HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Accelerating Deep Learning Training Through Transparent Storage Tiering

Marco Dantas, Diogo Leitão, Peter Cui[1], Ricardo Macedo, Xinlian Liu[2], Weijia Xu[1], **João Paulo**

INESC TEC & University of Minho [1] University of Texas at Austin [2] Hood College

IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing

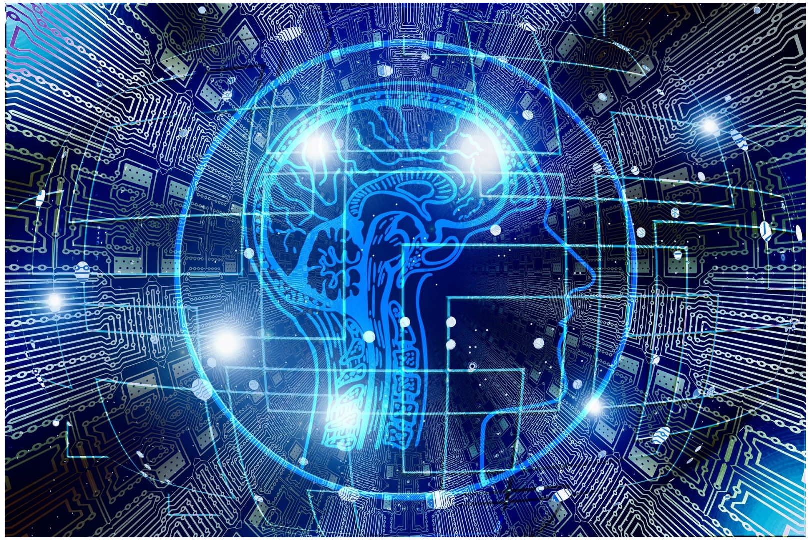
May 17, 2022

INESCTEC
TECNOLOGIA E CIÊNCIA
LABORATÓRIO ASSOCIADO

Universidade do Minho

DL and HPC Convergence

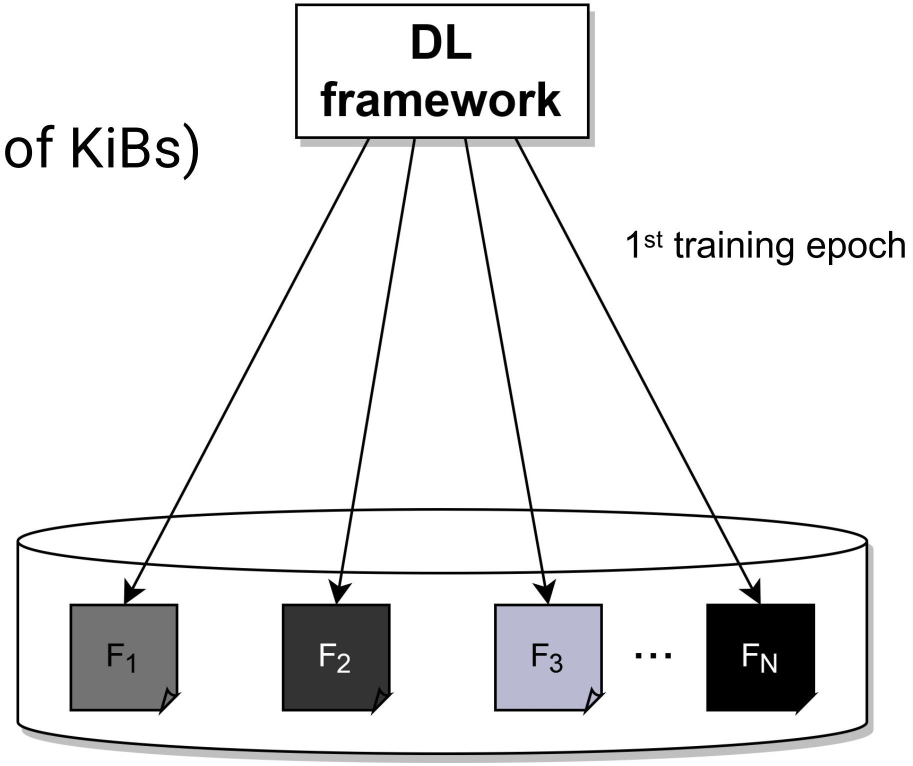
- Deep Learning (DL)
    - New models and predictions for
        - Healthcare, finance, natural sciences, …
    - Computational demanding workloads
    - Large datasets



- DL workloads can leverage the **computational** power offered by HPC!
- Is the same true for HPC’s **storage** resources?

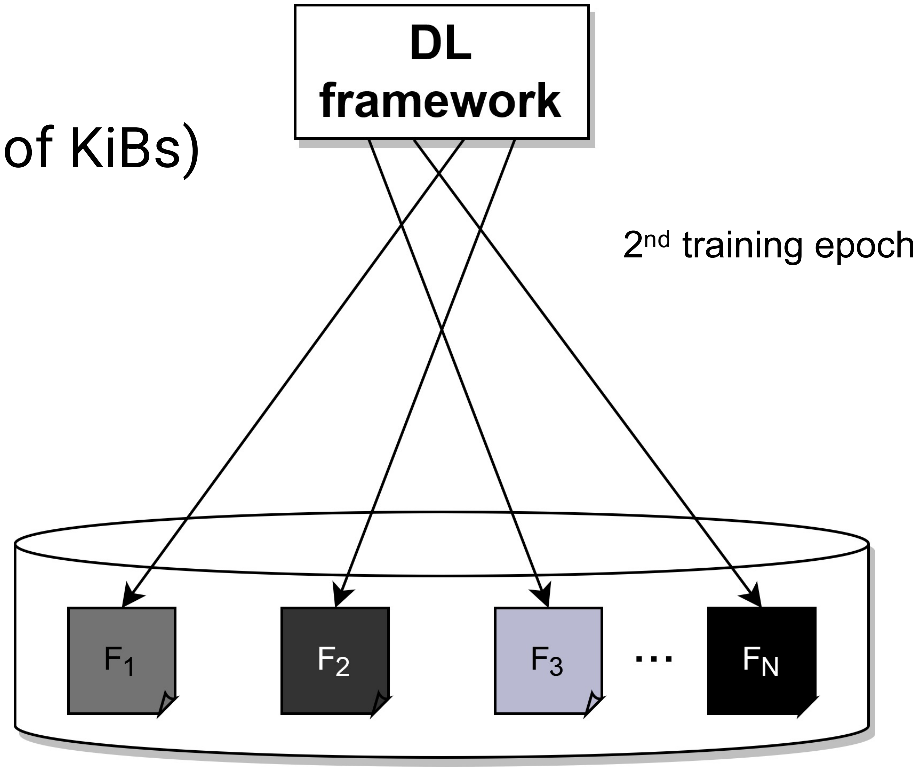
DL Model Training

- From the Storage I/O perspective
  - Datasets composed by **millions** of small files (order of KiBs)
    - Optimized data formats (order of MiBs)
      e.g., TFRecord
  - Read-oriented workload
  - Trained model's accuracy
    - **Epochs:** Full dataset is read at each training epoch



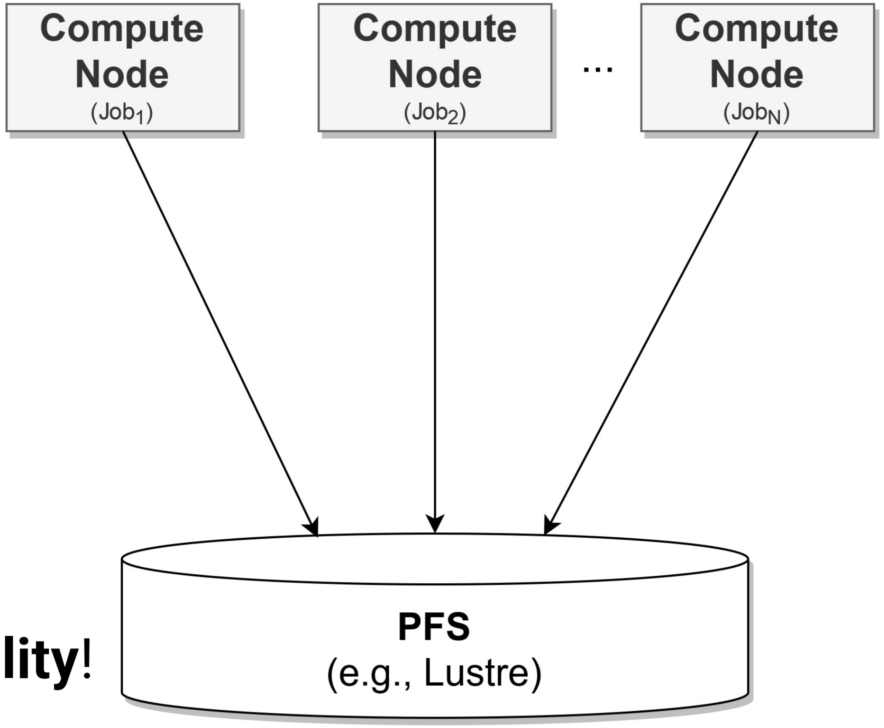
DL Model Training

- From the Storage I/O perspective
  - Datasets composed by **millions** of small files (order of KiBs)
    - Optimized data formats (order of MiBs)
      e.g., TFRecord
  - Read-oriented workload
  - Trained model's accuracy
    - **Epochs:** Full dataset is read at each training epoch
    - **Shuffling**: Random I/O accesses across epochs



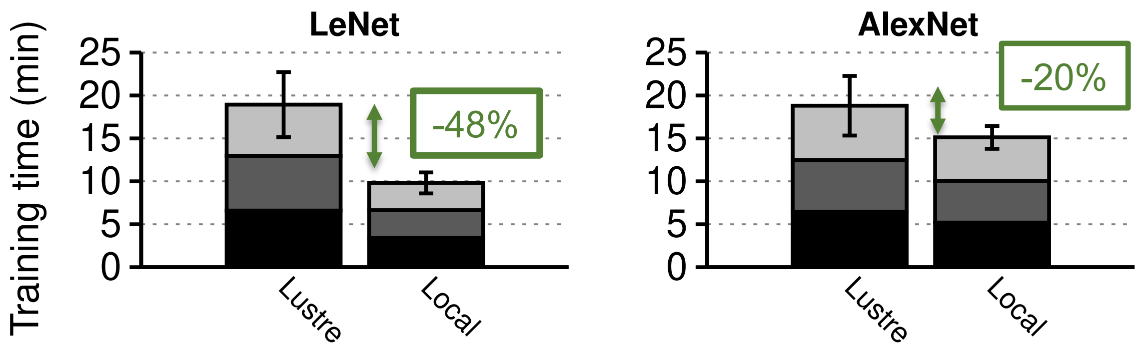
The Storage Bottleneck Problem

- ”Bad” data-centric I/O workloads
  - **Metadata-intensive** due to small files
  - **Hard to cache** due to random accesses
- Parallel File System (PFS)
  - Competition for shared storage resources
  - Can lead to **performance variability** or even **unavailability**!

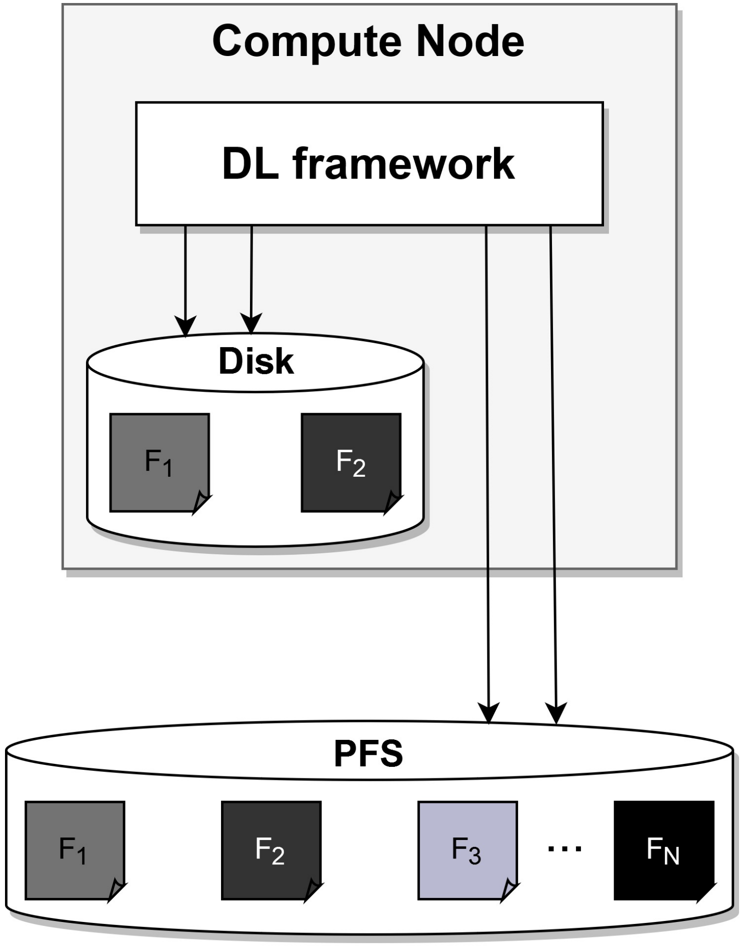


Solution: Use Local Storage?

