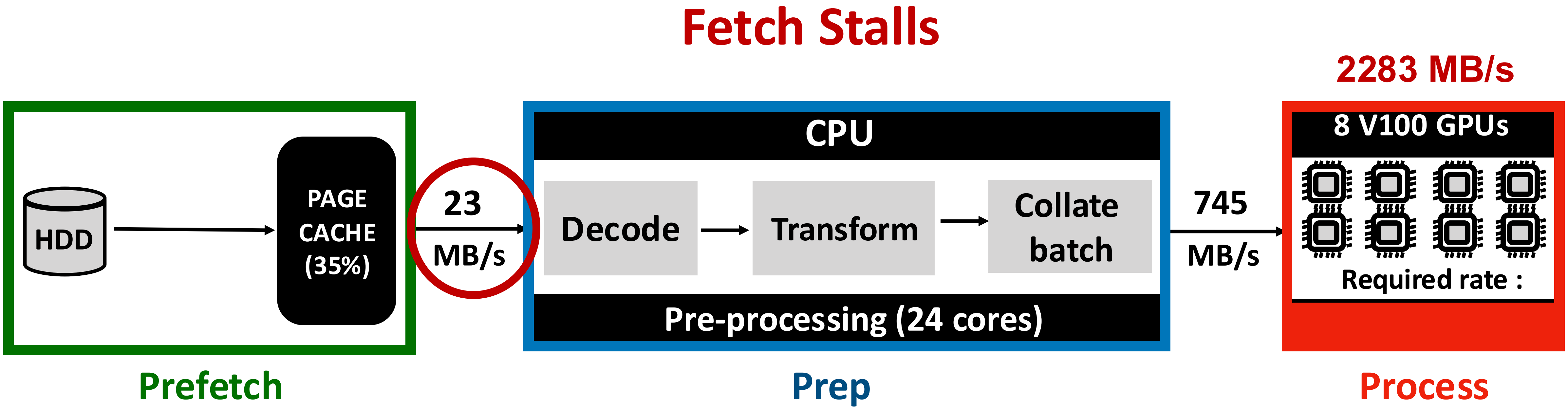
- **Data Stalls**
- Analyzing Data Stalls
- CoorDL : Mitigating Data Stalls
- Evaluation

Data Stall



Iteration n
GPU
Compute batch n-1
Compute batch n
Data Loader
Data Load batch n
Iteration time = compute time
Iteration n
GPU
Compute batch n-1
Compute batch n
Data Stall
Data Loader
Data Load batch n
Iteration time = compute time + data stall

