Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads

MONARCH: Hierarchical Storage Management for Deep Learning Frameworks

Marco Dantas, Diogo Leitão, Cláudia Correia, Ricardo Macedo, Weijia Xu*, João Paulo
INESC TEC & University of Minho, *Texas Advanced Computing Center

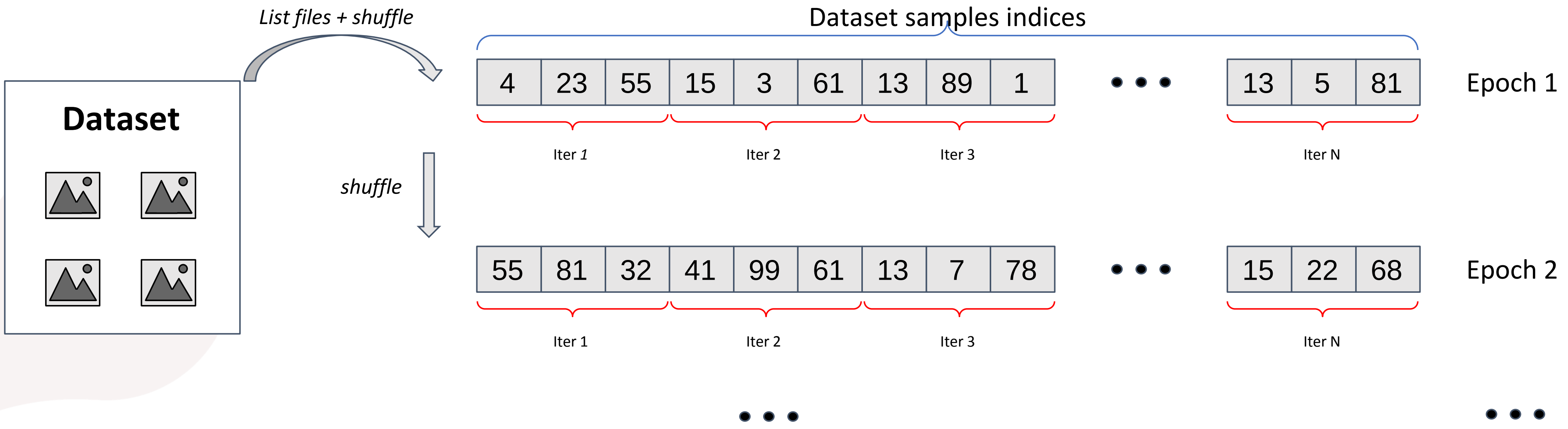
September 7, 2021

Deep Learning Jobs

- Emergence of High-Performance Computing (HPC) infrastructures for Deep Learning (DL) training.

Deep Learning Jobs

- Emergence of High-Performance Computing (HPC) infrastructures for Deep Learning (DL) training.
- DL training generally involves large datasets, computational intensive models and a large number of epochs.



HPC Shared Storage

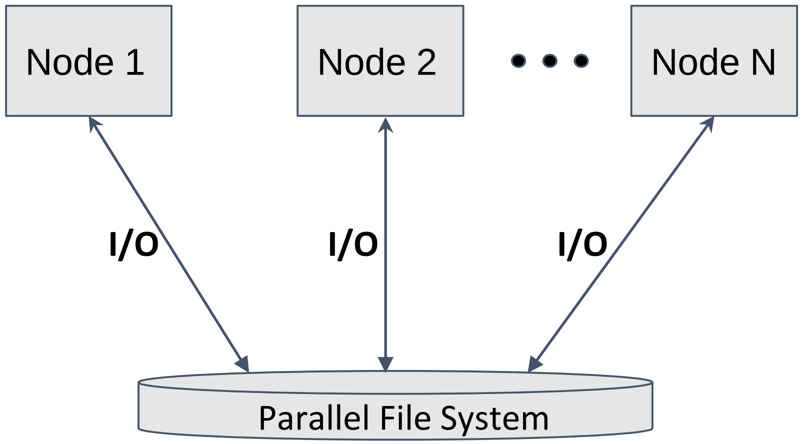
- DL jobs resort to the available shared Parallel File System (PFS) for storing and accessing training data when running in HPC environments.

HPC Shared Storage

- DL jobs resort to the available shared Parallel File System (PFS) for storing and accessing training data when running in HPC environments.
- DL training workloads can cause a lot of storage I/O pressure to the PFS.

HPC Shared Storage

- DL jobs resort to the available shared Parallel File System (PFS) for storing and accessing training data when running in HPC environments.
- DL training workloads can cause a lot of storage I/O pressure to the PFS.
- DL jobs suffer from the performance variability of the PFS.



Motivation

- Node' local resources can lessen the impact of DL jobs.

Motivation

- Node’ local resources can lessen the impact of DL jobs.
    - Users need to manually store the dataset.
    - The dataset might exceed the local storage space making the manual transfer impractical.

Motivation

- Node’ local resources can lessen the impact of DL jobs.
  - Users need to manually store the dataset.
  - The dataset might exceed the local storage space making the manual transfer impractical.
- DL frameworks are aware of the I/O performance problem and provide solutions to suppress them, such as optimized data formats and data loading pipelines.
- Local resources can go unnoticed or even poorly used (*i.e.,* TensorFlow’s *tf.data.Dataset.cache*) by the DL frameworks.

Experimental Results

Setup:

- Single Frontera compute node (4 GPUs).
- 128 GiB of RAM (reduced to 68 GiB) and a 119 GiB SSD partition.

Dataset: ImageNet-1k truncated to 100 GiB and converted to the TFRecord format.

Models: LeNet, AlexNet and ResNet-50.

DL framework: TensorFlow with I/O parallelism, prefetching and parallel preprocessing optimizations enabled.

Experimental Results

Setup:

- Single Frontera compute node (4 GPUs)
- 128 GiB of RAM (reduced to 68 GiB) and a 119 GiB SSD partition.

Dataset: ImageNet-1k truncated to 100 GiB in the TFRecord format

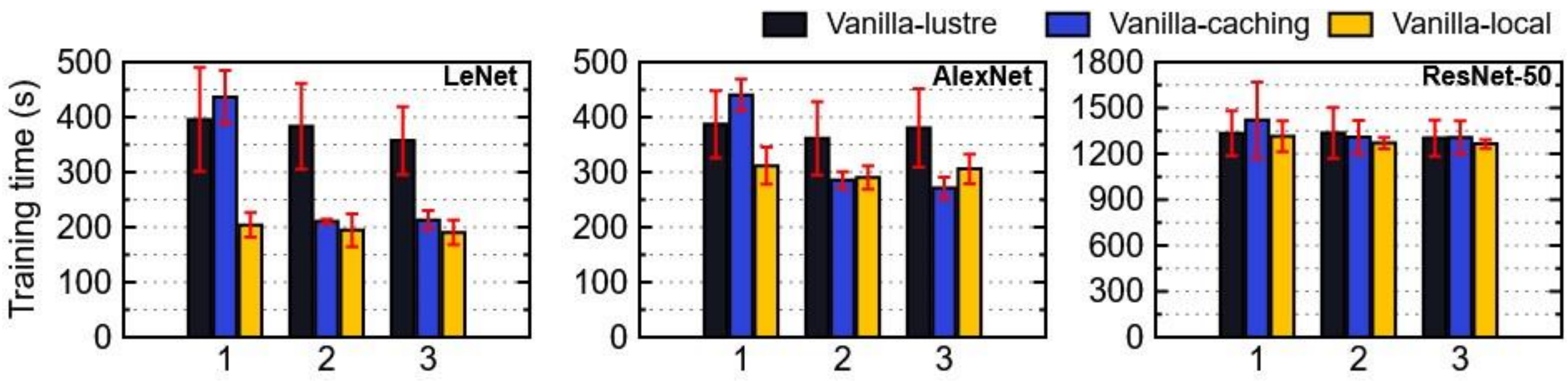
Models: LeNet, AlexNet and ResNet-50

DL framework: TensorFlow with I/O parallelism, prefetching and parallel preprocessing optimizations enabled.