- Caching data at compute node's local storage
  - **Reduces I/O pressure** at the PFS
  - **Improves DL training speed** for I/O-bound workloads



Average training time (3 epochs) for LeNet and AlexNet models with the ImageNet dataset (100 GiB) being read from **Lustre** and **Local** disk



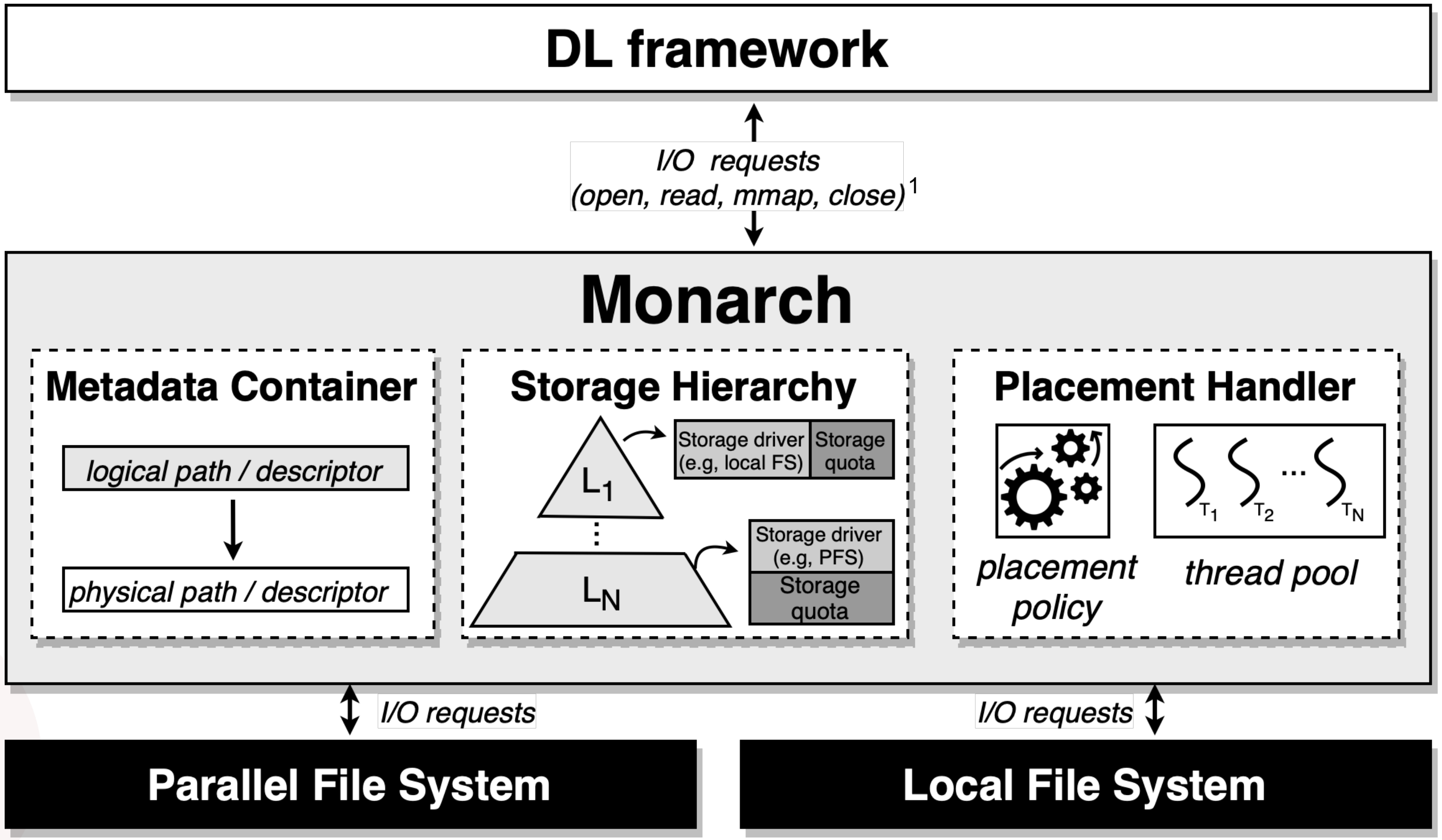
Challenges

- **Users** may **not** be **aware** of **local disks**
- **Manually** copying data to the local disk is **challenging**
  - The dataset may not fit entirely at the local disk
- The solution must be **portable** for different DL frameworks
  - **Non-intrusive** - i.e., avoids changing the framework's source-code
  - **Tuned** for DL I/O workloads

Contributions

- Monarch
  - **Transparent** and **portable** storage tiering **optimized** for **DL workloads**
  - **Compatible** with **I/O optimizations** implemented at existing DL frameworks
    - Caching, sample-based prefetching, optimized data formats
- Prototype and experimental validation
  - **Integration** with PyTorch and TensorFlow, **without** any **code changes**
  - Experimental validation with different **dataset sizes** and DL **models**

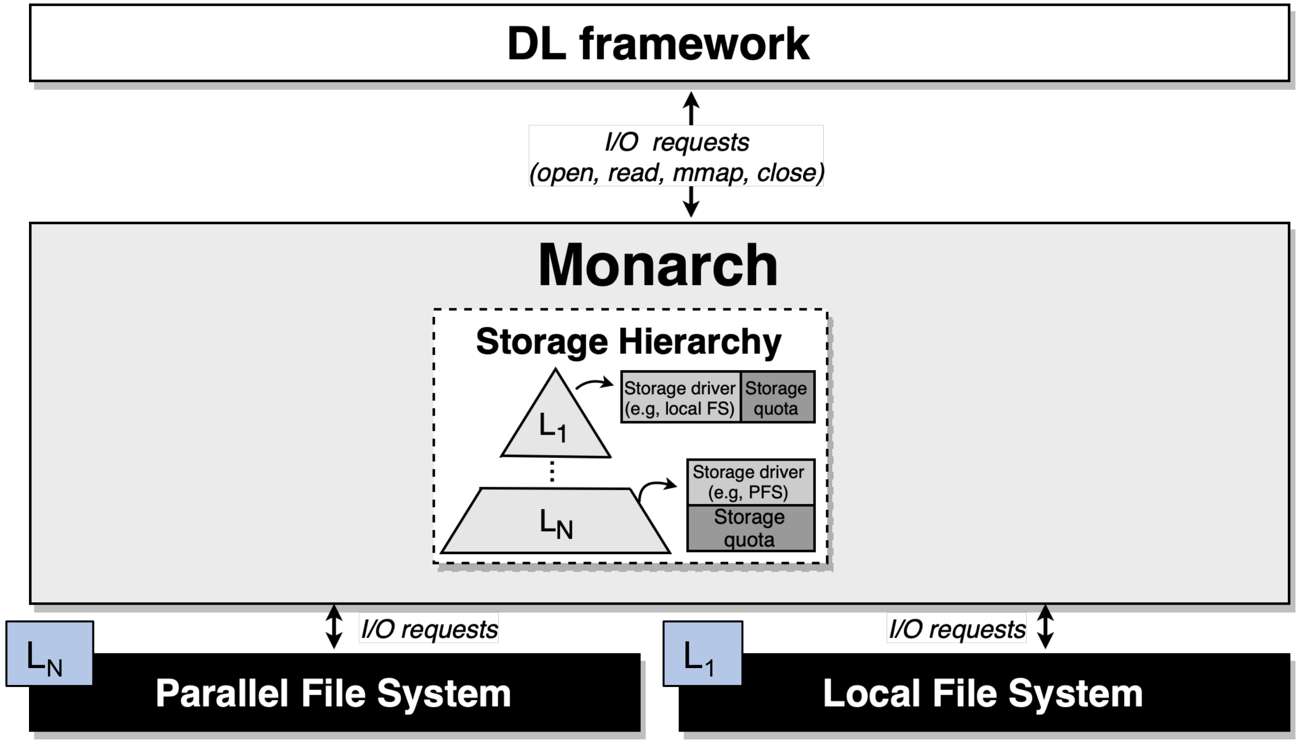
Monarch



[1]LD_PRELOAD is used for intercepting POSIX calls

Storage Hierarchy

- Layered design ($L_{1 \text{ to } N}$)
  - **$L_{1 \text{ to } N-1}$**: data caching
  - **L_N**: full dataset (read-only)
  - Organized by different criteria (e.g., performance, energy)
- Each layer includes
  - **Storage driver –** modular plugin abstracting different backends
  - **Storage quota** – tracks available storage space



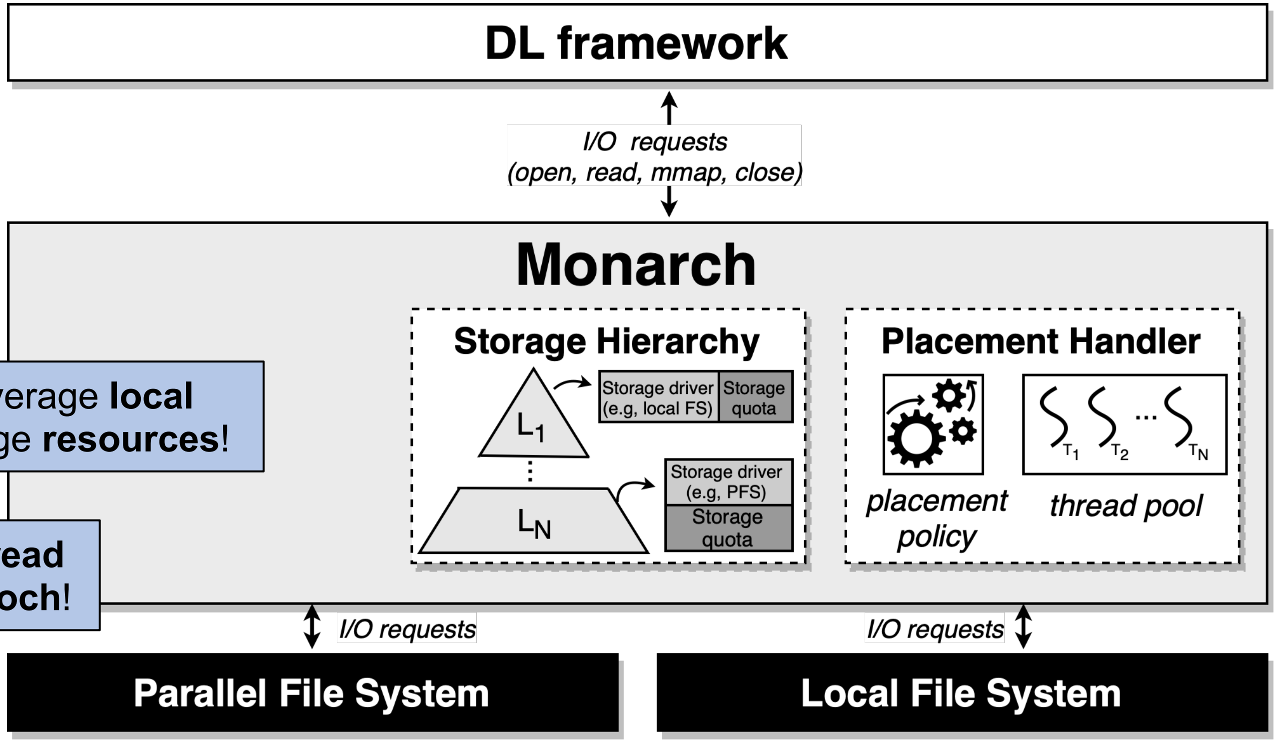
Placement Handler

- Placement policy
  - Defines the data to be cached and evicted at layers $L_{1 \text{ to } N-1}$
  - Monarch's placement strategy
    - **Top layers are filled first** (i.e., until their quota is reached)
    - **No eviction policy**

Leverage **local** storage **resources**!