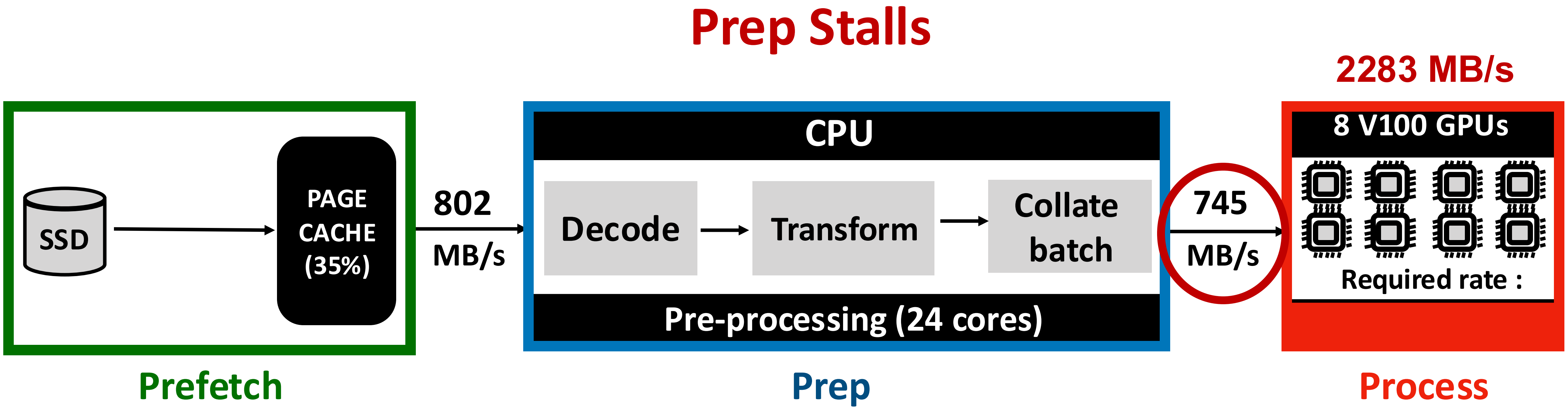
Fetch Stalls



Training pipeline is stalled on **data fetch**

Training is **I/O bound**

Prep Stalls



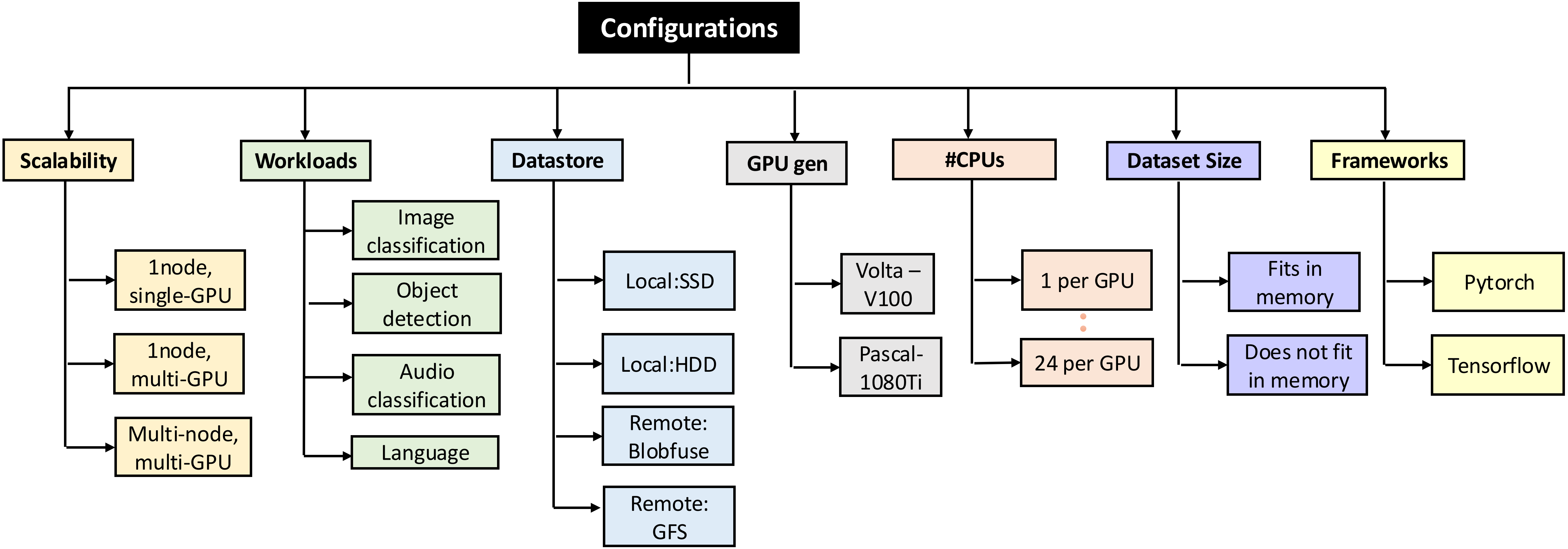
Training pipeline is stalled on **data prep**

Training is **CPU bound**

Outline

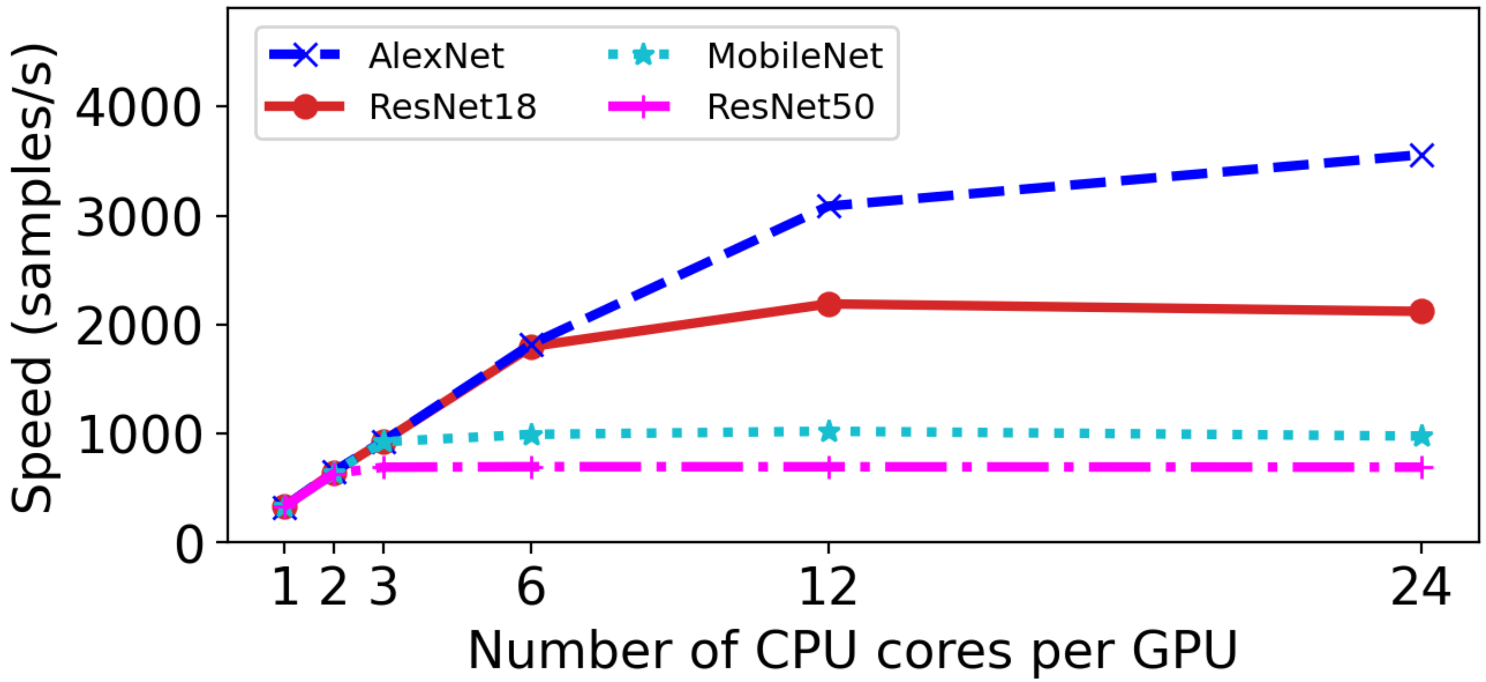
- Data Stalls
- **Analyzing Data Stalls**
- CoorDL : Mitigating Data Stalls
- Evaluation