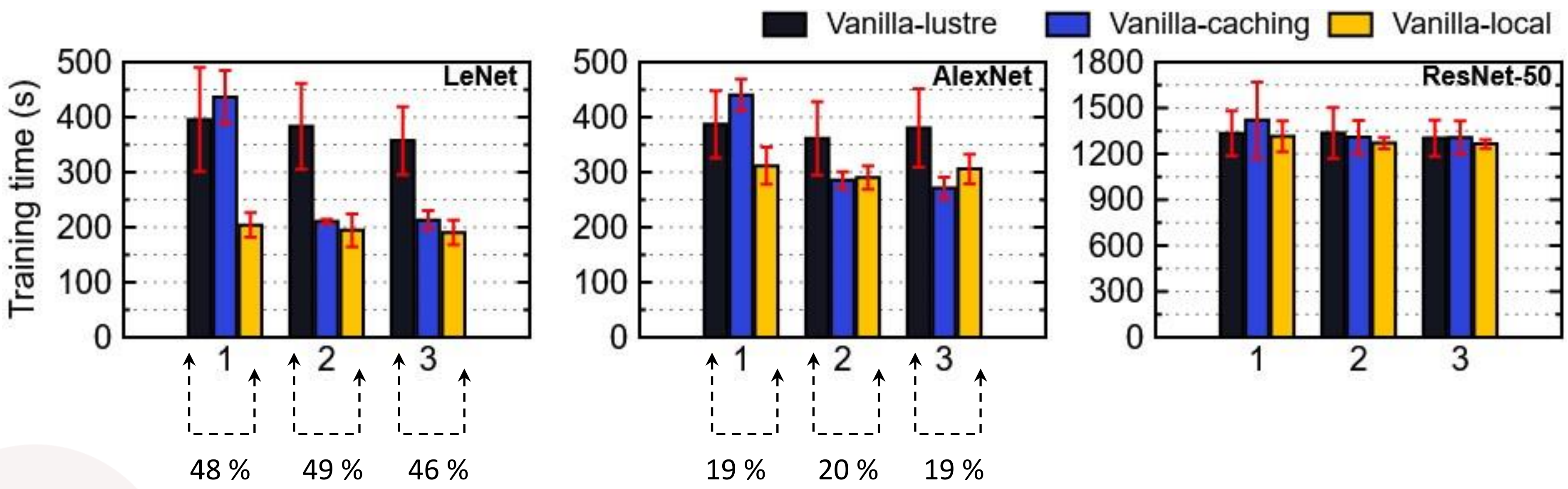
Scenarios:

- **Vanilla-lustre -** Data samples are read entirely from remote storage (Lustre PFS).
- **Vanilla-local -** Data samples are read entirely from local storage (SSD).
- **Vanilla-caching -** Data samples are served from Lustre on the first epoch and written to local storage. On the next epochs data samples are read from local storage.

Experimental Results

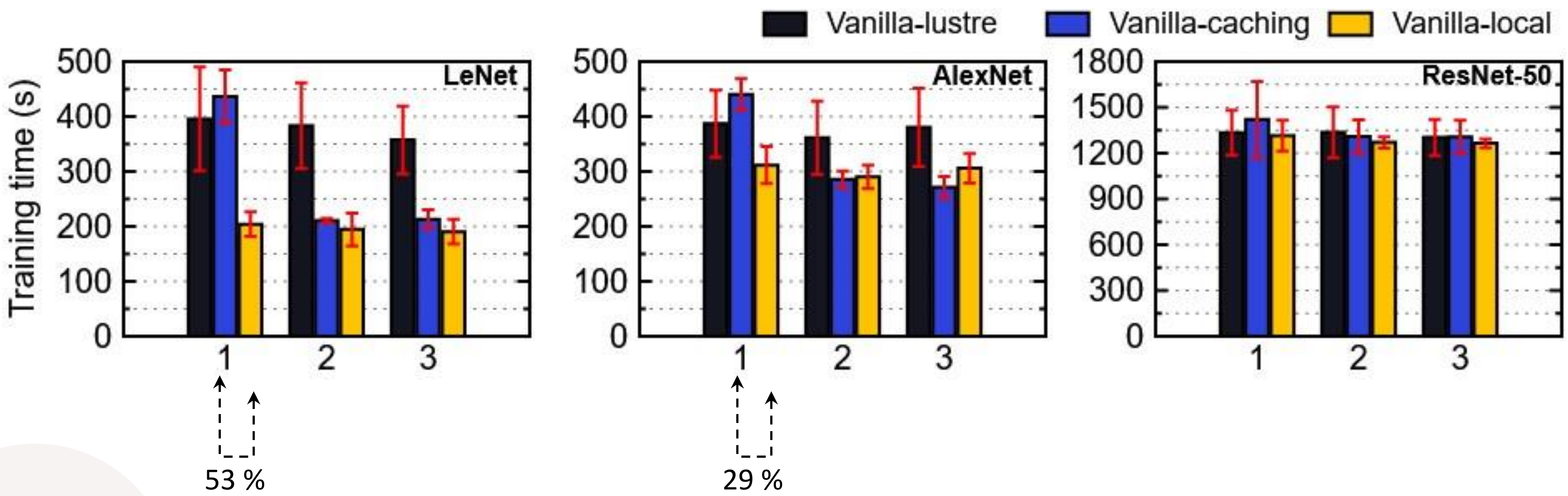


Experimental Results



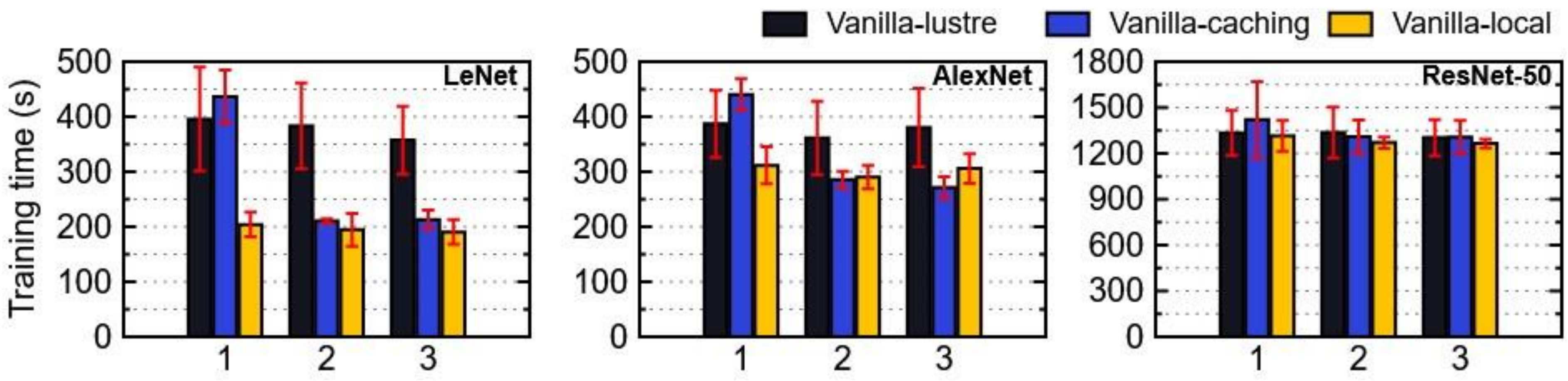
- **I/O bound models** training times are reduced when reading from local storage.