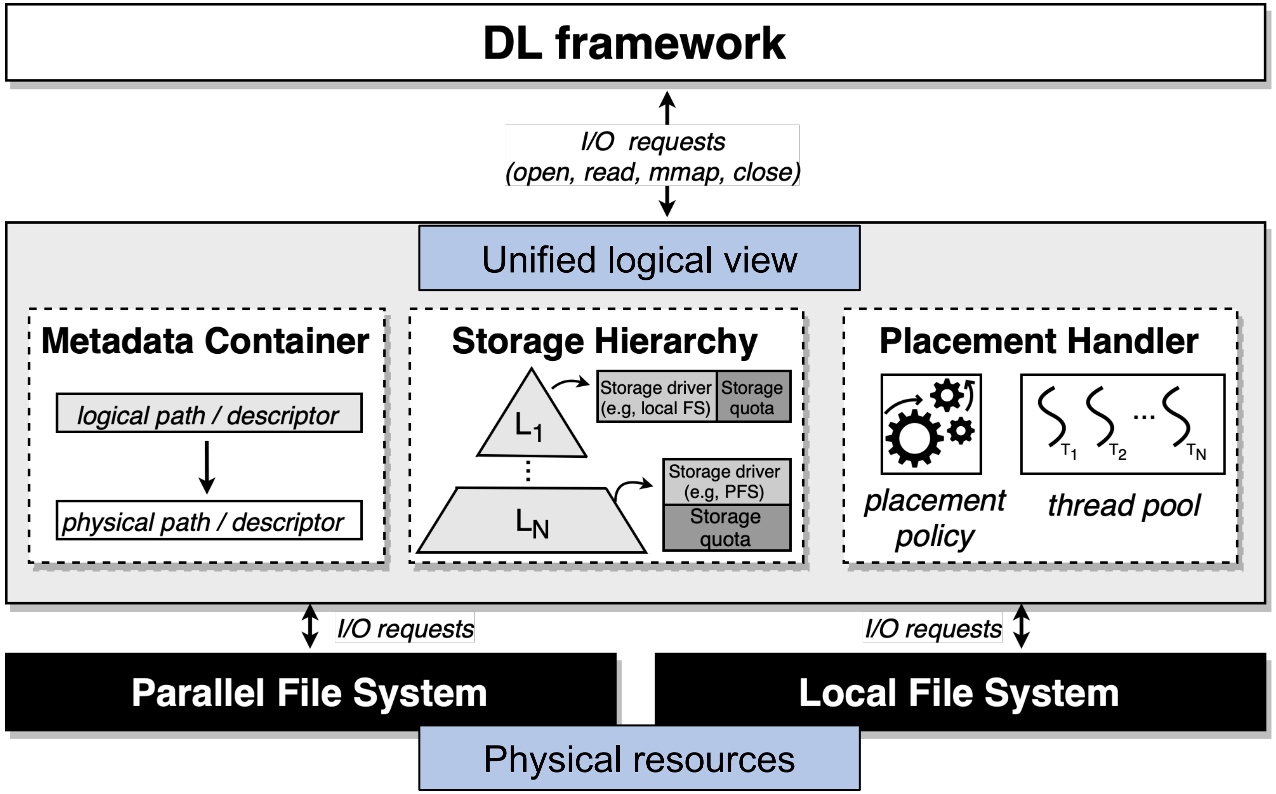
The **full dataset** is **read** for each **training epoch**!

- Thread pool
  - **Background** data fetching and caching
  - **Prefetching** for large files (e.g., TFRecords)

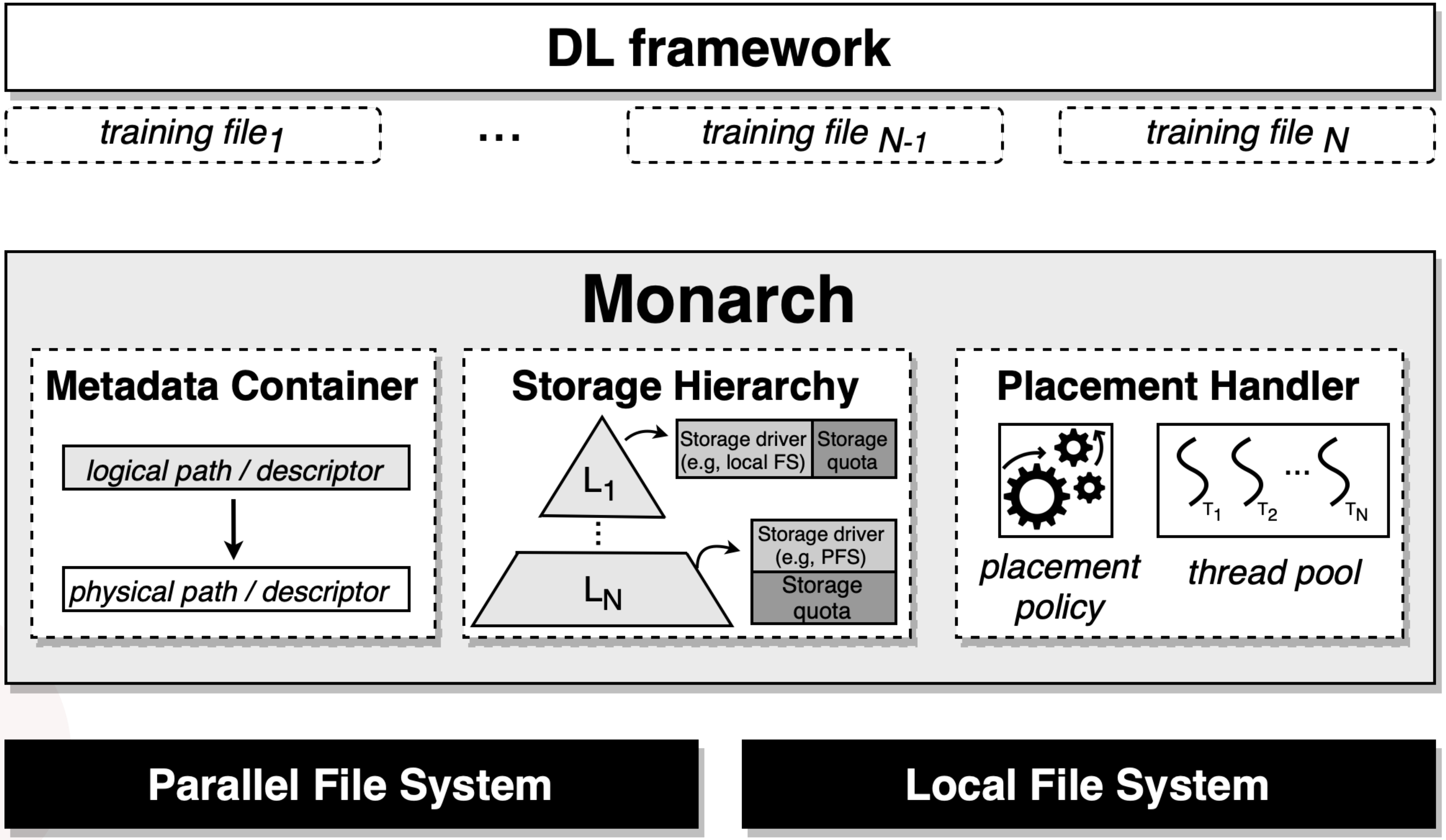


Metadata Container

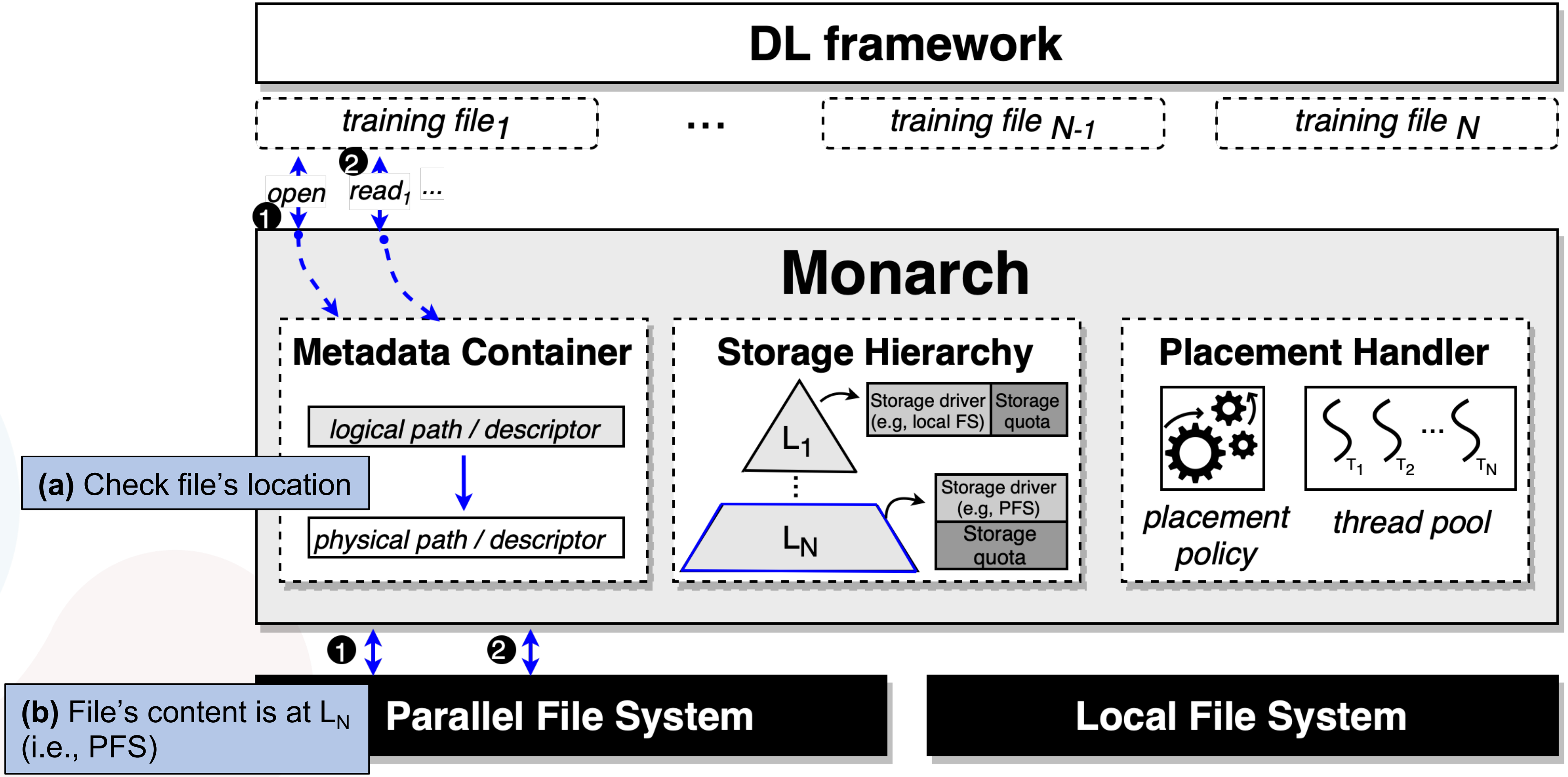
- Enables **transparent** tiering
- **Unified logical view** of storage resources for DL frameworks
  - Avoids modifying existing frameworks
- **Translation** of **logical** to **physical** storage resources
  - File paths and descriptors