Analyzing data stalls



Data Stall Analysis

1. DNNs need anywhere between 3 – 24 CPU cores **per GPU** for data pre-processing

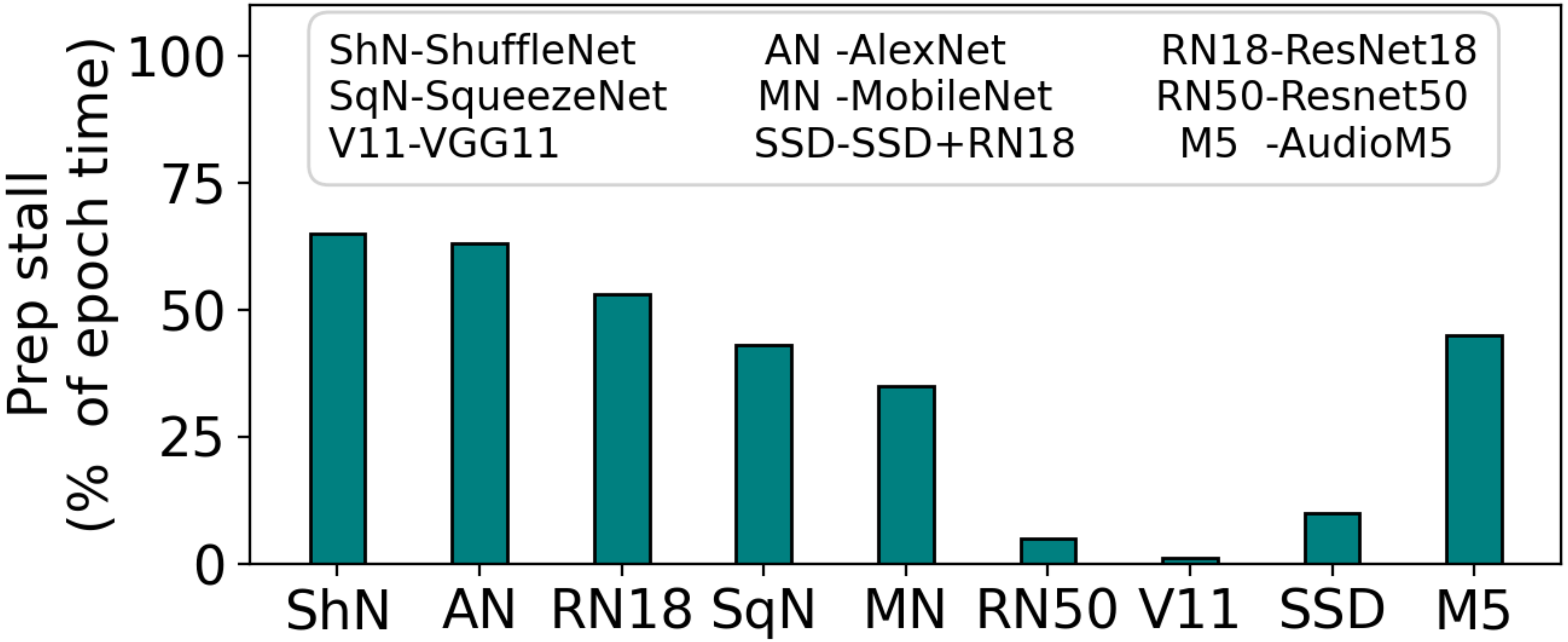


Setup

- V100 GPU
- 100% Cached
- 1GPU Training
- Vary #CPU cores

Data Stall Analysis

1. DNNs need anywhere between 3 – 24 CPU cores **per GPU** for data pre-processing

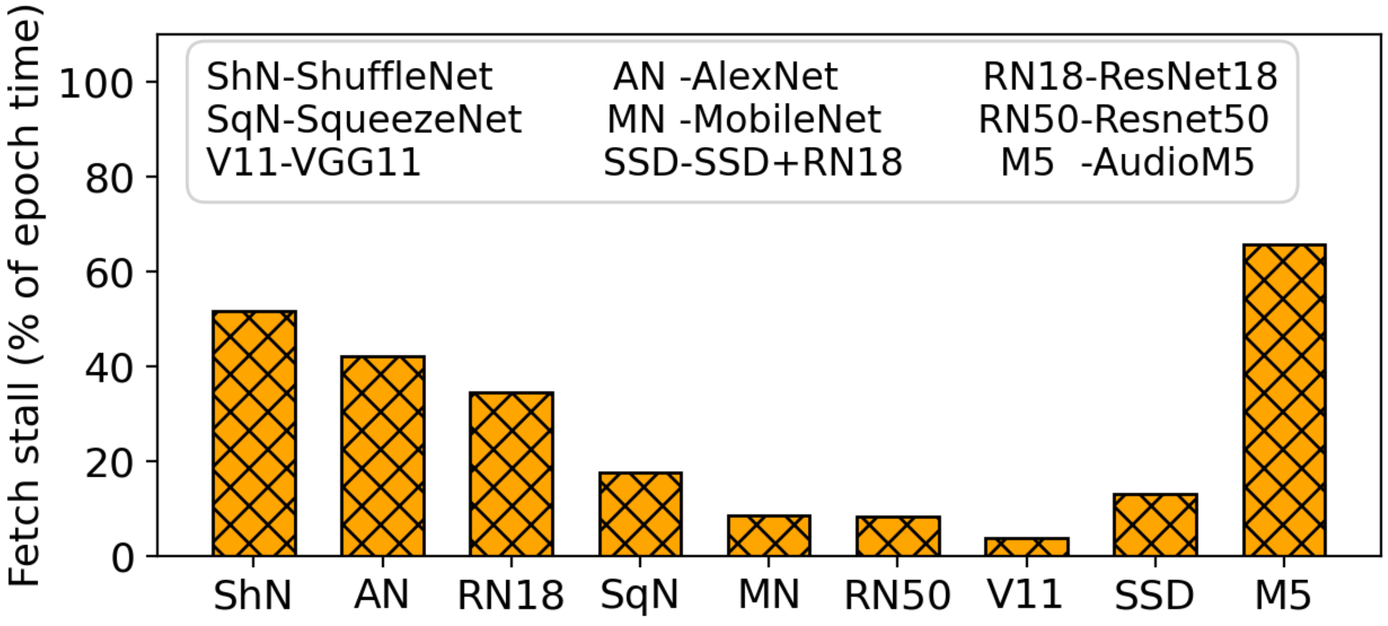


Setup

- V100 GPU
- 100% Cached
- 8GPU Training
- Use all CPU cores

Data Stall Analysis

1. DNNs need anywhere between 3 – 24 CPU cores **per GPU** for data pre-processing
2. Fetch stalls exist across models with large datasets
   - OS Page Cache is inefficient for DNN training due to thrashing



Setup

- V100 GPU
- 35% Cached
- 8GPU Training

Data Stall Analysis

1. DNNs need anywhere between 3 – 24 CPU cores **per GPU** for data pre-processing
2. Fetch stalls exist across models with large datasets
   - OS Page Cache is inefficient for DNN training due to thrashing
3. Redundancy in data fetch and pre-processing

Outline

- Data Stalls
- Analyzing Data Stalls
- **CoorDL : Mitigating Data Stalls**
- Evaluation

CoorDL: Insights

Finding	Insight
OS Page Cache is inefficient for DNN training due to thrashing	Optimize DNN cache to eliminate thrashing across epochs **(MinIO Cache)**
Redundant data fetch in distributed training	Local caches of servers can be coordinated to fetch data from the remote cache to overcome storage I/O bottlenecks **(Partitioned Caching)**
Redundant data fetch and prep in HP search	HP search jobs must coordinate data fetch & prep **(Coordinated Prep)**

CoorDL: Insights

Finding	Insight
OS Page Cache is inefficient for DNN training due to thrashing	Optimize DNN cache to eliminate thrashing across epochs **(MinIO Cache)**