Experimental Results



- **I/O bound models** training times are reduced when reading from local storage.
- Caching mechanisms **might increase training time** on the placement epoch.

Experimental Results



- **I/O bound models** training times are reduced when reading from local storage.
- Caching mechanisms **might increase training time** on the placement epoch.
- Compute bound models, such as ResNet-50, **do not gain from improved I/O performance.**

Related Work

- Data loading and preprocessing solutions improve reading efficiency (*Dali*, *CoorDL*), but **fail to leverage local storage**.
- Staging techniques that are **highly focused on distributed training** (*Fanstore*, *Diesel*) can enforce the use of unnecessary resources.
- Existing storage tiering solutions also take advantage of framework specific semantics to improve training performance (*Quiver*, *NoPFS*), but **lead to less portable and more intrusive solutions**.

MONARCH

Objectives

- Transparently manage local storage mediums to cache datasets.
- Have a portable solution applicable over many DL frameworks.
- Make a solution that is less intrusive, thus more applicable for the HPC environment.

MONARCH

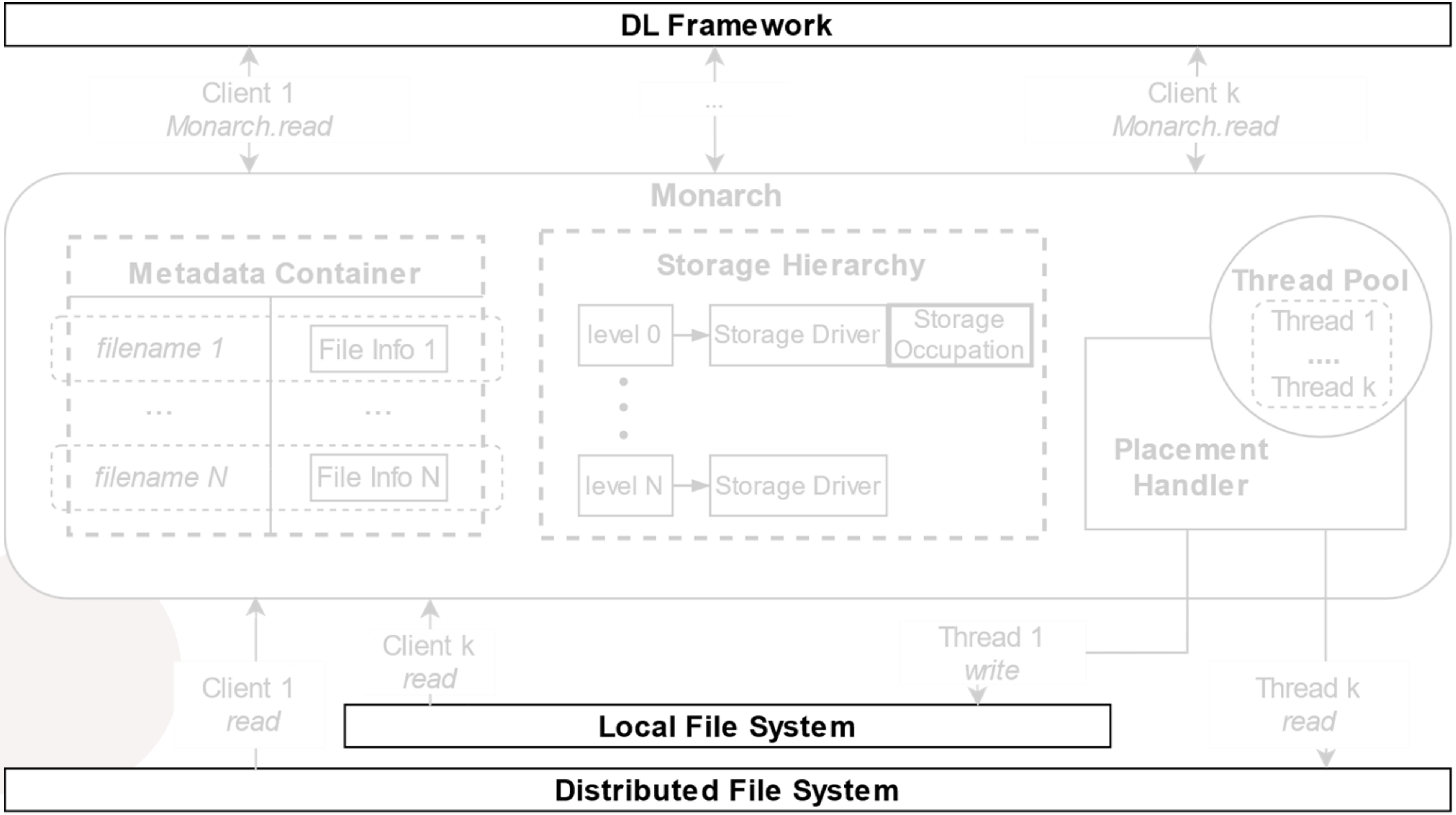
Objectives

- Transparently manage local storage mediums to cache datasets.
- Have a portable solution applicable over many DL frameworks.
- Make a solution that is not intrusive, thus more applicable for the HPC environment.