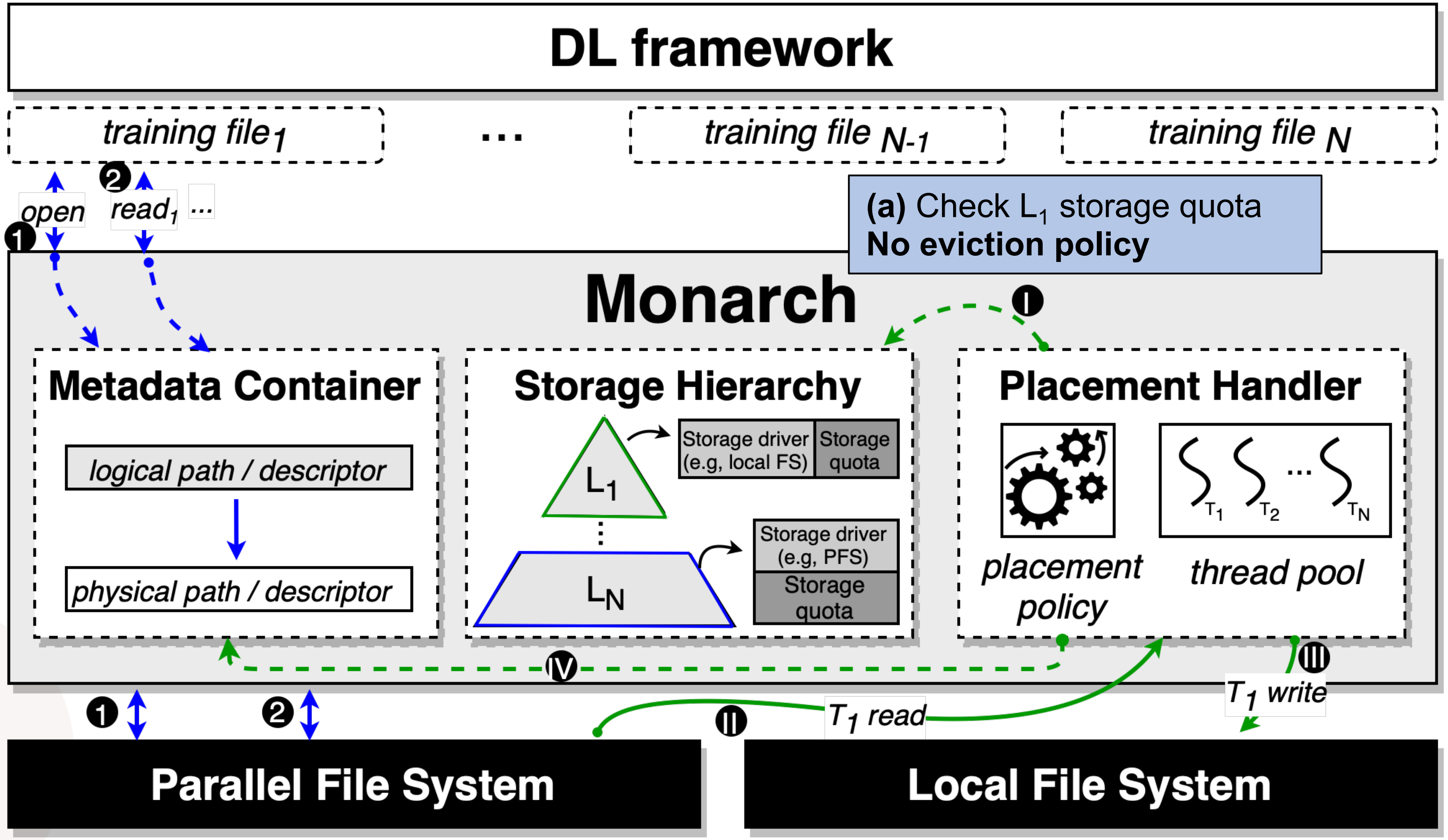
Flow of I/O Requests



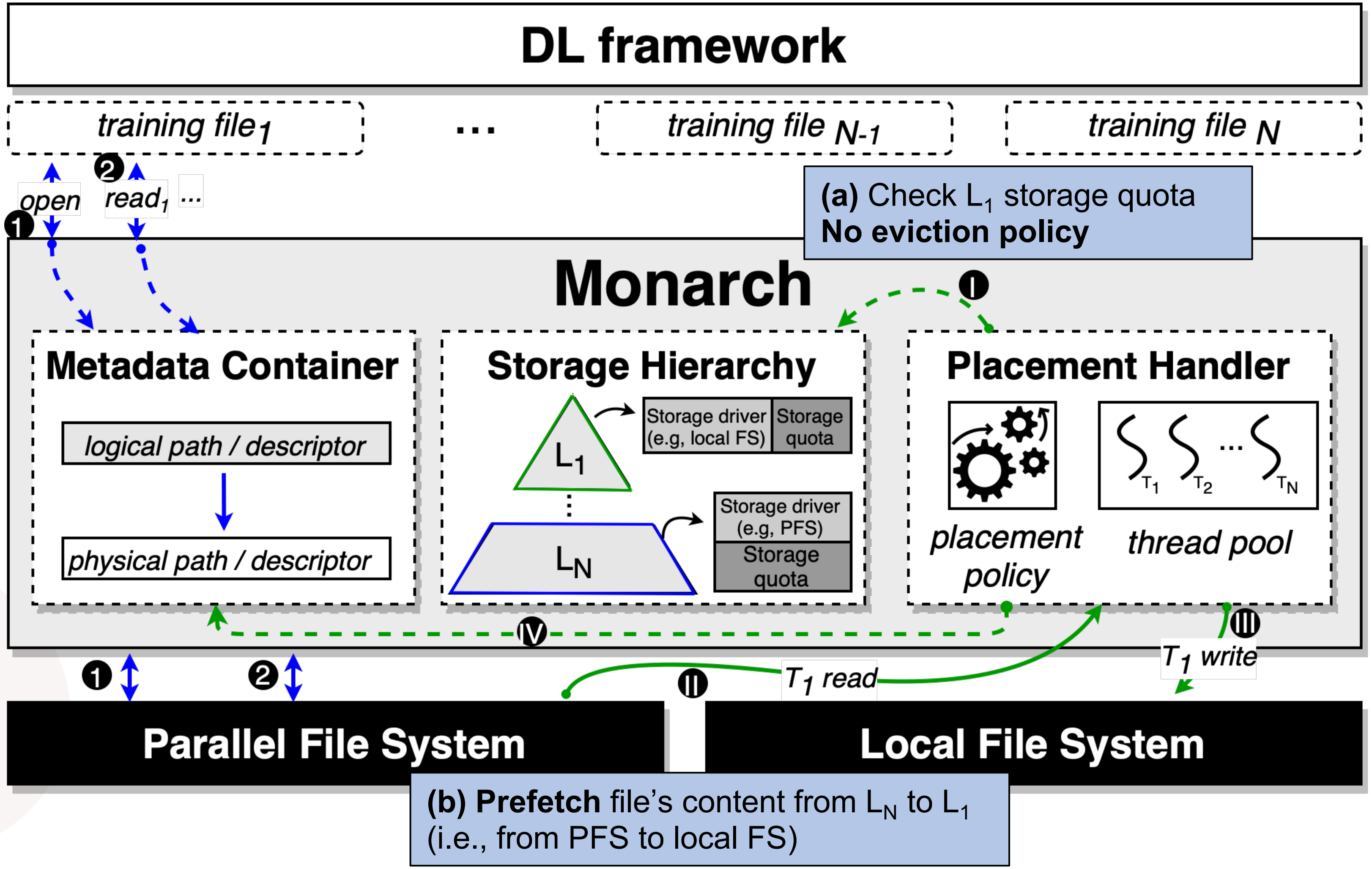
Data is Stored at L_N



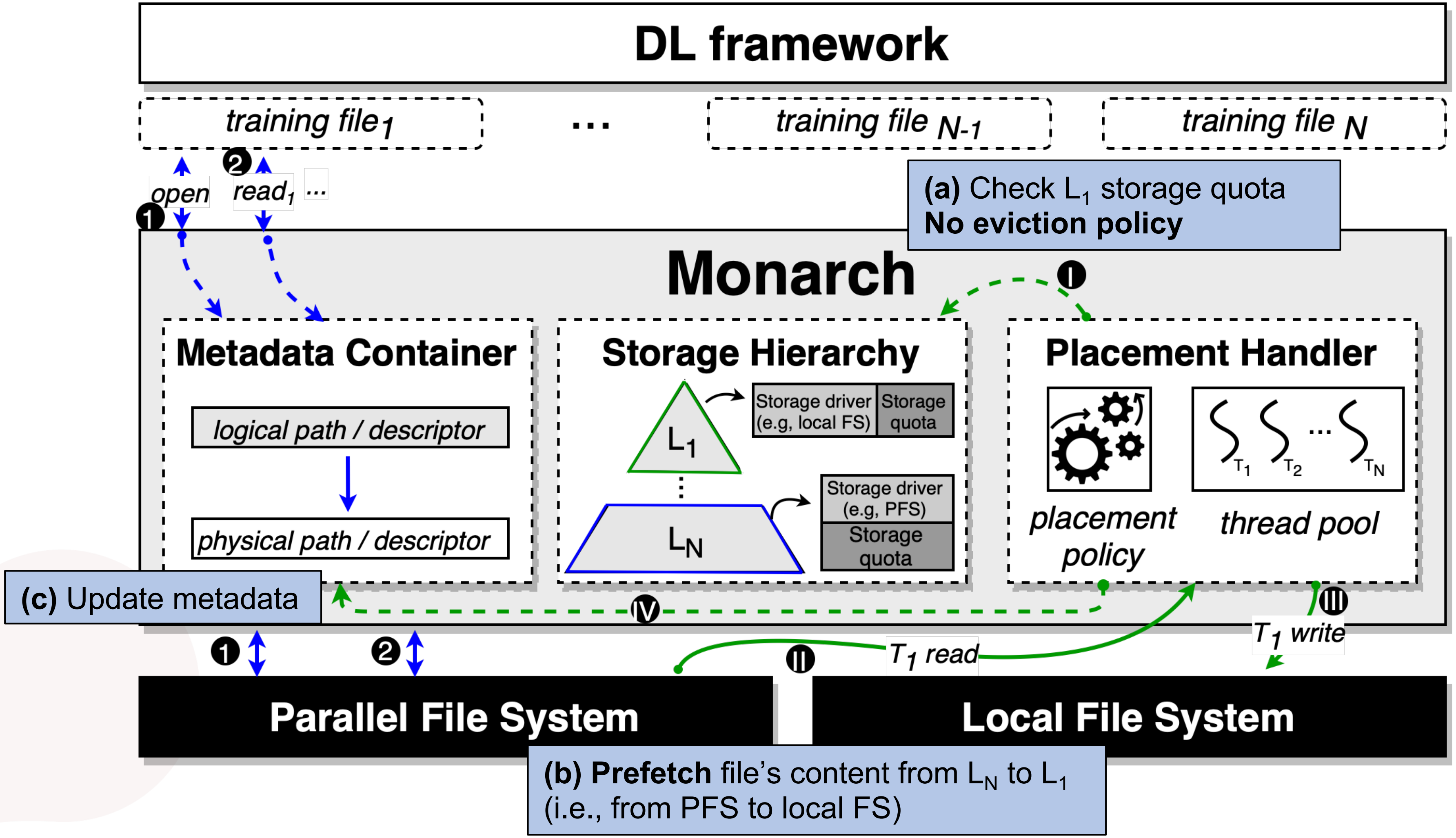
Background Data Placement



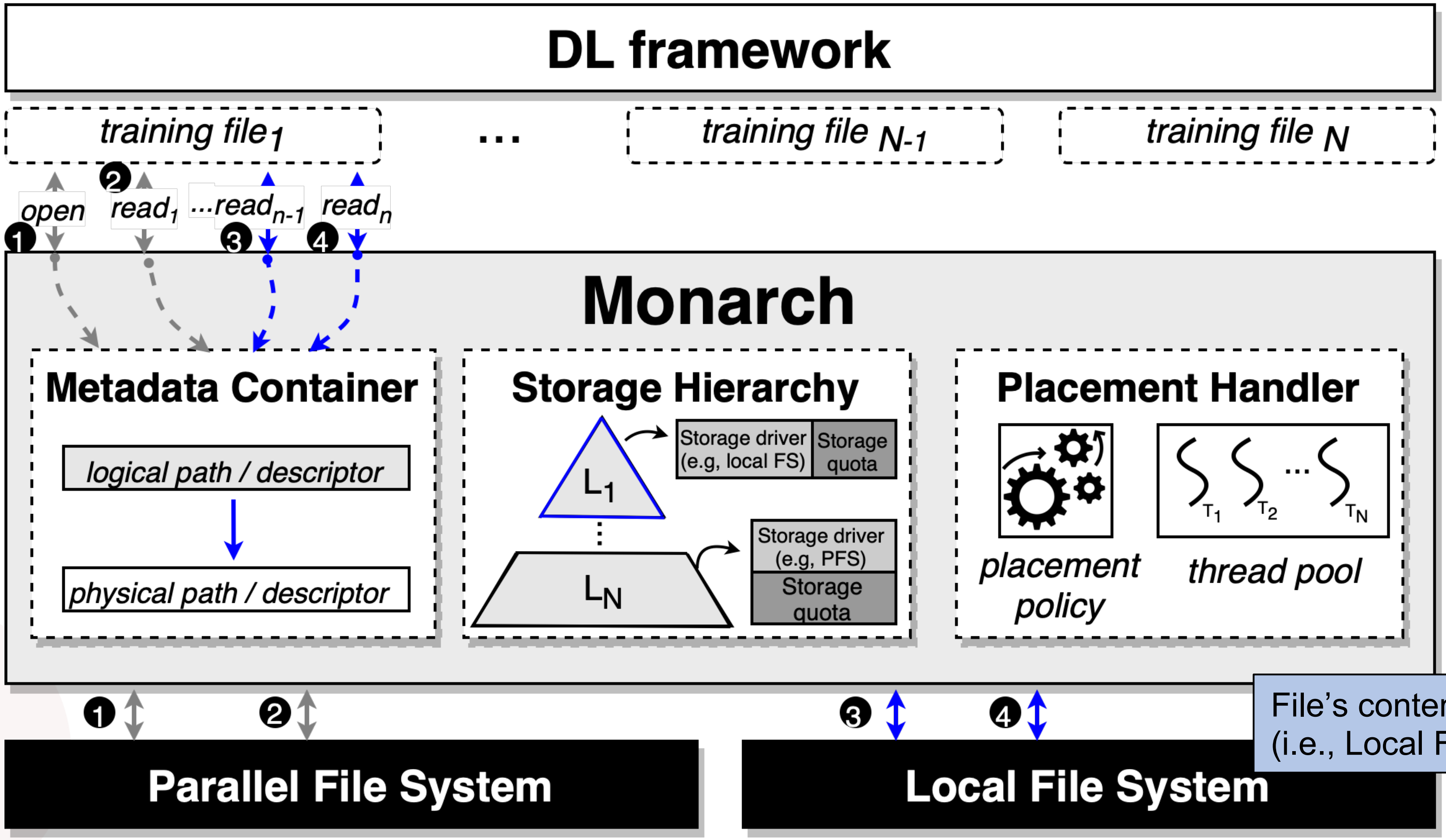
Background Data Placement



Background Data Placement

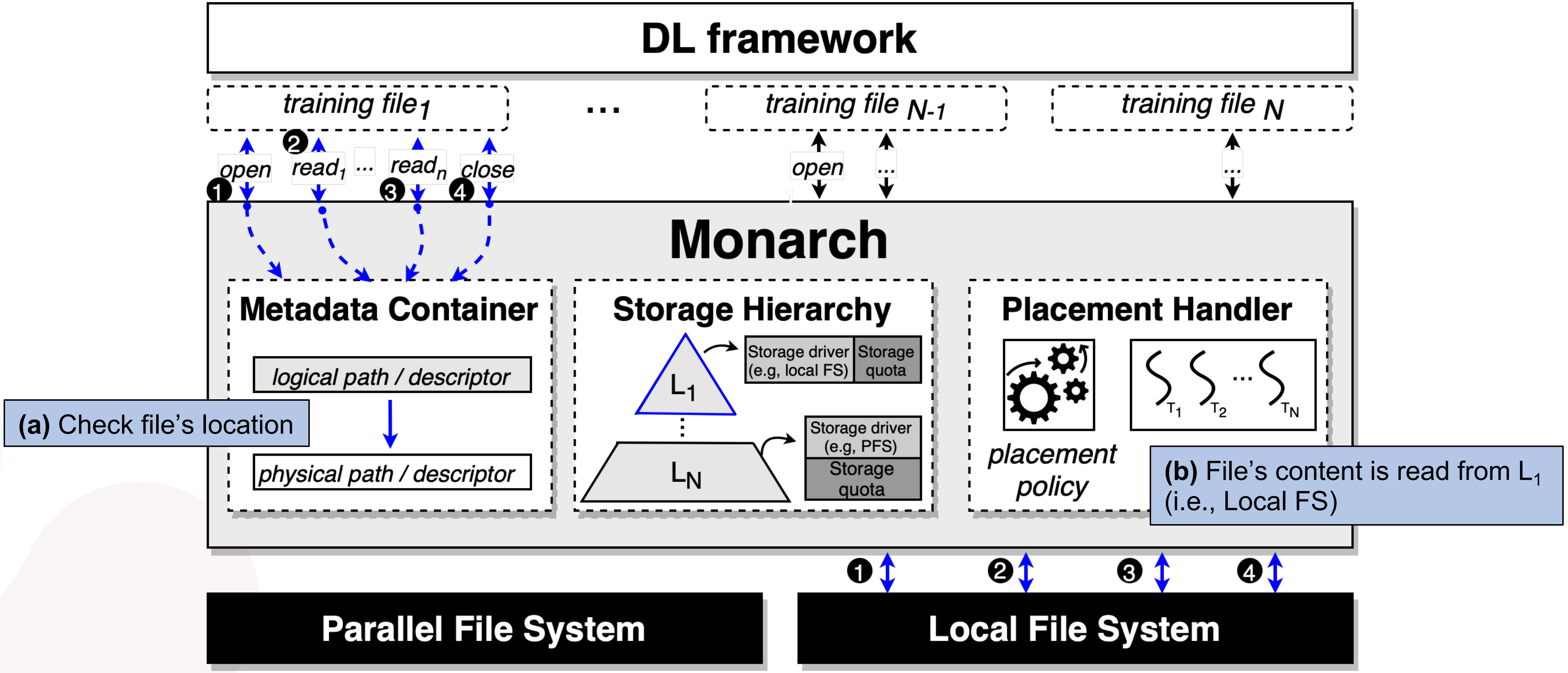


Subsequent I/O Requests



File's content is now at L_1 (i.e., Local FS)