Redundant data fetch in distributed training	Local caches of servers can be coordinated to fetch data from the remote cache to overcome storage I/O bottlenecks **(Partitioned Caching)**
Redundant data fetch and prep in HP search	HP search jobs must coordinate data fetch & prep **(Coordinated Prep)**

OS Page Cache is ineffective across epochs!

- Uses OS Page cache to cache the prefetched data items for subsequent epochs
- Unaware of DNN access pattern
- ResNet18 on OpenImages Dataset (Server – 8V100 GPUs, 500GB DRAM)

Dataset size	**645GB**
Cache size	65% (420GB)
Expected disk access (stable state)	**225GB (35%)**

OS Page Cache is ineffective across epochs!

- ResNet18 on OpenImages Dataset (Server – 8V100 GPUs, 500GB DRAM)

Dataset size	**645GB**
Cache size	65% (420GB)
Expected disk access (stable state)	**225GB (35%)**
DALI-Seq	**422GB (87%)**
DALI-Shuffle	**340GB (53%)**

Increased disk access makes training I/O bound => Fetch stalls

OS Page Cache is ineffective across epochs!

Across epochs, the items in OS Page Cache are not used effectively!

- Prefetched items replace existing, unused items in Page cache (LRU)
- These evicted items are prefetched from storage later in the epoch

- Models like ShuffleNet spend **40% of epoch time in blocking I/O**

MinIO cache

- Given a cache capacity, fill it up with random data items when first accessed
- Once cache is full, unlike traditional caching, there is **no cache replacement**
- Disk accesses per epoch = capacity misses

Outline

- Data Stalls
- Analyzing Data Stalls
- CoorDL : Mitigating Data Stalls
- **Evaluation**
    - **Setup**
    - Single-Node Training
    - Multi-Node Training
    - Hyperparameter Search