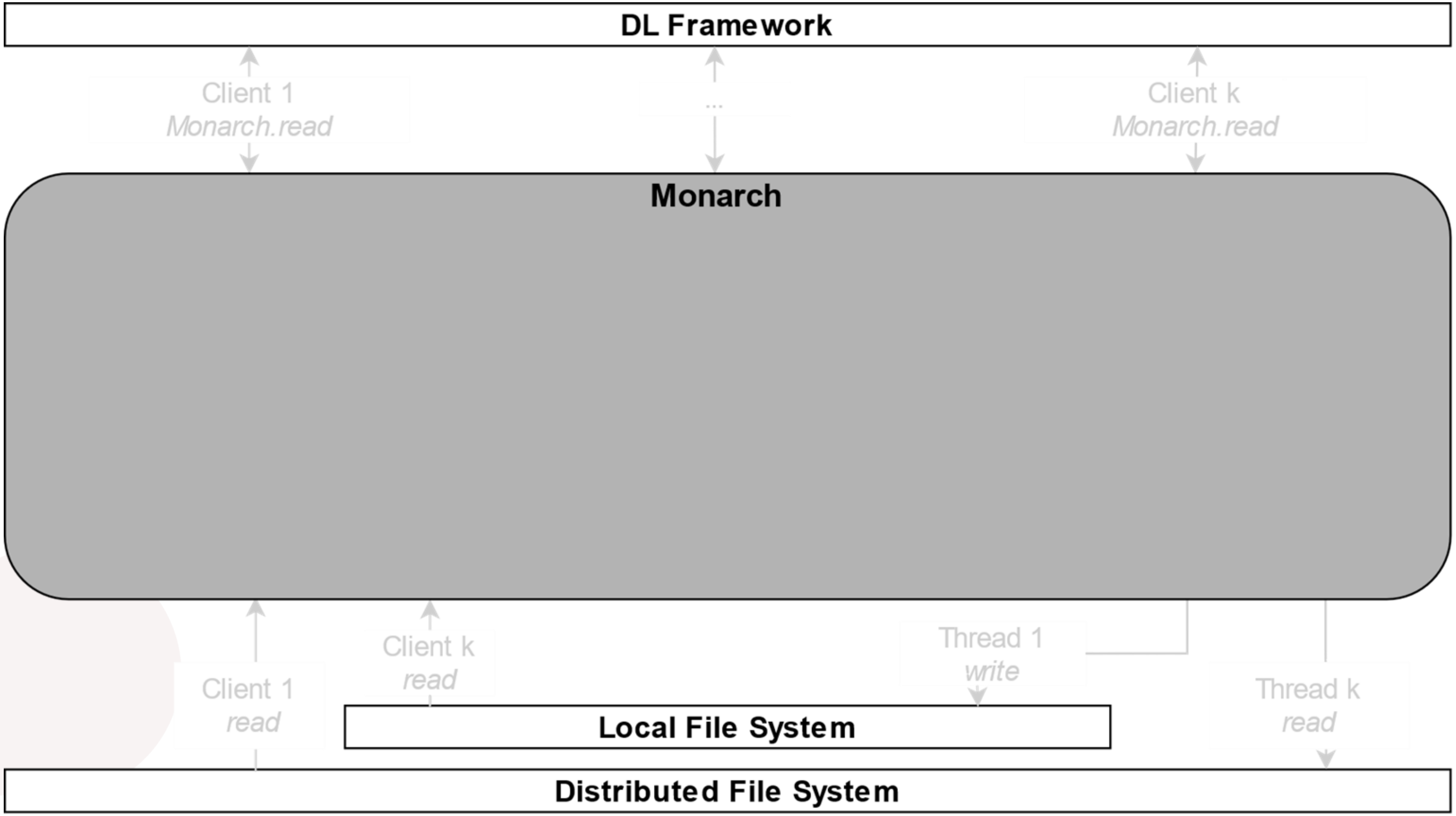
Contributions

- An experimental study demonstrating the performance impact of running DL jobs under the Lustre PFS and the compute nodes' local storage.
- MONARCH, a novel storage middleware that mediates I/O requests between DL frameworks and HPC storage resources.
- An early prototype of Monarch and its integration with TensorFlow.
    - https://github.com/dsrhaslab/monarch
- An experimental evaluation of the prototype.

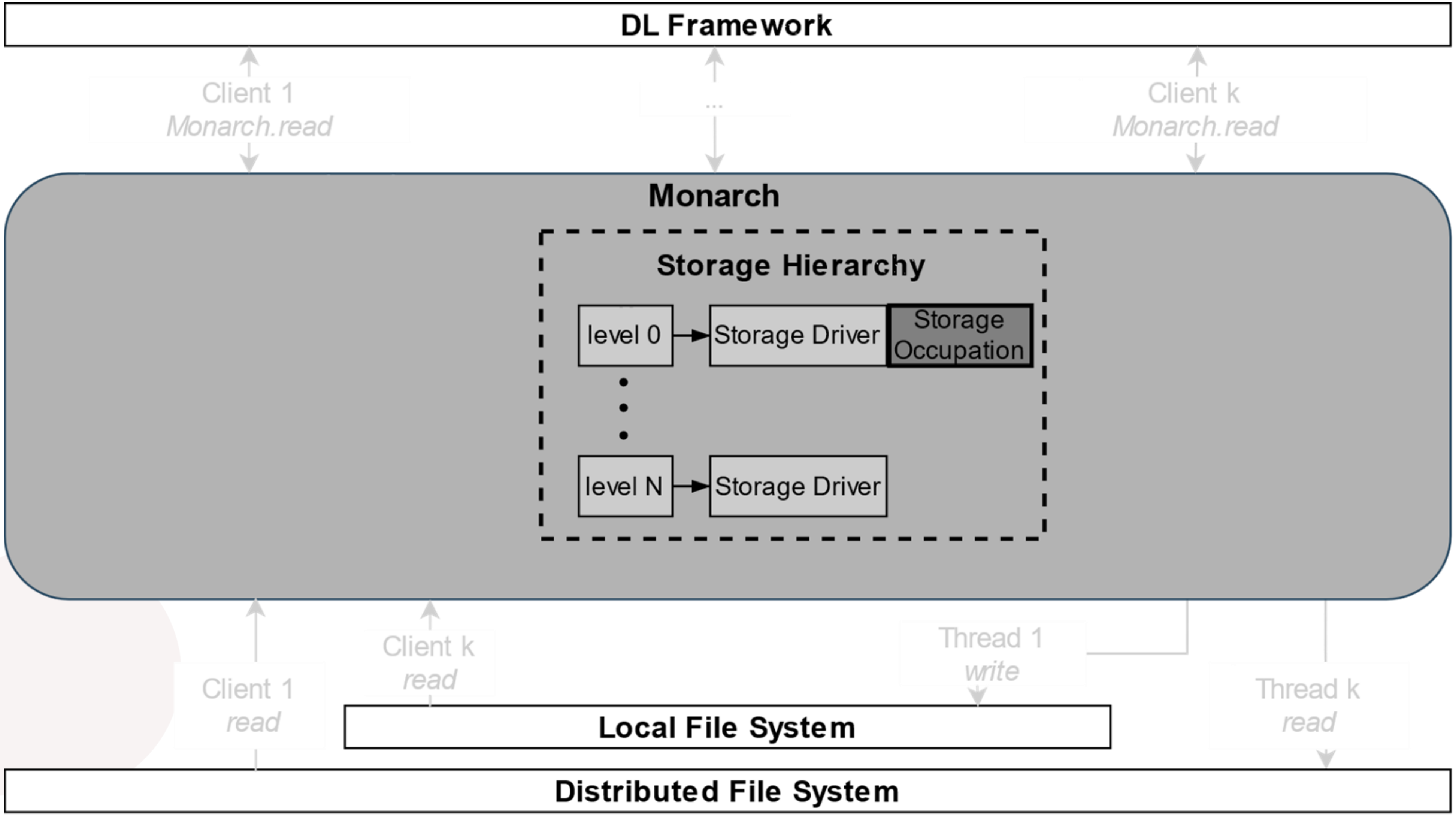
MONARCH


DL Framework
Client 1
Monarch.read
...
Client k
Monarch.read
Monarch
Metadata Container
filename 1
File Info 1
...
...
filename N
File Info N
Storage Hierarchy
level 0
Storage Driver
Storage Occupation
level N
Storage Driver
Thread Pool
Thread 1
....
Thread k
Placement Handler
Client k
read
Client 1
read
Thread 1
write
Thread k
read
Local File System
Distributed File System