Next Epoch - Data is Cached at L_1



Experimental Evaluation

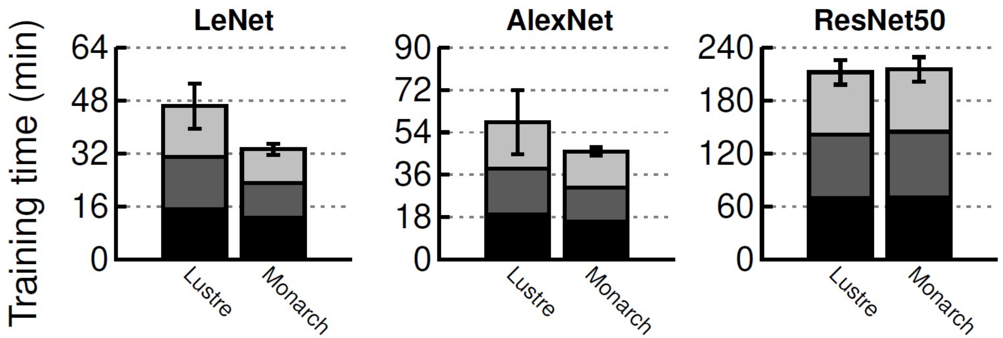
- Frontera compute node with **2x 16-core Intel Xeon processors, 4x Nvidia Quadro, 68 GiB of RAM**, and a **119 GiB SSD disk partition**[1]

- Dataset, workloads and setups[2]
    - **ImageNet-1** dataset with **200 GiB** (TFRecords)
    - **LeNet**, **AlexNet** (I/O-bound) and **ResNet-50** (compute-bound) models
    - **TensorFlow** and **PyTorch + DALI** (caching and prefetching enabled)
        - **Lustre:** data is read from the PFS (without using Monarch)
        - **Monarch:** storage tiering (local disk + PFS) is enabled by Monarch

[1] RAM and disk space were limited to ensure that the 200 GiB ImageNet-1 dataset cannot be fully cached

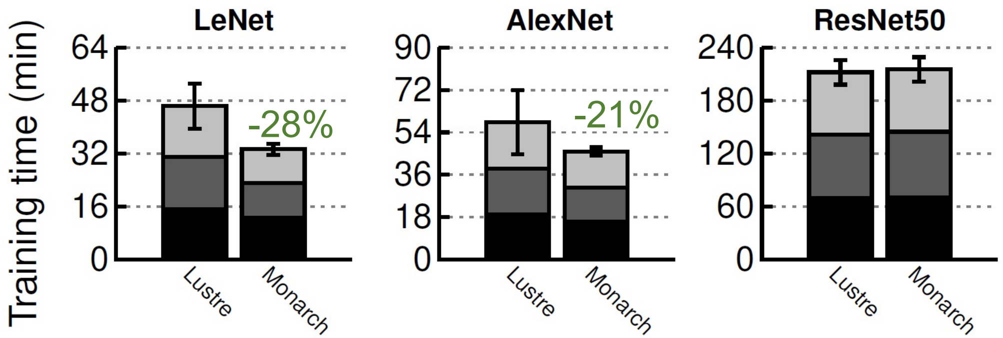
[2] Results for other dataset sizes, workloads and setups can be checked at the paper

TensorFlow - 200GiB Dataset



Average training time (3 epochs) when reading data from the PFS (**Lustre**) and with **Monarch**

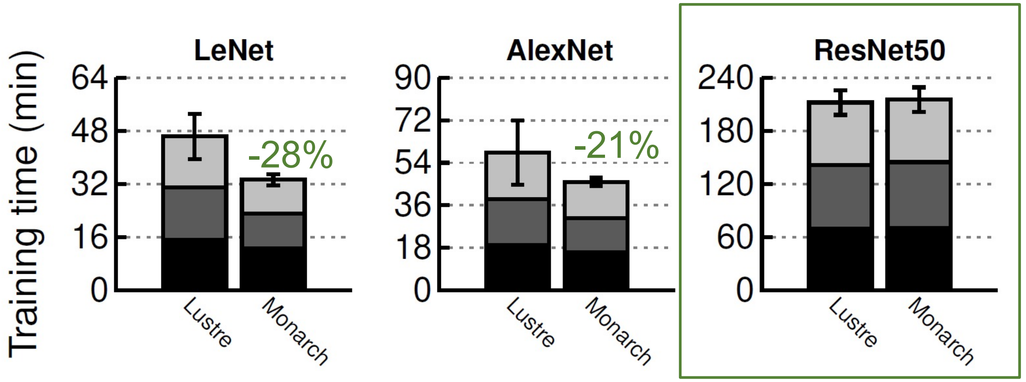
TensorFlow - 200GiB Dataset



Average training time (3 epochs) when reading data from the PFS (**Lustre**) and with **Monarch**

With Monarch:
- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)

TensorFlow - 200GiB Dataset

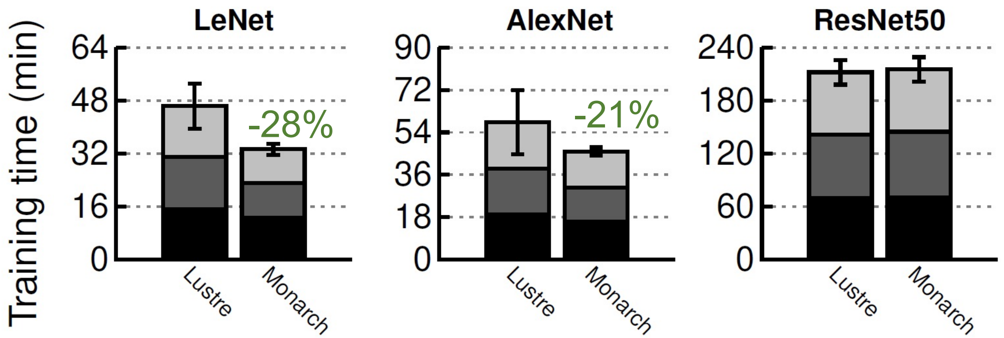


Average training time (3 epochs) when reading data from the PFS (**Lustre**) and with **Monarch**

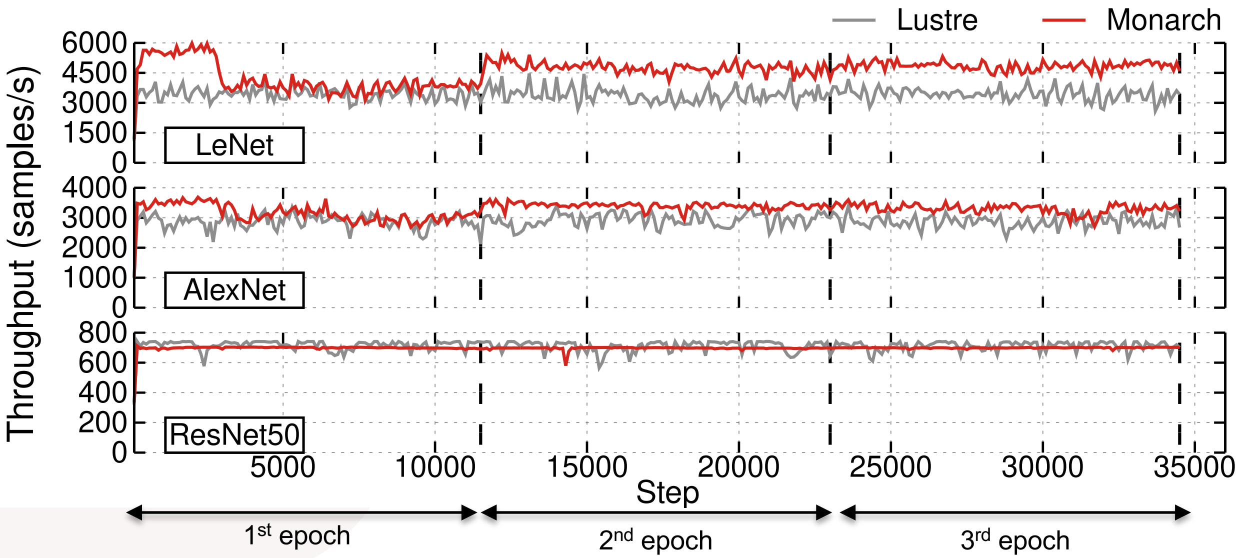
With Monarch:

- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)
- Training time is similar for ResNet50 (compute-bound)

TensorFlow - 200GiB Dataset

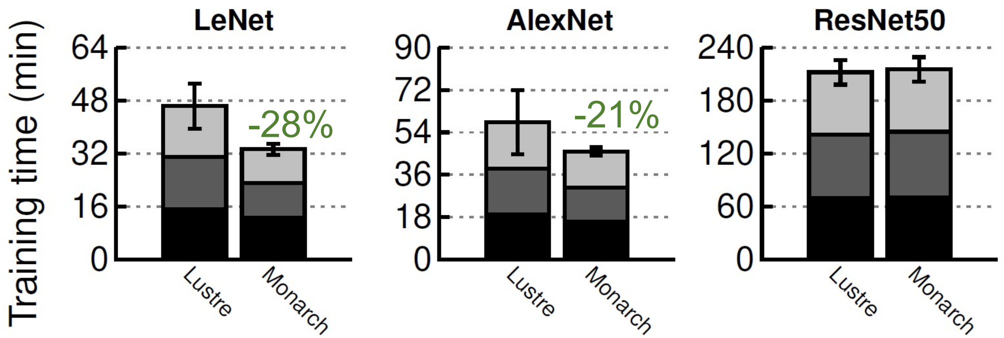


With Monarch:
- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)
- Training time is similar for ResNet50 (compute-bound)

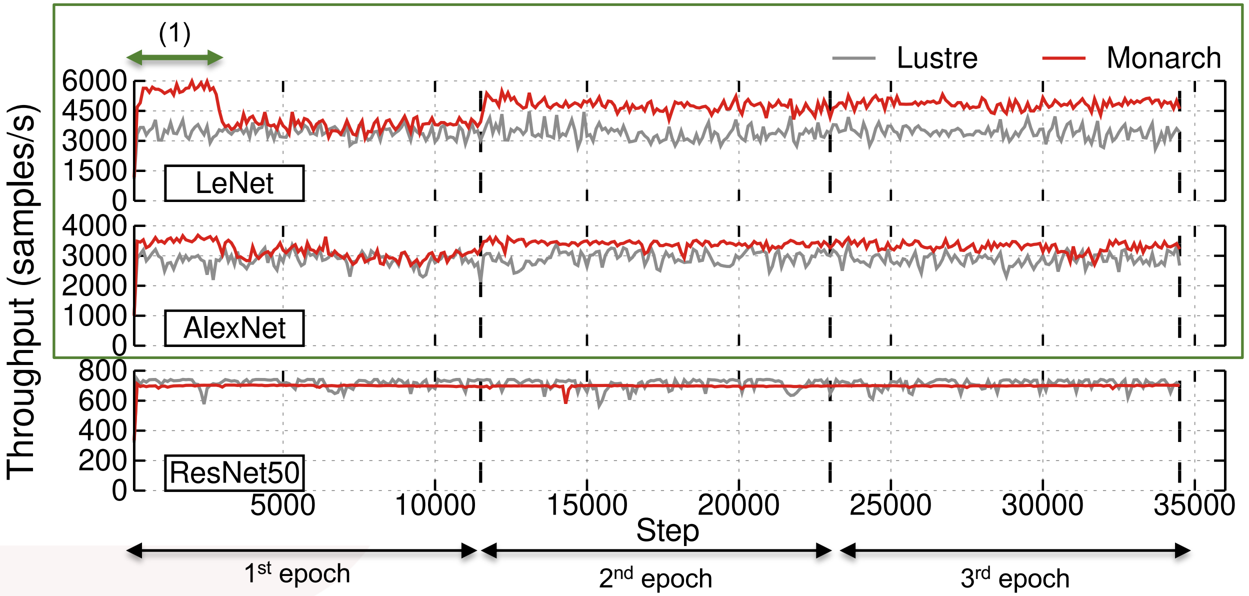


Throughput, in samples per second, when reading data from the PFS (**Lustre**) and with **Monarch**