Evaluation : Setup

PyTorch
DALI Data Loading Pipeline

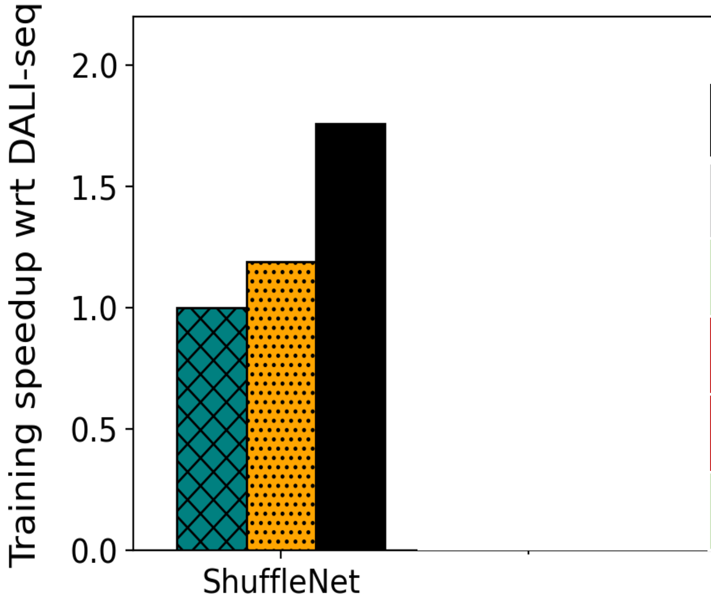
Task	Model	Dataset (Size)
Image Classification	AlexNet ShuffleNetv2 ResNet18 SqueezeNet MobileNetv2 ResNet50 VGG11	ImageNet-1K (146GB) Imagenet-22K (1.3TB) OpenImages-Extended (645GB)
Object Detection	SSD + ResNet18	OpenImages (561GB)
Audio Classification	M5	Free Music Arxiv (950GB)

Servers	GPU Config	GPU Mem (GB)	Storage Media	Rand Read (MBps)	DRAM (GB)	CPU cores
SSD-V100	8 x V100	32	SSD	530	500	24
HDD-1080Ti	8 x 1080Ti	11	HDD	15-100	500	24

Outline

- Data Stalls
- Analyzing Data Stalls
- CoorDL : Mitigating Data Stalls
- **Evaluation**
  - Setup
  - **Single-Node Training**
  - Multi-Node Training
  - Hyperparameter Search

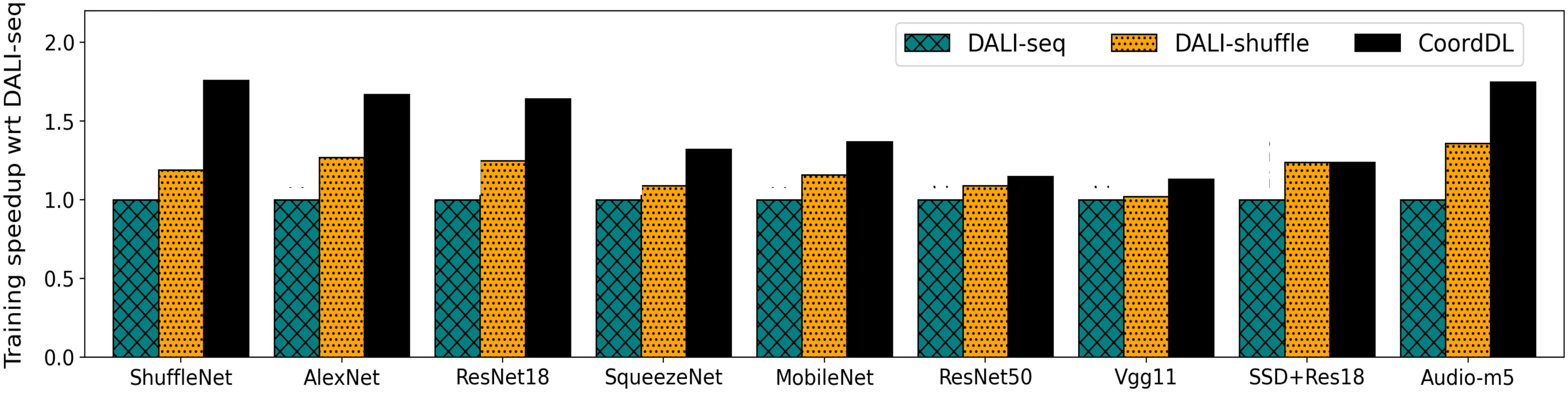
1. Single-server training



Dataset size	645GB
Cache size	65% (420GB)
Expected disk access	**225GB (35%)**
DALI-Seq	**422GB (87%)**
DALI-Shuffle	**340GB (53%)**
CoorDL	**225GB (35%)**

Upto 1.8x faster training on SSD-V100 over DALI by reducing cache misses (minIO)