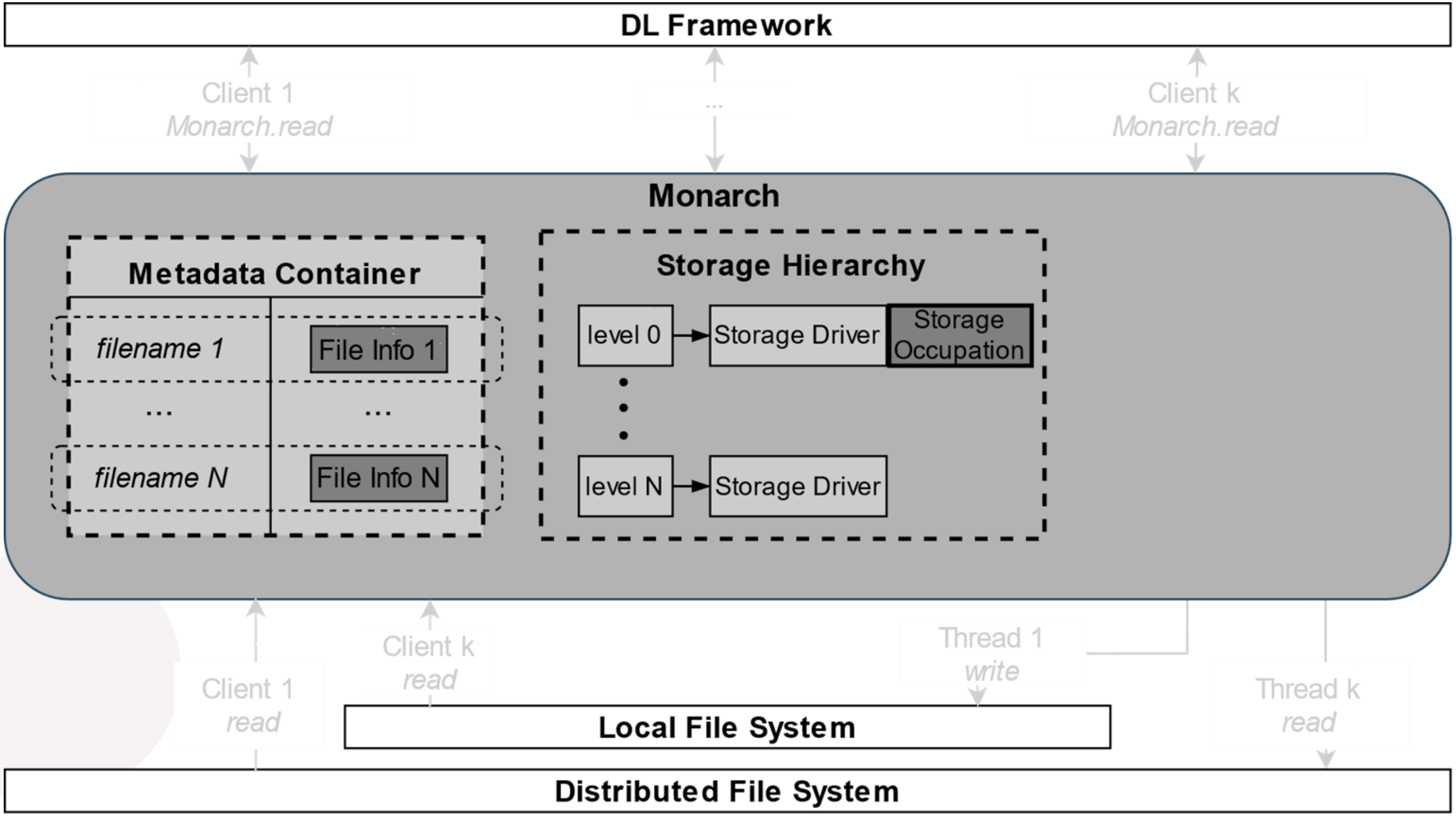
MONARCH Architecture



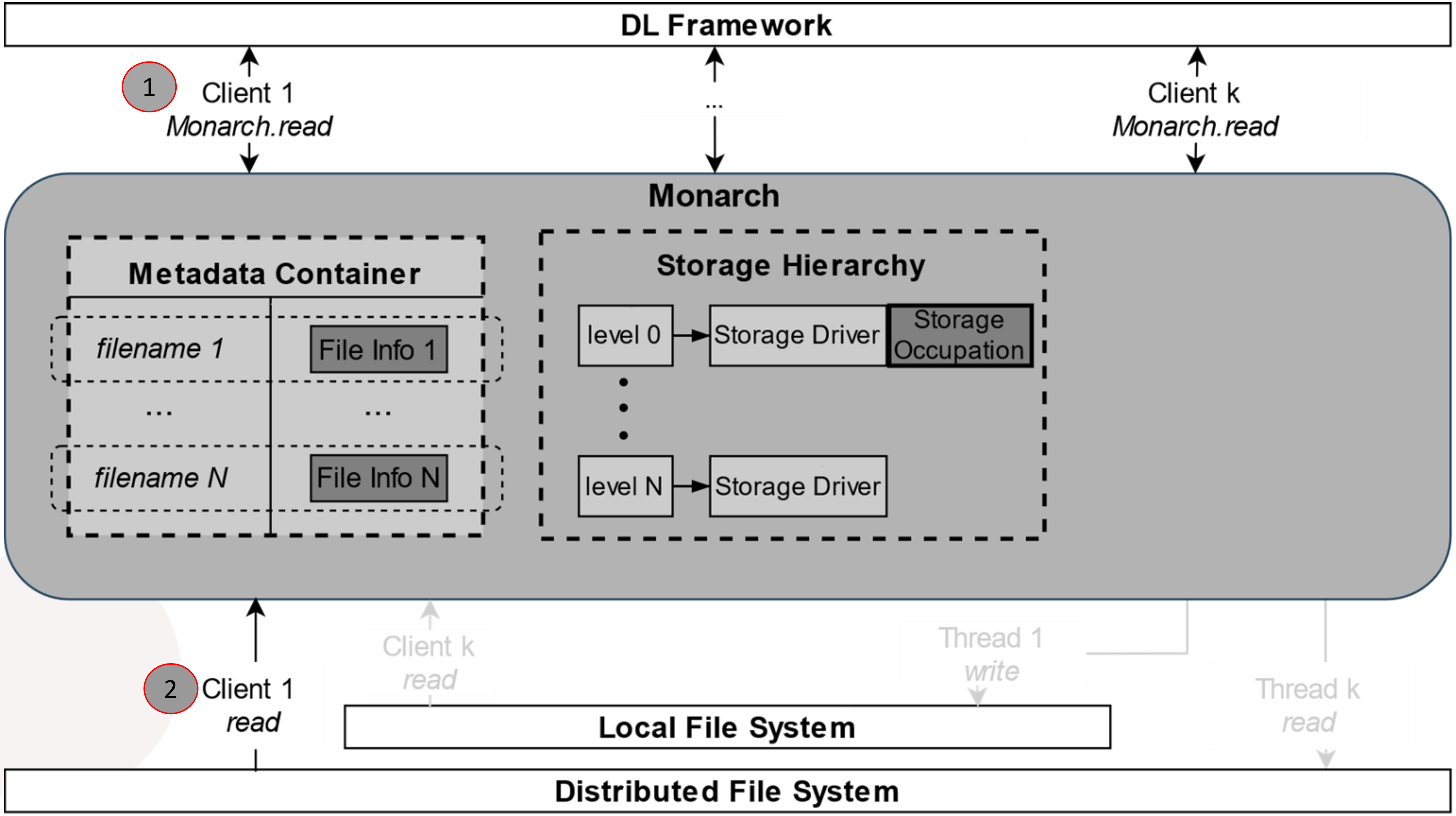
MONARCH Architecture



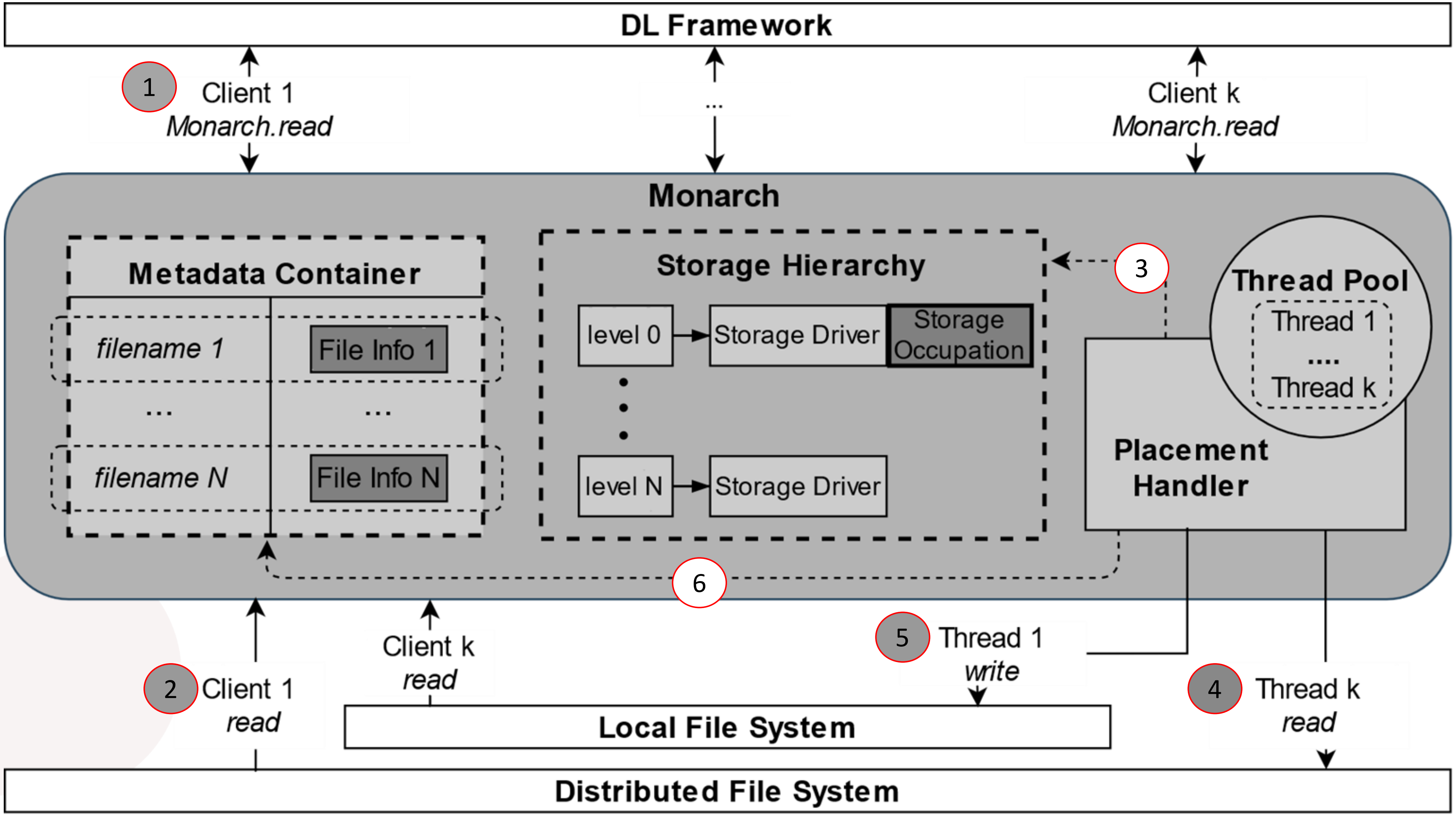
MONARCH Architecture



MONARCH Architecture



MONARCH Architecture



Experimental Evaluation

Setup, Models and DL framework: The same as in “Motivational Results”