TensorFlow - 200GiB Dataset



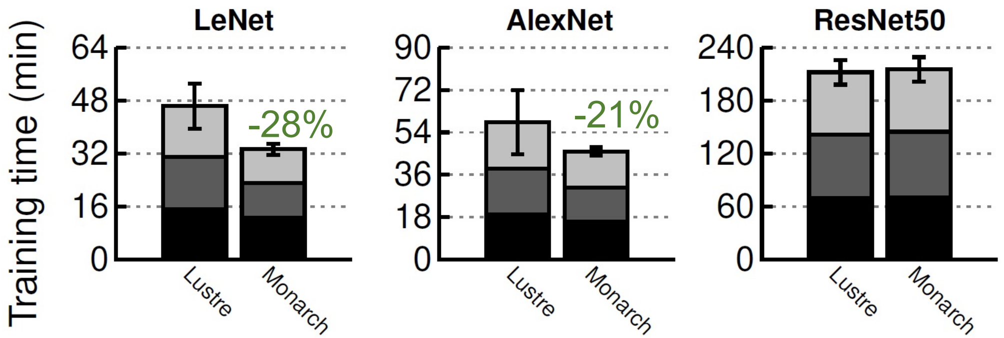
With Monarch:
- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)
- Training time is similar for ResNet50 (compute-bound)



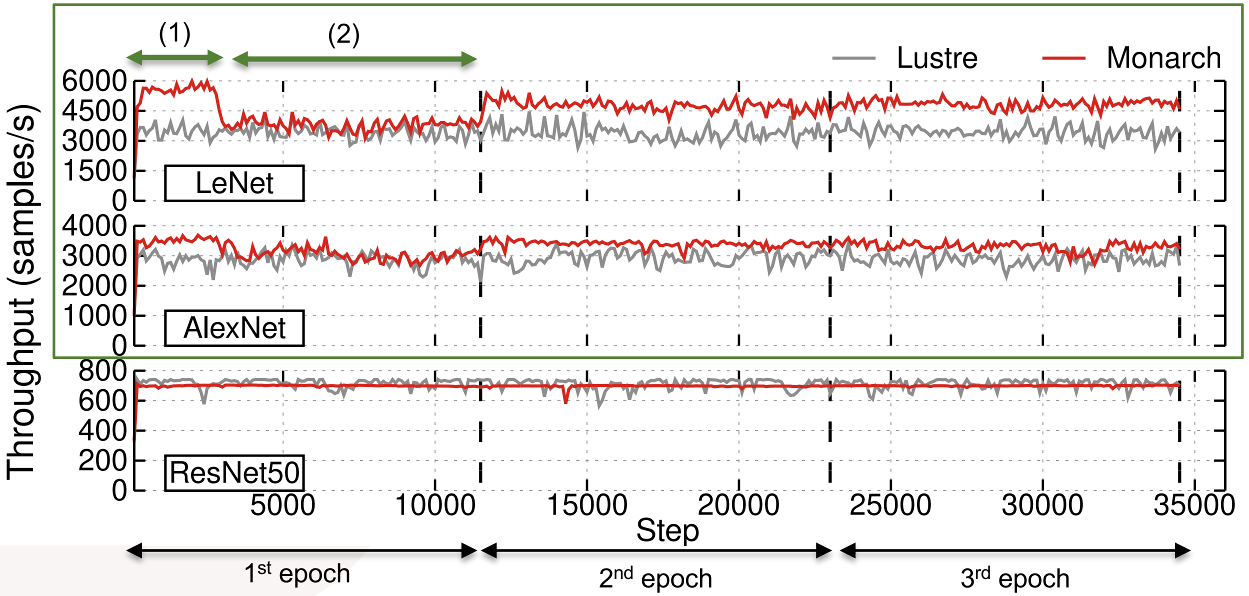
Throughput, in samples per second, when reading data from the PFS (**Lustre**) and with **Monarch**

With Monarch, for LeNet and AlexNet models:
1. **Improved performance** due to Monarch's file prefetching (better usage of the local page cache)

TensorFlow - 200GiB Dataset



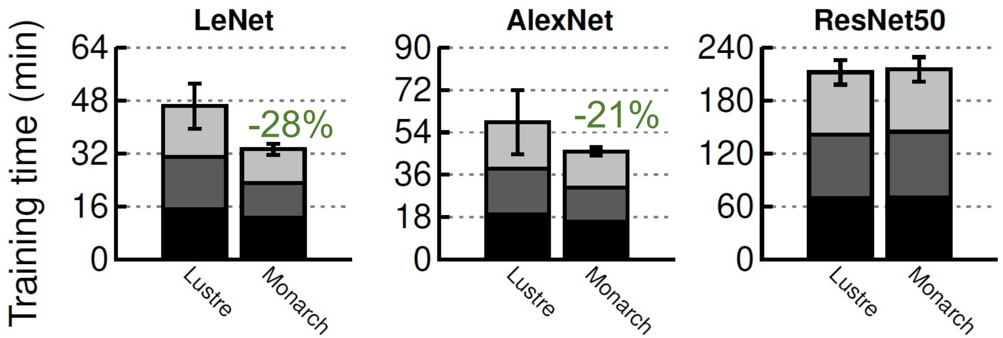
With Monarch:
- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)
- Training time is similar for ResNet50 (compute-bound)



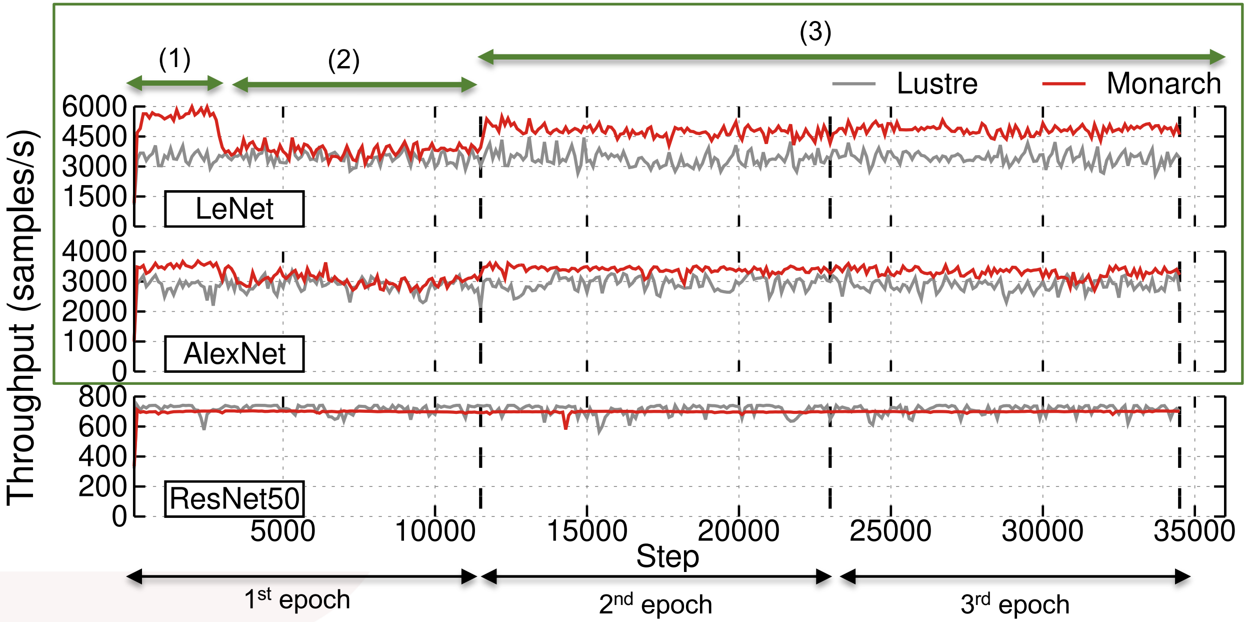
Throughput, in samples per second, when reading data from the PFS (**Lustre**) and with **Monarch**

With Monarch, for LeNet and AlexNet models:
1. **Improved performance** due to Monarch's file prefetching (better usage of the local page cache)
2. **Similar performance** when the page cache becomes full

TensorFlow - 200GiB Dataset



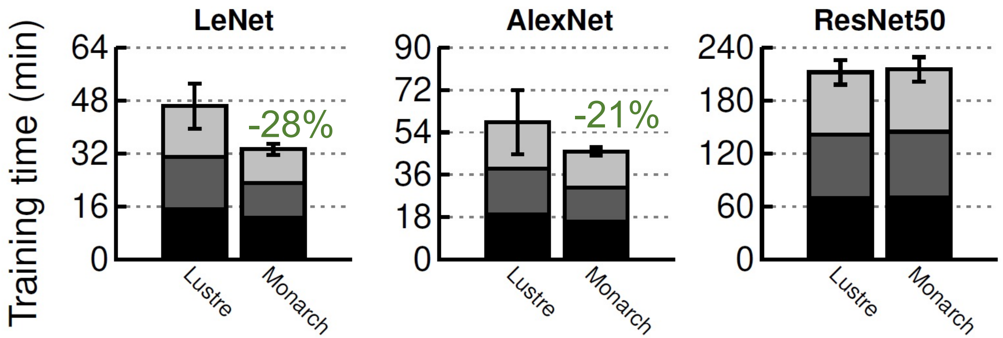
With Monarch:
- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)
- Training time is similar for ResNet50 (compute-bound)



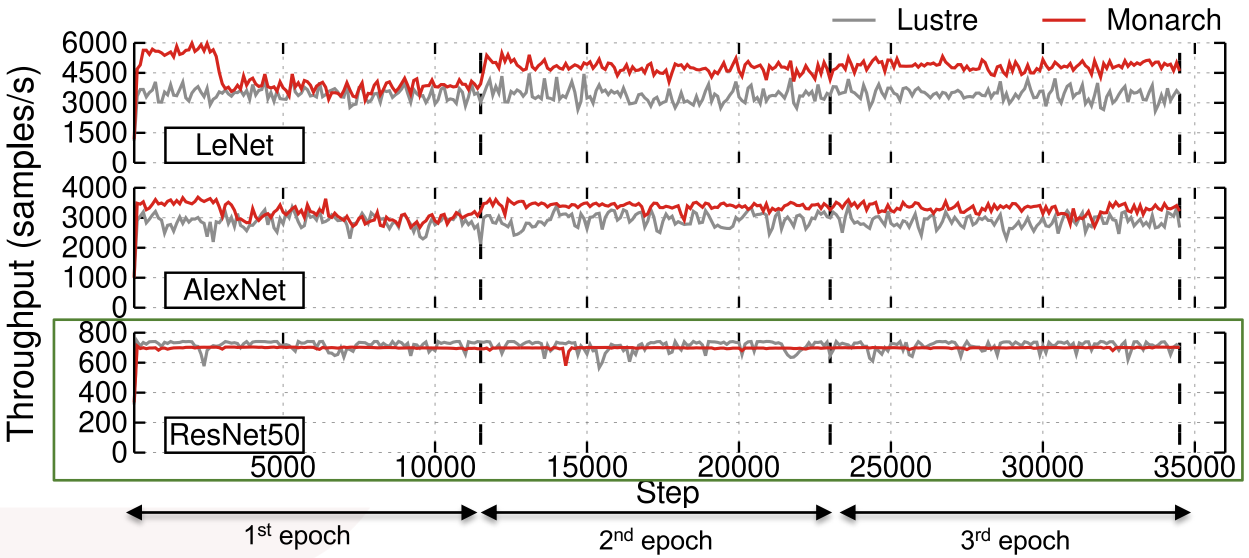
Throughput, in samples per second, when reading data from the PFS (**Lustre**) and with **Monarch**

With Monarch, for LeNet and AlexNet models:
1. **Improved performance** due to Monarch's file prefetching (better usage of the local page cache)
2. **Similar performance** when the page cache becomes full
3. **Better performance** for the second and third training epochs

TensorFlow - 200GiB Dataset



With Monarch:
- Training time is reduced by **28%** for LeNet (-13 min)
- Training time is reduced by **21%** for AlexNet (-12.5 min)
- Training time is similar for ResNet50 (compute-bound)