1. Single-server training

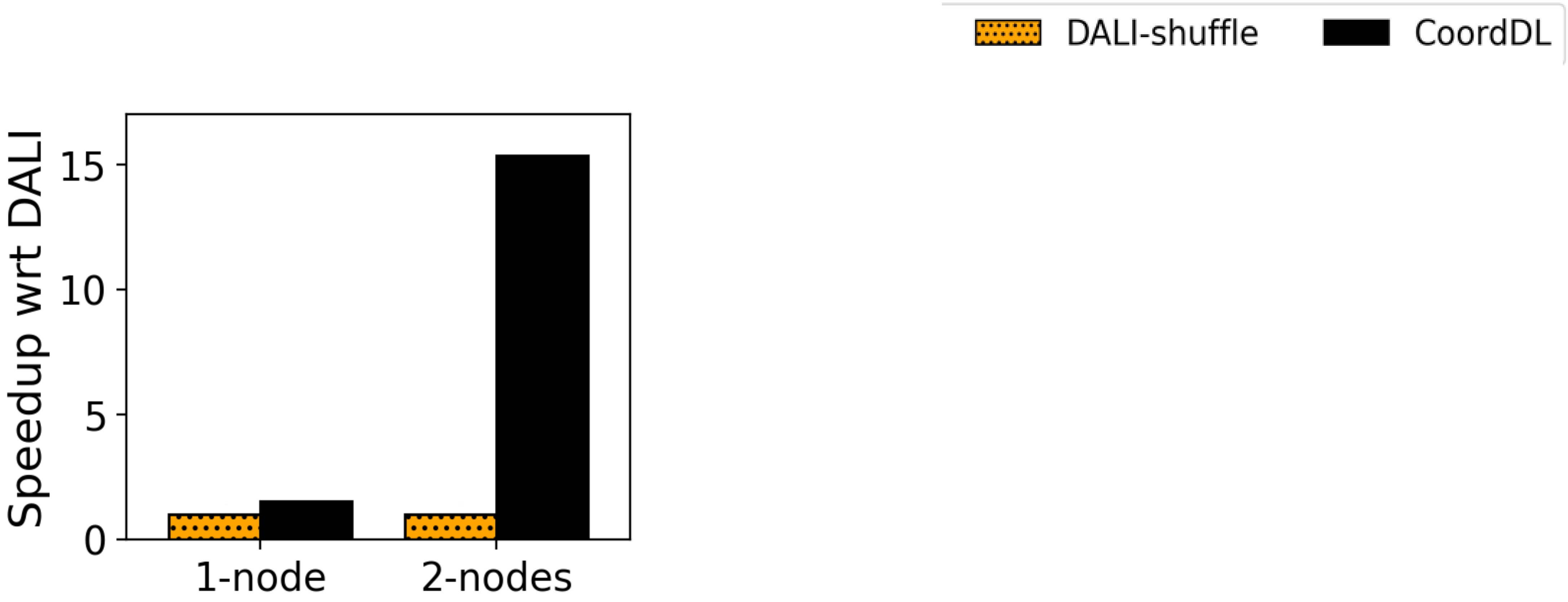


Upto 1.8x faster training on SSD-V100 over DALI by reducing cache misses (minIO)

Outline

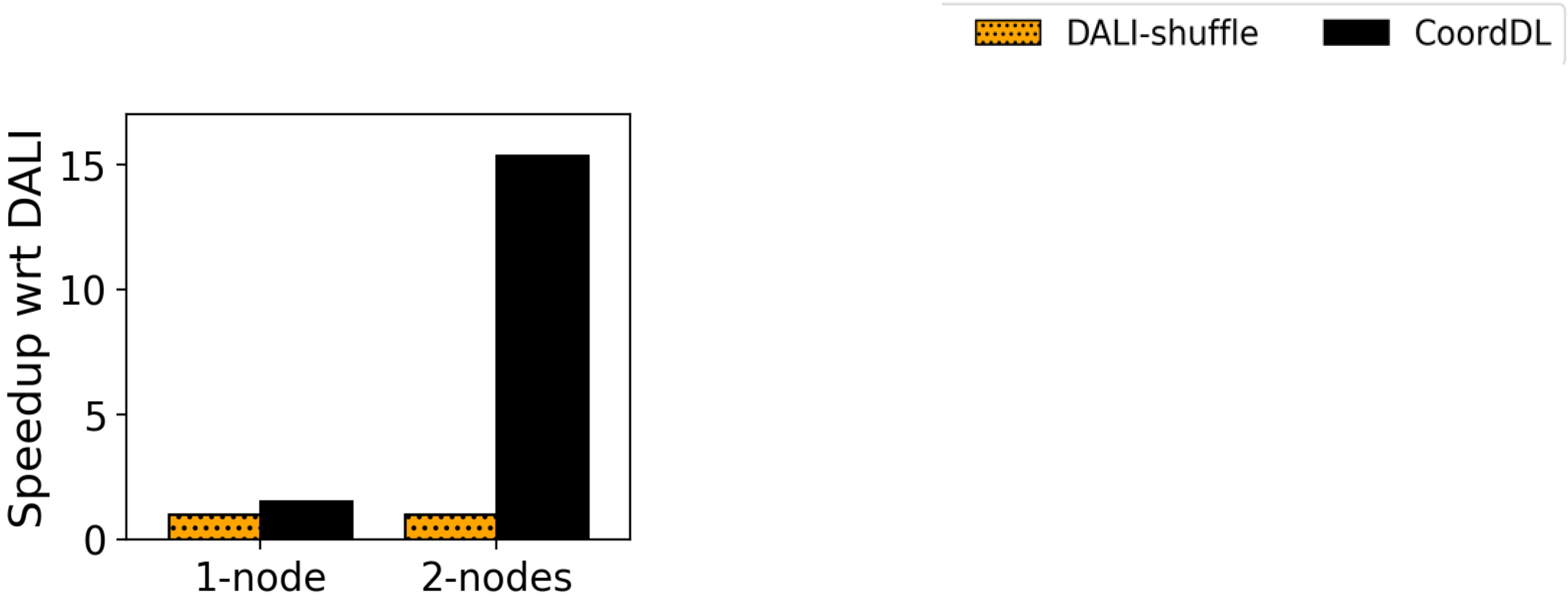
- Data Stalls
- Analyzing Data Stalls
- CoorDL : Mitigating Data Stalls
- **Evaluation**
    - Setup
    - Single-Node Training
    - **Multi-Node Training**
    - Hyperparameter Search