Datasets:

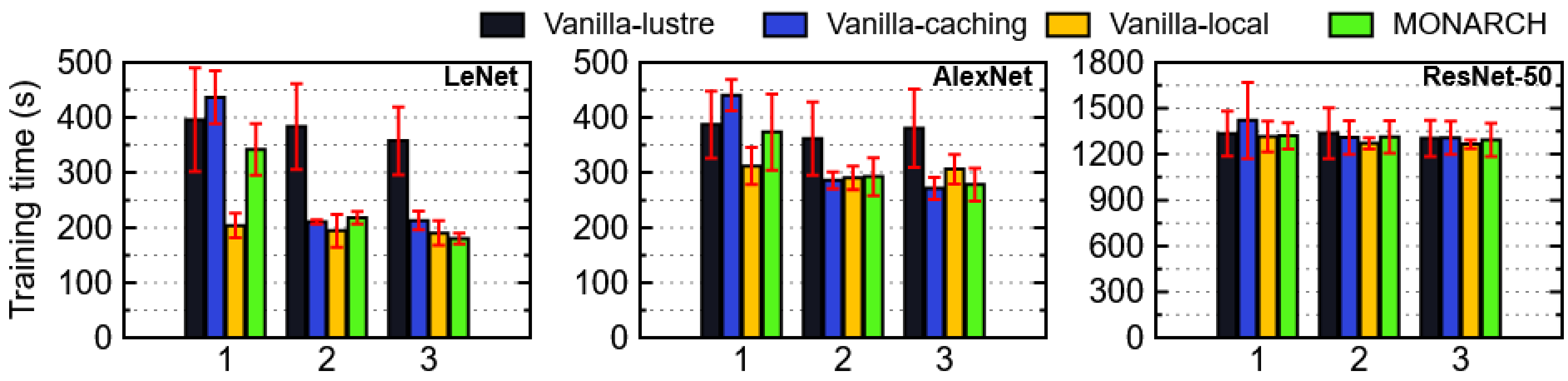
- ImageNet-1k truncated to 100 GiB in the TFRecord format.
- ImageNet-1k expanded to 200 GiB in the TFRecord format.

MONARCH prototype: Made of 1,500 lines of C++ code and its integration with TensorFlow lead to changing 6 lines of code.