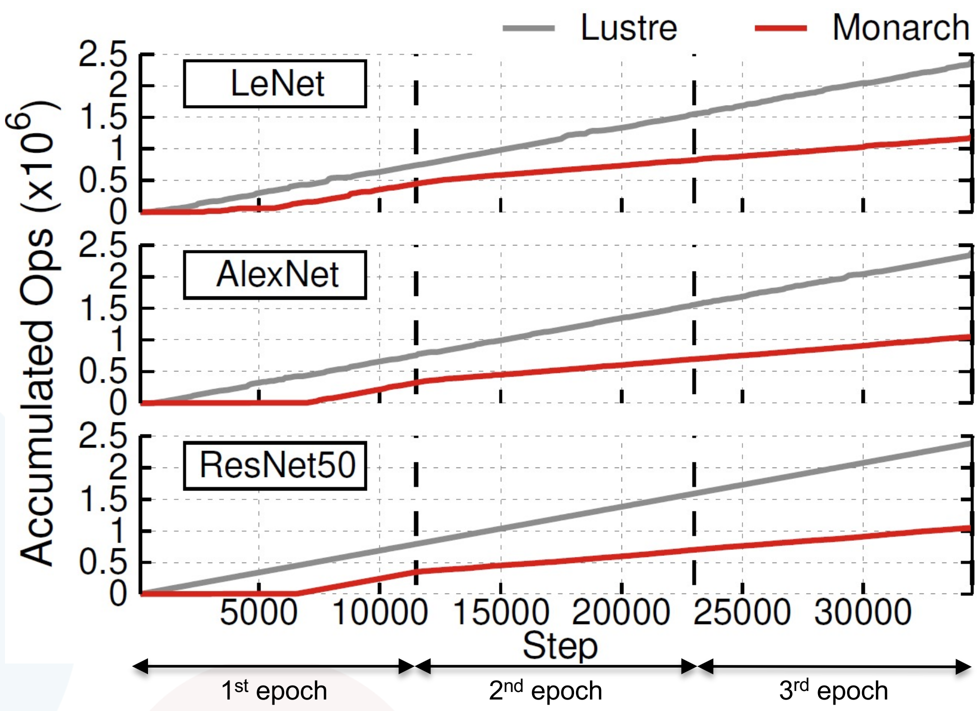


Throughput, in samples per second, when reading data from the PFS (**Lustre**) and with **Monarch**

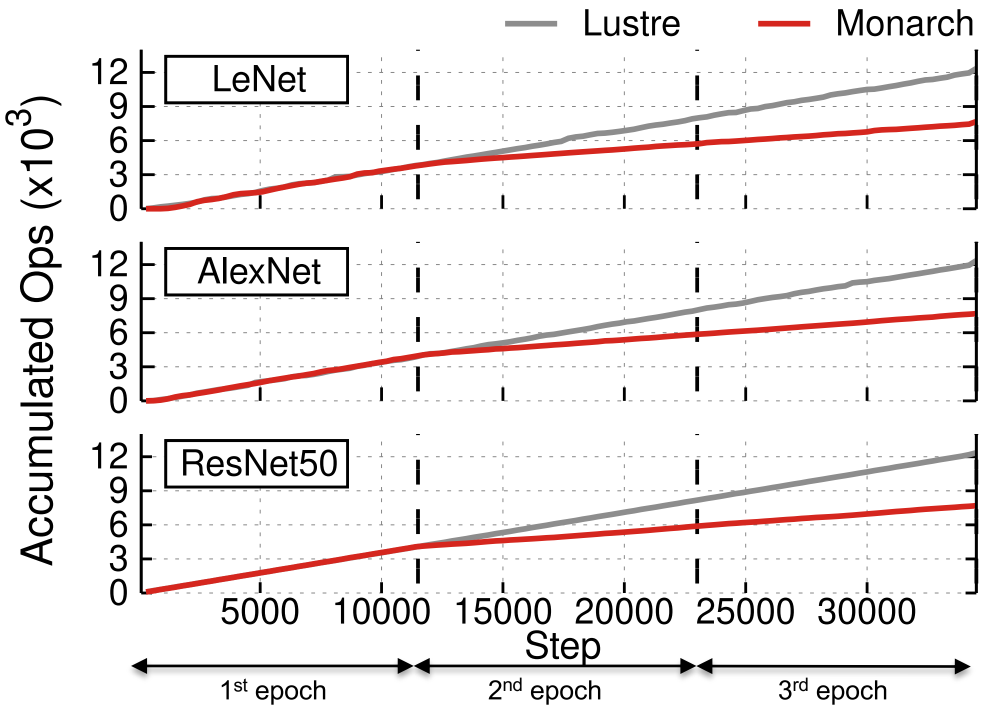
With Monarch, for LeNet and AlexNet models:
1. **Improved performance** due to Monarch's file prefetching (better usage of the local page cache)
2. **Similar performance** when the page cache becomes full
3. **Better performance** for the second and third training epochs

For ResNet50:
- Similar throughput but **less variance**

TensorFlow - 200GiB Dataset

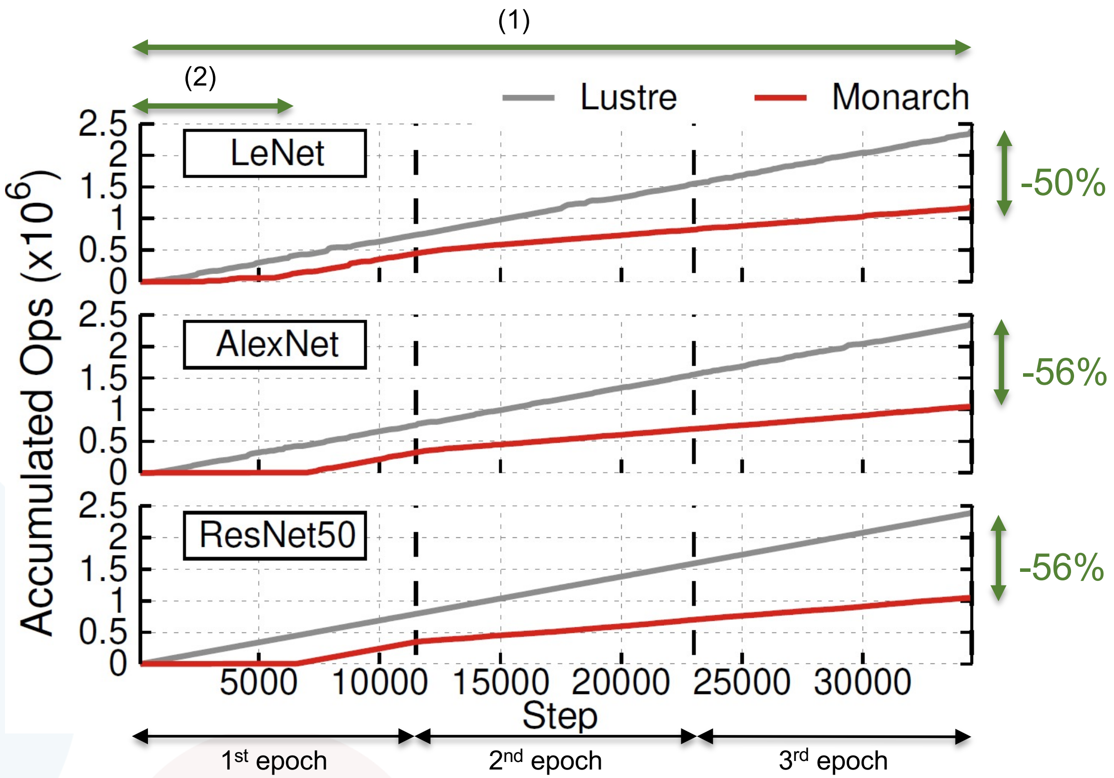


PFS's read operations by **Lustre** and **Monarch**

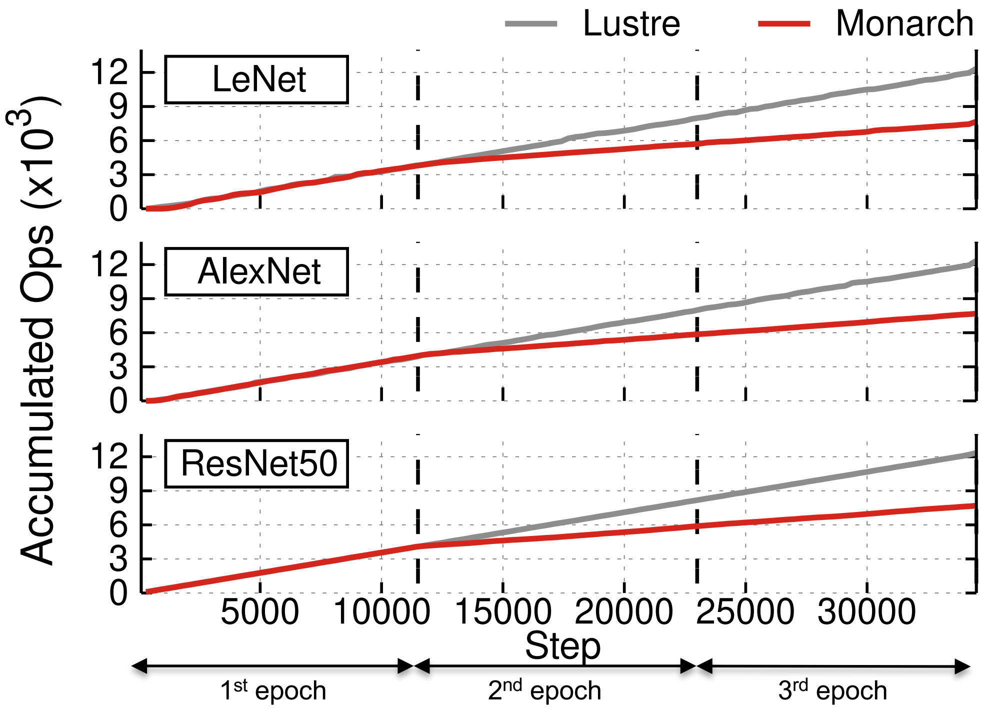


PFS's metadata (open + close) operations by **Lustre** and **Monarch**

TensorFlow - 200GiB Dataset



PFS's read operations by **Lustre** and **Monarch**

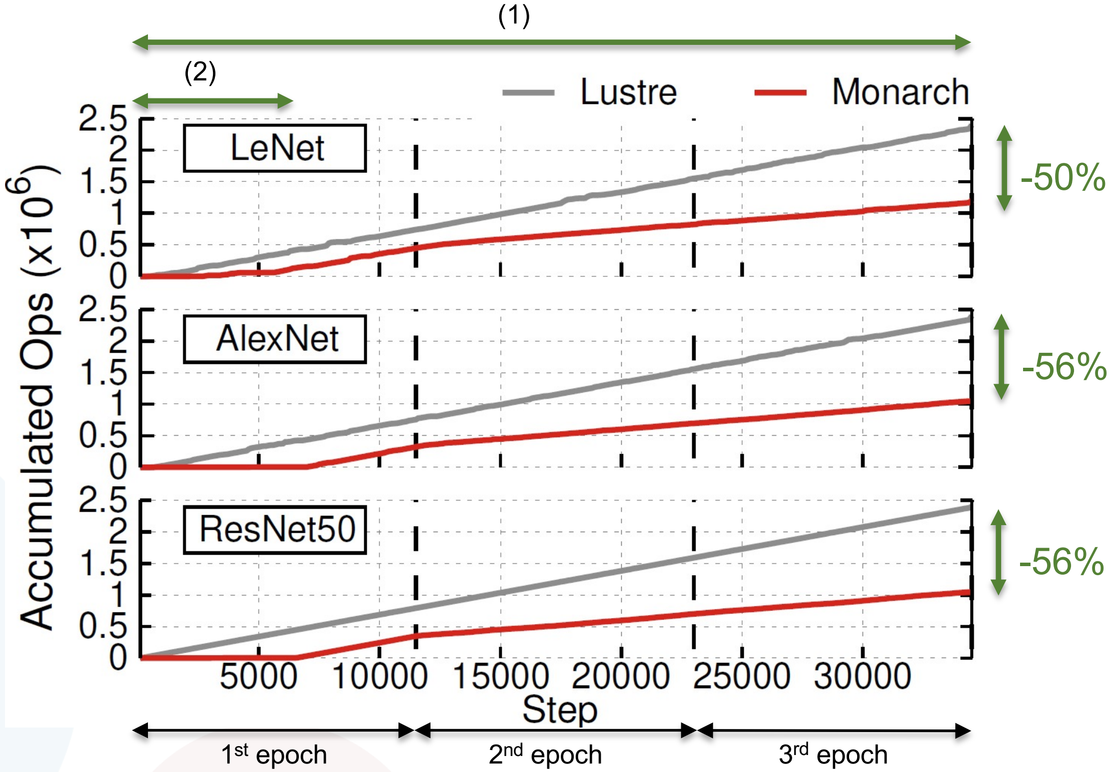


PFS's metadata (open + close) operations by **Lustre** and **Monarch**

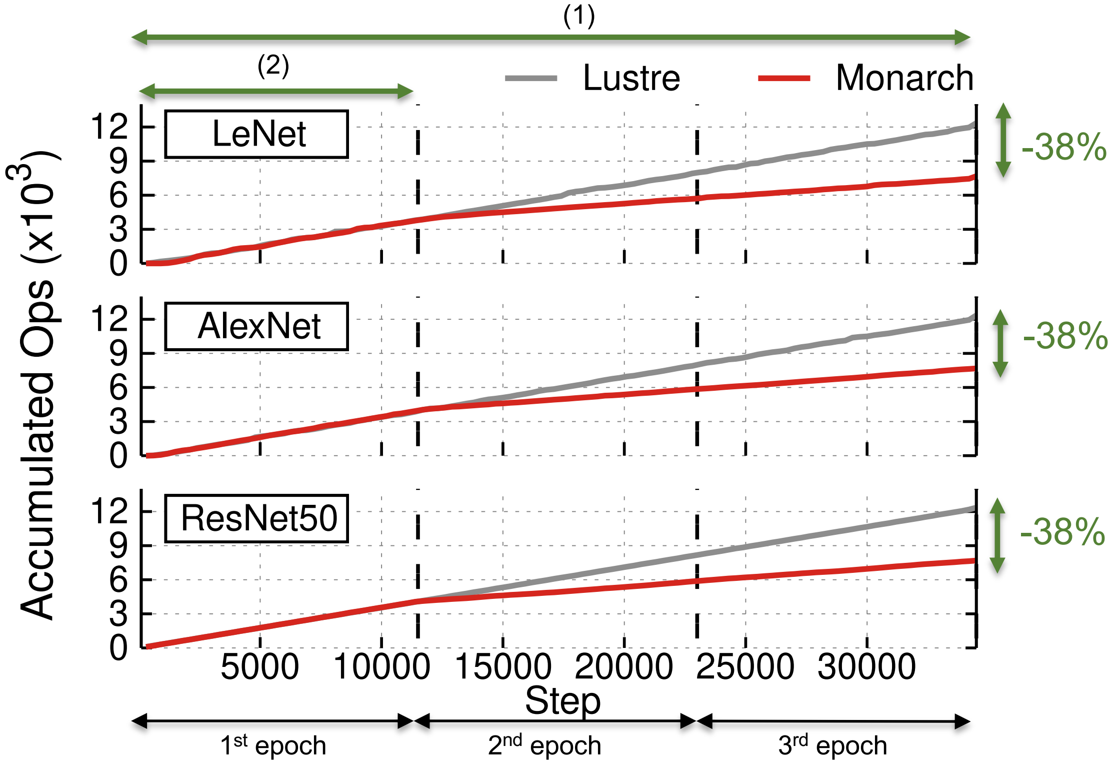
With Monarch:

1. PFS read operations reduced by up to **56%**
2. Prefetching reduces number of reads at first epoch

TensorFlow - 200GiB Dataset



PFS's read operations by **Lustre** and **Monarch**

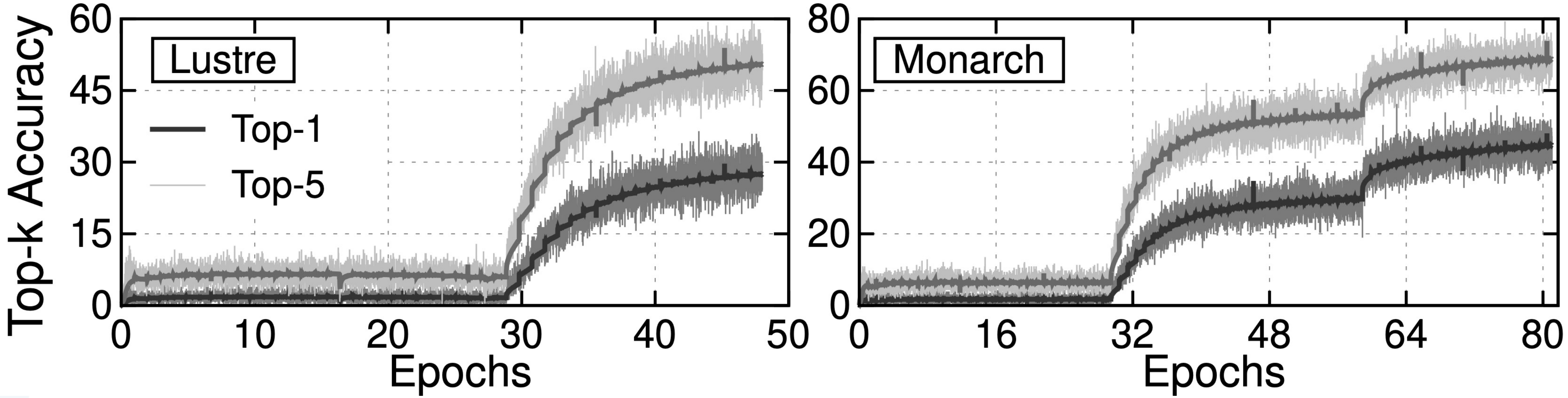


PFS's metadata (open + close) operations by **Lustre** and **Monarch**

With Monarch:
1. PFS read operations reduced by up to **56%**
2. Prefetching reduces number of reads at first epoch

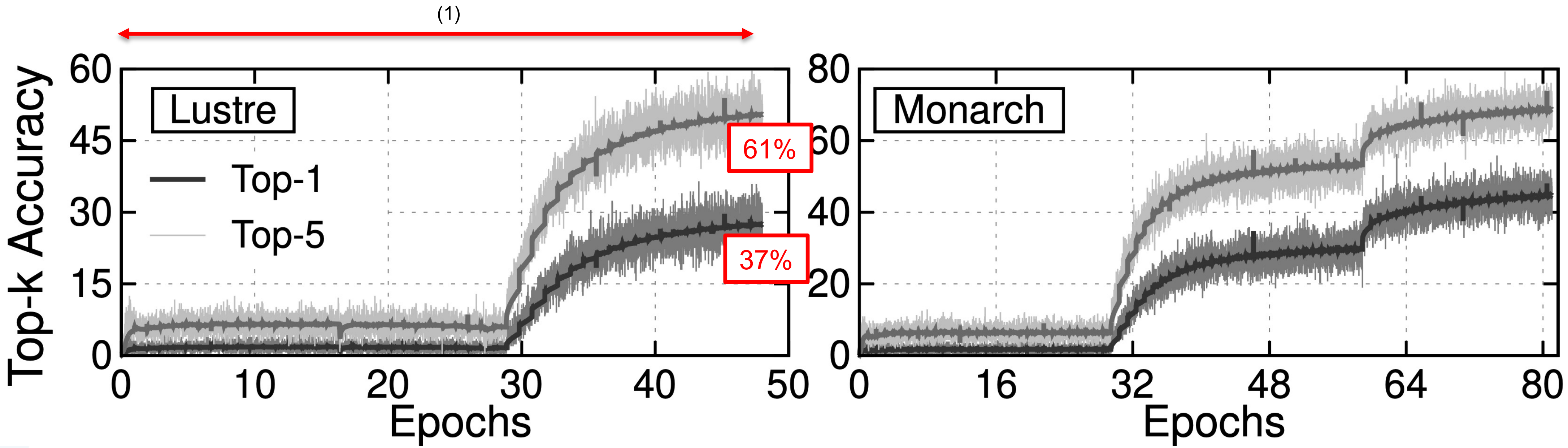
With Monarch:
1. PFS open + close operations reduced by up to **38%**
2. Same number of operations for the first training epoch

PyTorch - Long Run and Accuracy



Top-1 and top-5 accuracy for **Lustre** and **Monarch** training the AlexNet model over a 48 hours period.

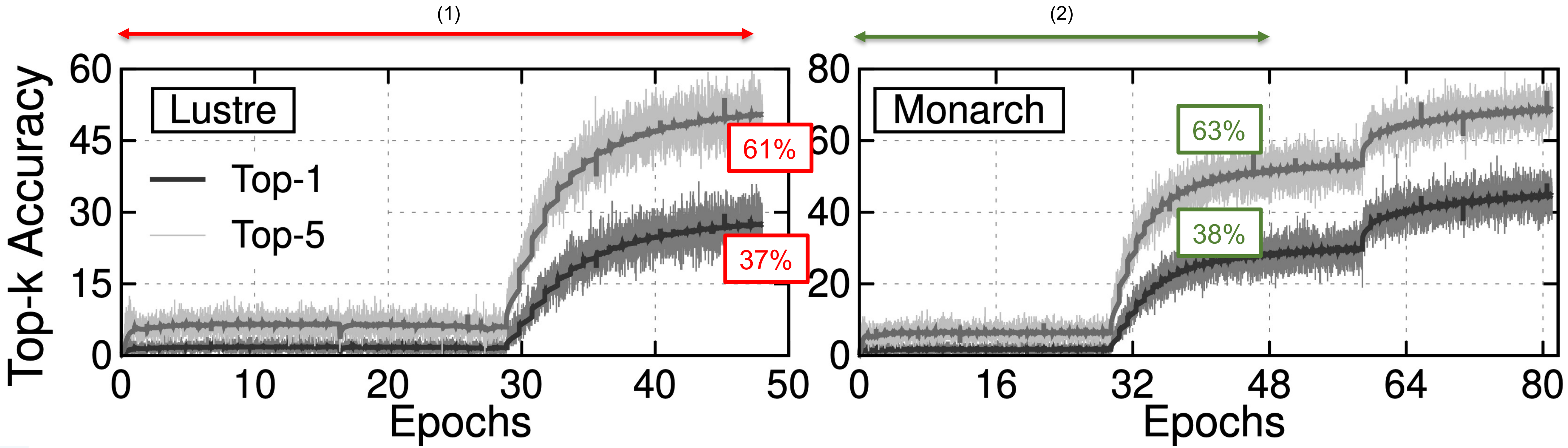
PyTorch - Long Run and Accuracy



Top-1 and top-5 accuracy for **Lustre** and **Monarch** training the AlexNet model over a 48 hours period.

1. In 48 hours, **Lustre** runs **48 epochs** and achieves **37%** and **61%** for top-1 and top-5 accuracy

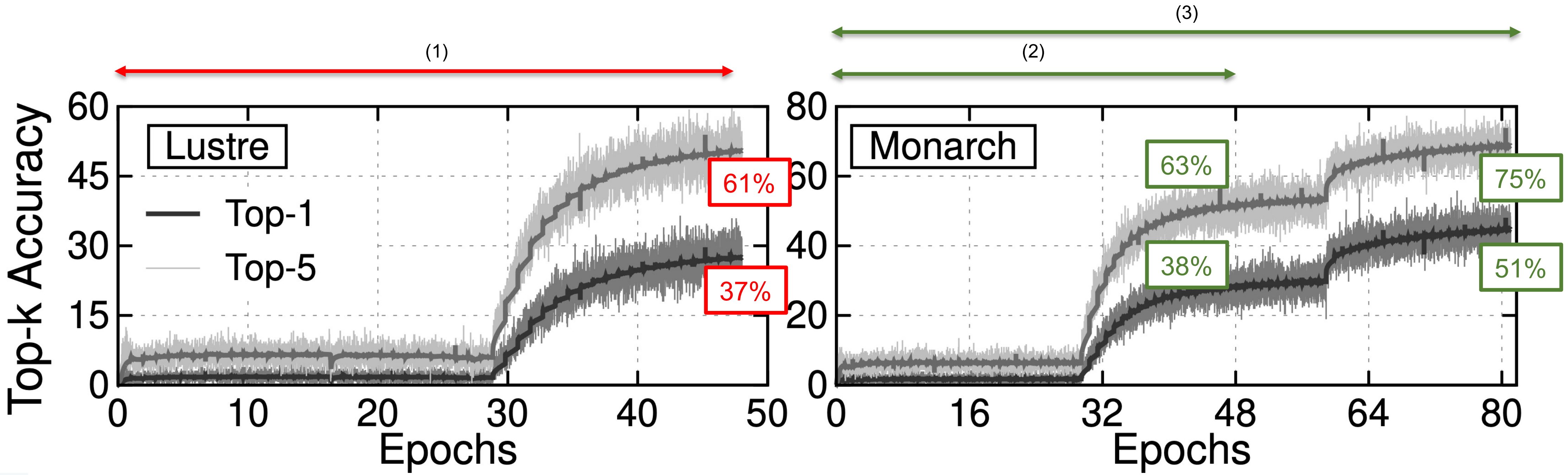
PyTorch - Long Run and Accuracy



Top-1 and top-5 accuracy for **Lustre** and **Monarch** training the AlexNet model over a 48 hours period.

1. In 48 hours, **Lustre** runs **48 epochs** and achieves **37%** and **61%** for top-1 and top-5 accuracy
2. **Monarch** completes the same number of epochs (**48**) and achieves similar accuracy in **28 hours**

PyTorch - Long Run and Accuracy



Top-1 and top-5 accuracy for **Lustre** and **Monarch** training the AlexNet model over a 48 hours period.

1. In 48 hours, **Lustre** runs **48 epochs** and achieves **37%** and **61%** for top-1 and top-5 accuracy
2. **Monarch** completes the same number of epochs (**48**) and achieves similar accuracy in **28 hours**
3. In 48 hours, **Monarch** runs **81 epochs** and achieves **51%** and **75%** for top-1 and top-5 accuracy

Conclusions

- Monarch, storage tiering for DL workloads running on HPC centers
  - **Transparent** to users
  - **Applicable** to different DL frameworks
  - **Optimized** for DL I/O patterns and large datasets
- TensorFlow and PyTorch training time **reduced** by up to **28%** and **37%**
- Number of I/O operations at the PFS **reduced** by up to **56%**
- Open-sourced at https://github.com/dsrhaslab/monarch

Accelerating Deep Learning Training Through Transparent Storage Tiering

Marco Dantas, Diogo Leitão, Peter Cui[1], Ricardo Macedo, Xinlian Liu[2], Weijia Xu[1], **João Paulo**

INESC TEC & University of Minho [1] University of Texas at Austin [2] Hood College

IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing

May 17, 2022