2. Multi-server training



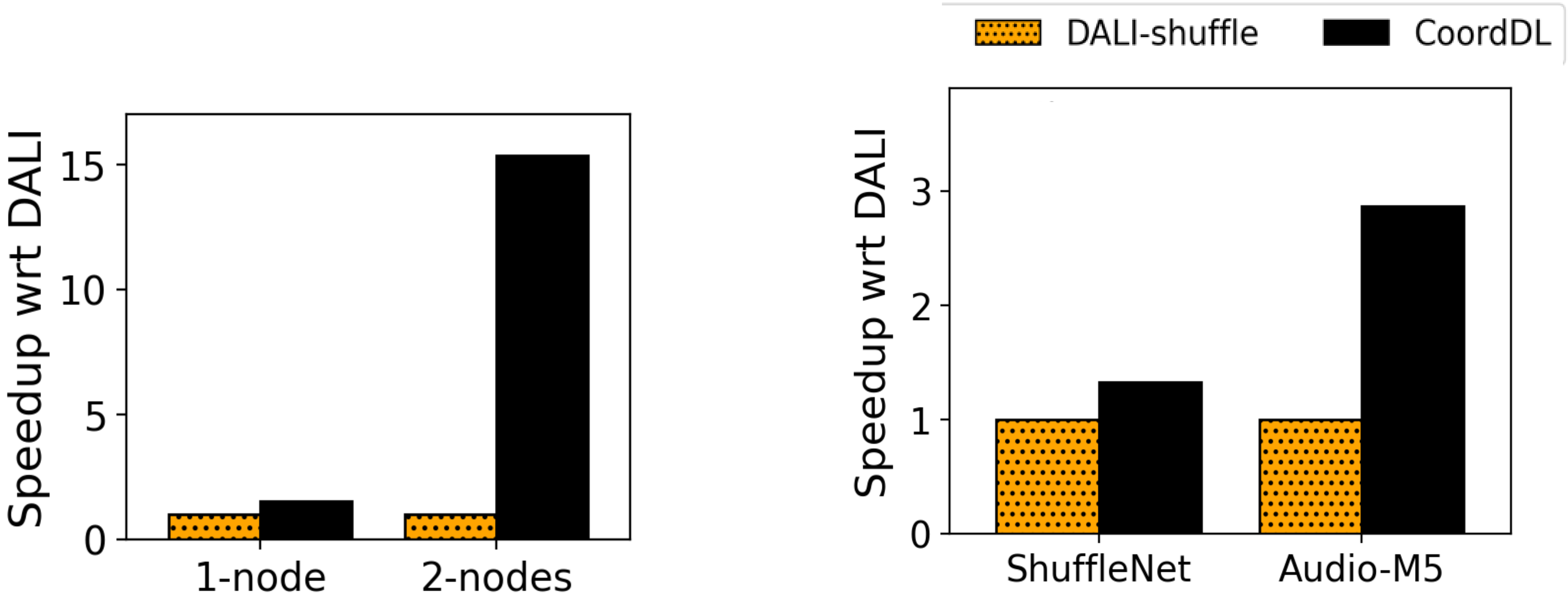
Alexnet – HDD-1080Ti

2. Multi-server training



minIO + Partitioned caching minimizes disk IO and accelerates training by upto 15x