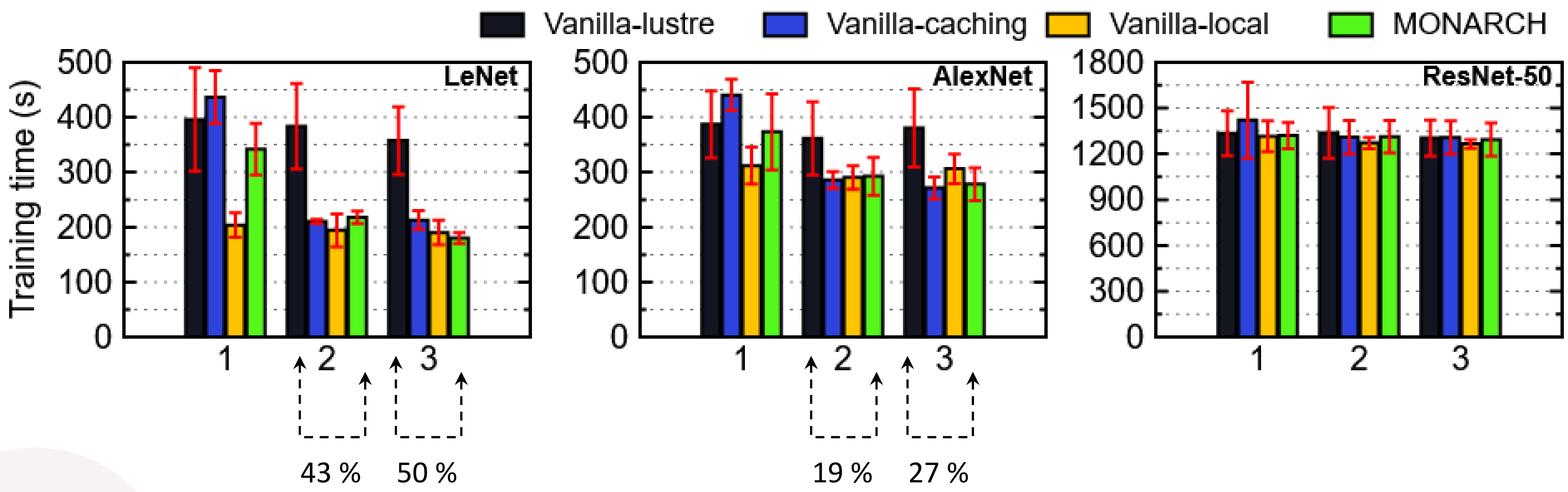
What we want to know:

- Can MONARCH improve training performance for different DL models and dataset sizes?
- Can MONARCH reduce the I/O pressure on the PFS backend?

TensorFlow Experimental Evaluation

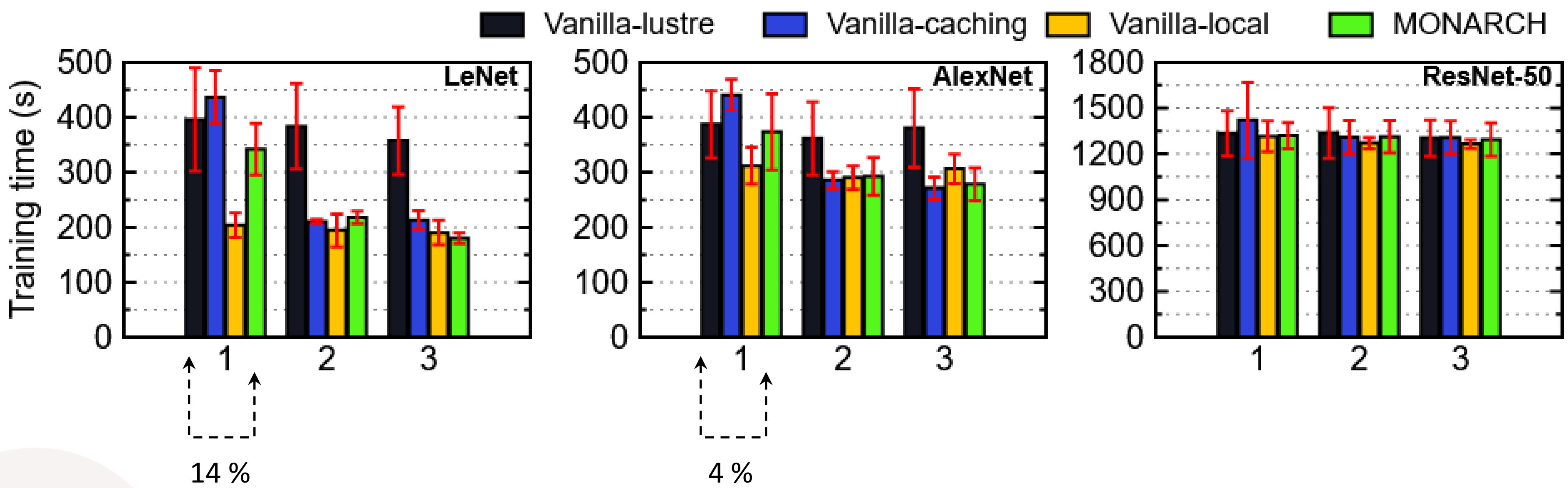


TensorFlow Experimental Evaluation



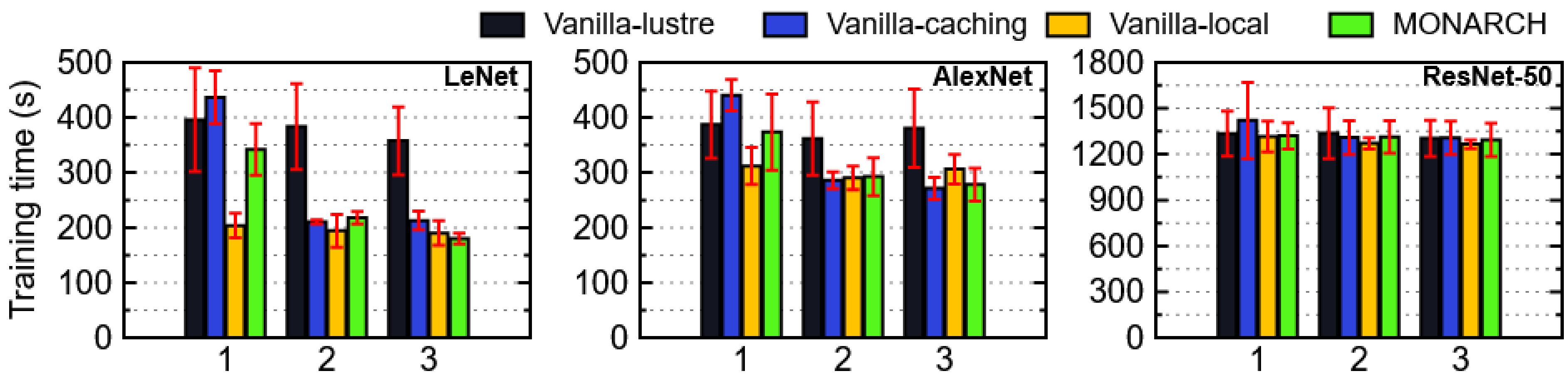
- MONARCH **improves I/O bound models performance**, more predominantly after the first epoch.

TensorFlow Experimental Evaluation



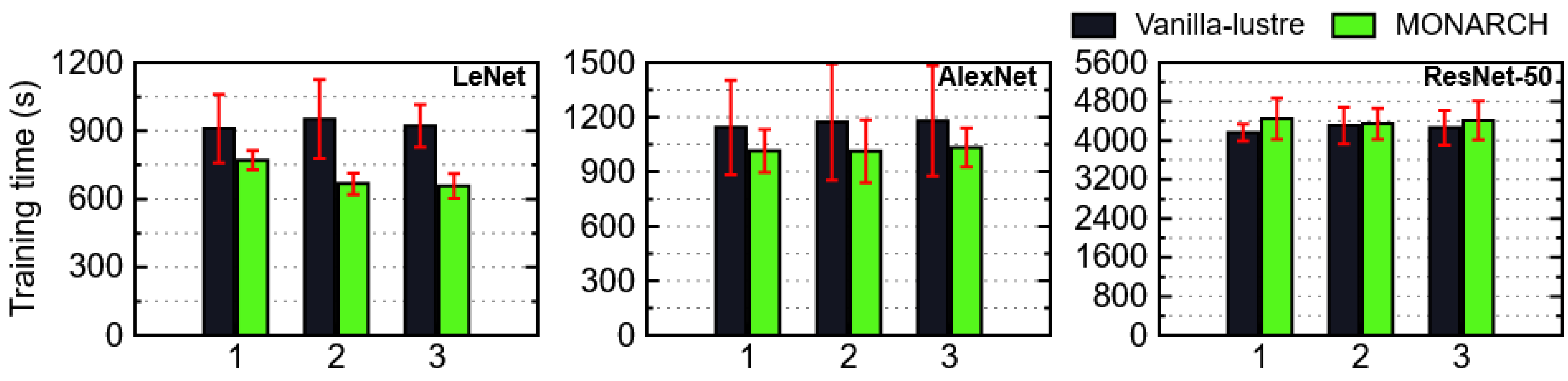
- MONARCH **improves I/O bound models performance**, more predominantly after the first epoch.
- Due to monarch placement strategy, in this setup, **the first epochs have a performance boost**, not incurring in additional overhead for the placement occurring in background.

TensorFlow Experimental Evaluation



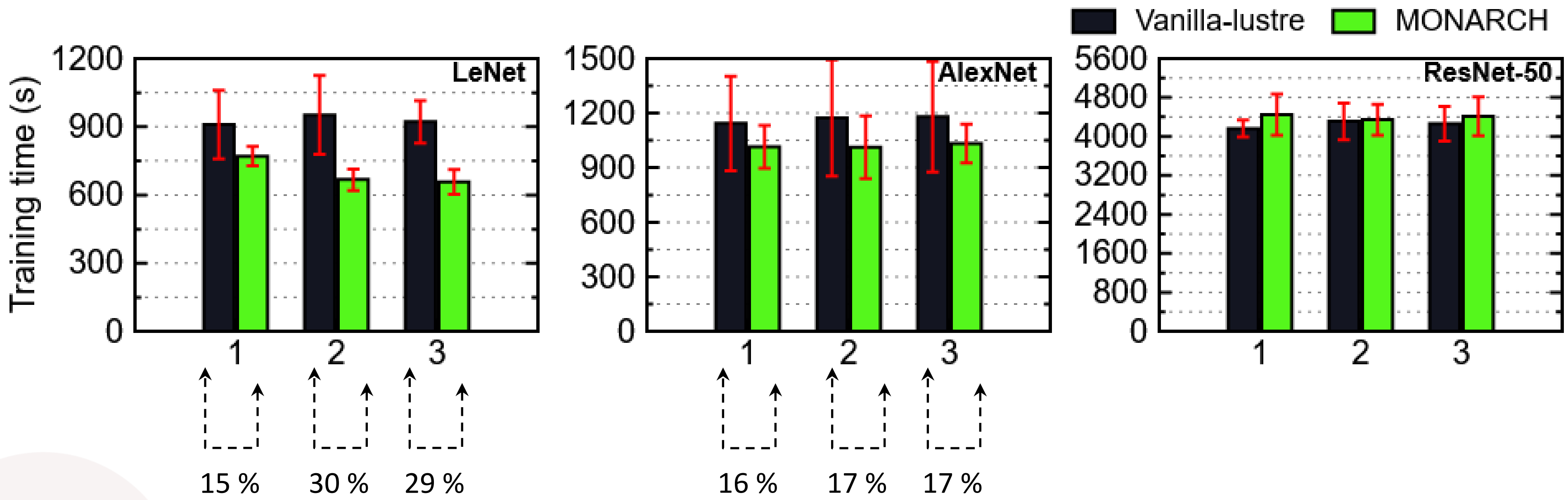
- MONARCH **improves I/O bound models performance**, more predominantly after the first epoch.
- Due to monarch placement strategy, in this setup, **the first epochs have a performance boost**, not incurring in additional overhead for the placement occurring in background.
- Similar to the other scenarios, MONARCH maintains the ResNet-50 training performance.

TensorFlow Experimental Evaluation



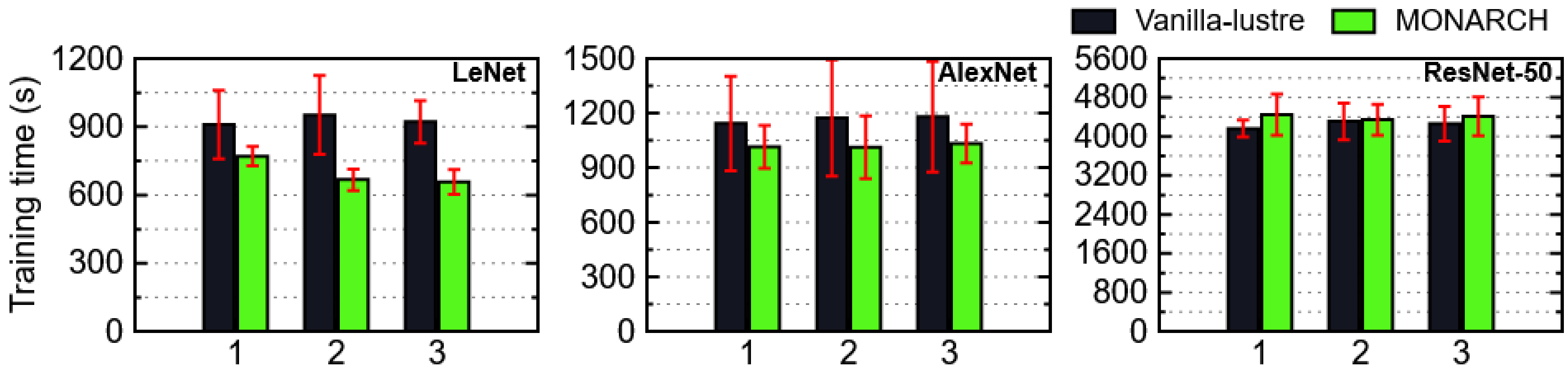
- MONARCH only caches ≈ **56 %** of the dataset.

TensorFlow Experimental Evaluation



- MONARCH only caches ≈ **56 %** of the dataset.
- MONARCH continues to **improve I/O bound models performance.**

TensorFlow Experimental Evaluation



- MONARCH only caches ≈ **56 %** of the dataset.
- MONARCH continues to **improve I/O bound models performance.**
- For the ResNet-50 model **the impact of caching the dataset is unnoticed**.

Conclusions and Future Work

Preliminar results, resorting to various models and dataset sizes, show that MONARCH-enabled TensorFlow can speed up DL training and reduce I/O pressure on the shared PFS backend.

As future work:

- Additional experiments.
- Consider more storage layers.
- Distributed Training.

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads

MONARCH: Hierarchical Storage Management for Deep Learning Frameworks

Marco Dantas, Diogo Leitão, Cláudia Correia, Ricardo Macedo, Weijia Xu*, João Paulo
INESC TEC & University of Minho, *Texas Advanced Computing Center

September 7, 2021

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

INESCTEC
TECNOLOGIA E CIÊNCIA
LABORATÓRIO ASSOCIADO

Universidade do Minho