2. Multi-server training

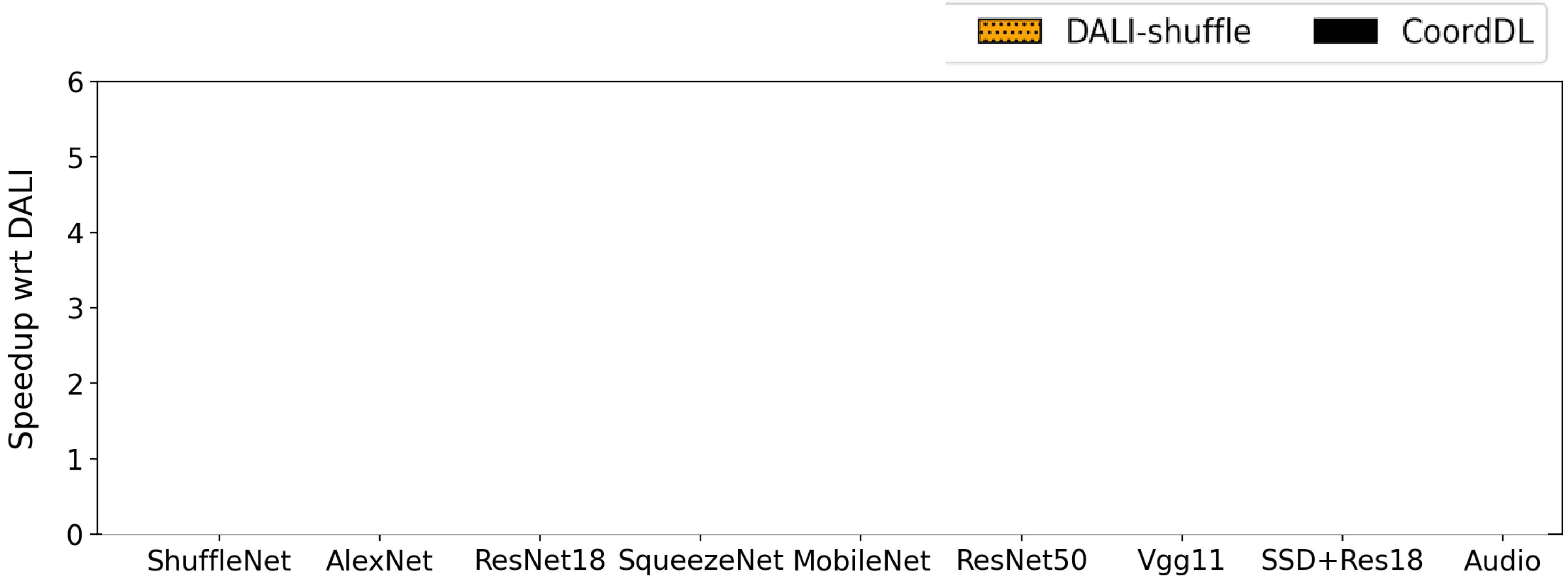


minIO + Partitioned caching minimizes disk IO and accelerates training by upto 15x on HDD and upto 3x on SSD

Outline

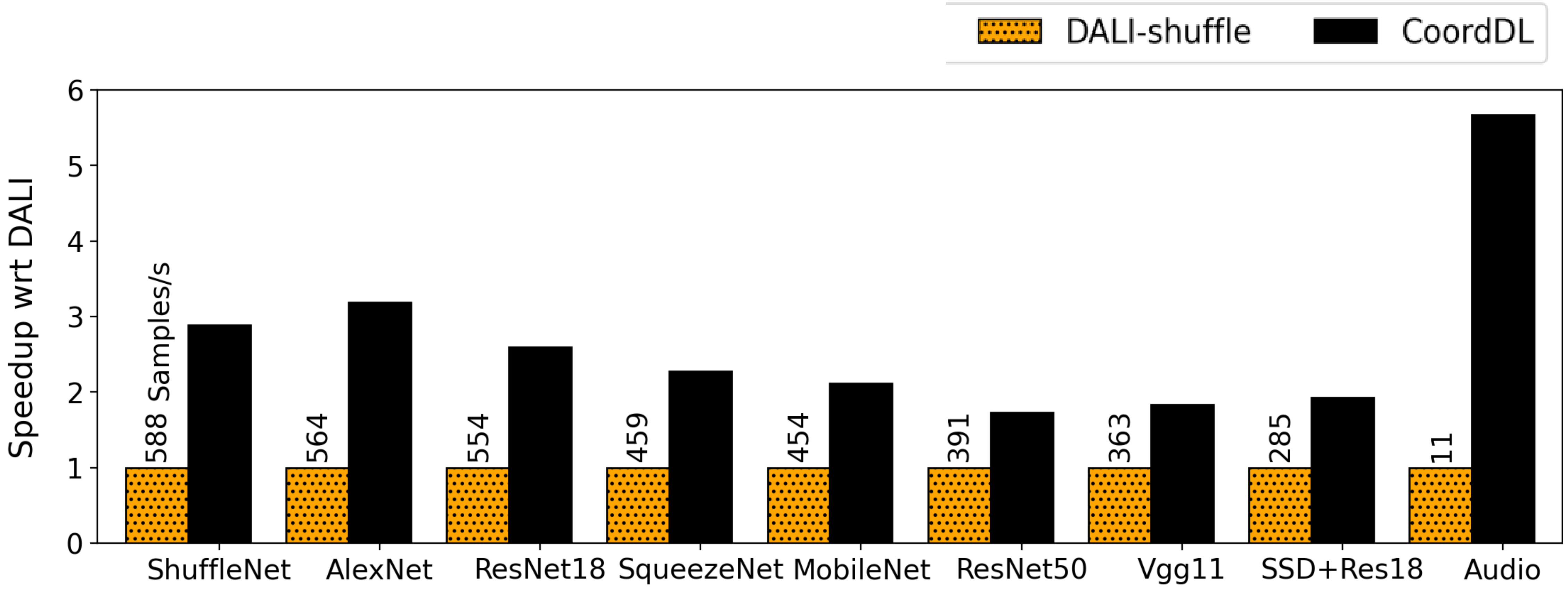
- Data Stalls
- Analyzing Data Stalls
- CoorDL : Mitigating Data Stalls
- **Evaluation**
  - Setup
  - Single-Node Training
  - Multi-Node Training
  - **Hyperparameter Search**

3. HP search



DALI-shuffle
CoordDL
Speedup wrt DALI
0
1
2
3
4
5
6
ShuffleNet
AlexNet
ResNet18
SqueezeNet
MobileNet
ResNet50
Vgg11
SSD+Res18
Audio

3. HP search



Coordinated prep is able to speed up training by upto 5.5x by eliminating redundant pre-processing and disk IO.

Summary

- Data stalls exist in DNN training on commodity servers
  - Squander away benefits from fast GPUs
- Analyzed causes for data stalls
- Built CoorDL to mitigate I/O and CPU bottlenecks in some scenarios

Analysis

Data Stalls

Mitigation using CoorDL

DS-Analyzer

- Precisely measure data stalls
- Predictive what-if analysis

Challenges

- Decoded data cache
- Cost-performance tradeoff

Thank you!

Source code : https://github.com/msr-fiddle/DS-Analyzer

Contact : jaya@cs.utexas.edu