THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

商汤 sensetime

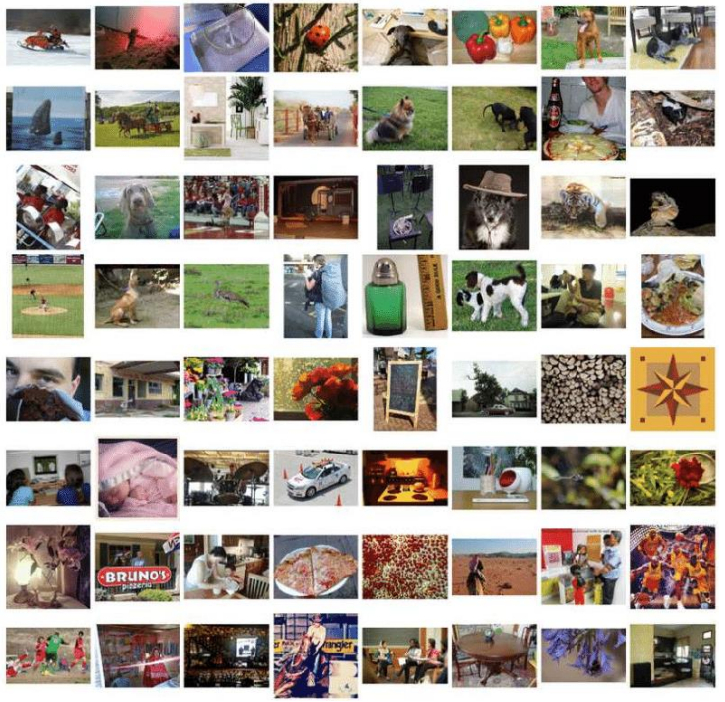
DIESEL: A Dataset-Based Distributed Storage and Caching System for Large-Scale Deep Learning Training

Lipeng Wang[1], Songgao Ye[2], Baichen Yang[1], Youyou Lu[3], Hequan Zhang[2], Shengen Yan[2], and Qiong Luo[1]

[1]Hong Kong University of Science and Technology
[2]SenseTime Research
[3]Tsinghua University

Deep Learning training (DLT): an important workload on clusters

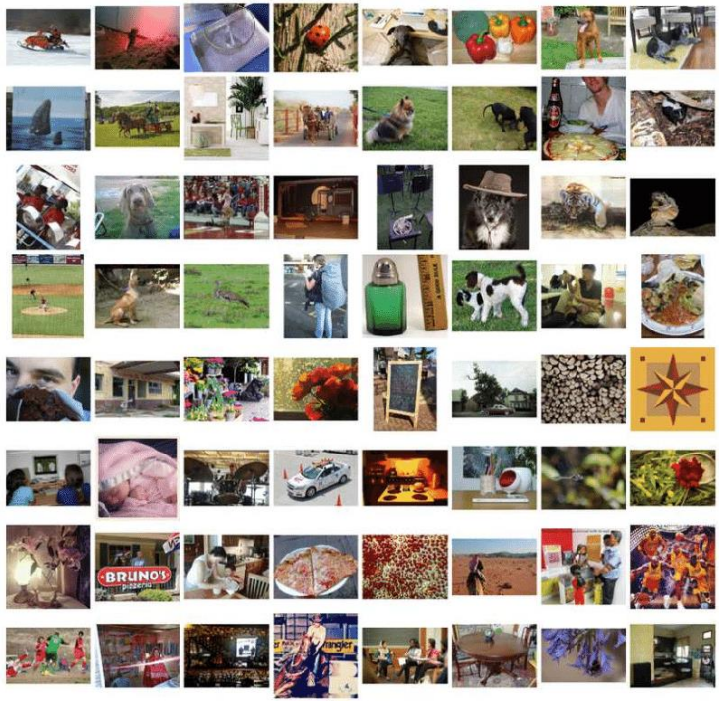
- Widely deployed in many areas
  - Image Classification
  - Object Detection
  - Natural Language Processing
  - Recommender Systems
- Data intensive
  - ImageNet-1K:
    - 1.28 million images
  - Open Image:
    - 9 million images
- Expensive accelerators, i.e., GPUs



Training the well-known ResNet-50 model on the ImageNet-1K dataset takes more than 30 hours in a cluster

Deep Learning training (DLT): an important workload on clusters

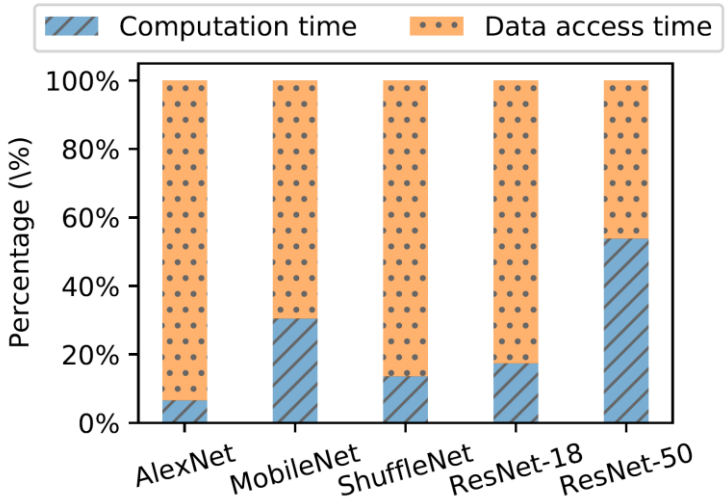
- Widely deployed in many areas
  - Image Classification
  - Object Detection
  - Natural Language Processing
  - Recommender Systems
- Data intensive
  - ImageNet-1K:
    - 1.28 million images
  - Open Image:
    - 9 million images
- Expensive accelerators, i.e., GPUs



Training the well-known ResNet-50 model on the ImageNet-1K dataset takes more than 30 hours in a cluster

How to reduce the total training time?

File size distribution and training time breakdown



The data access time takes a significant part in the total training time

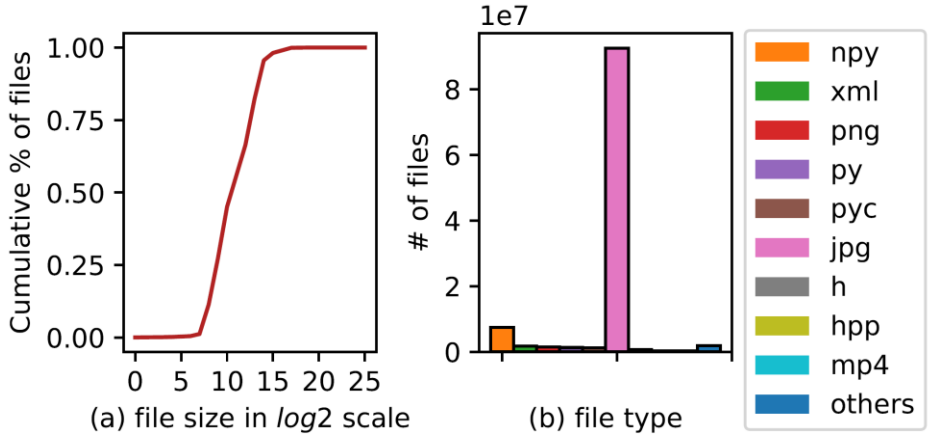
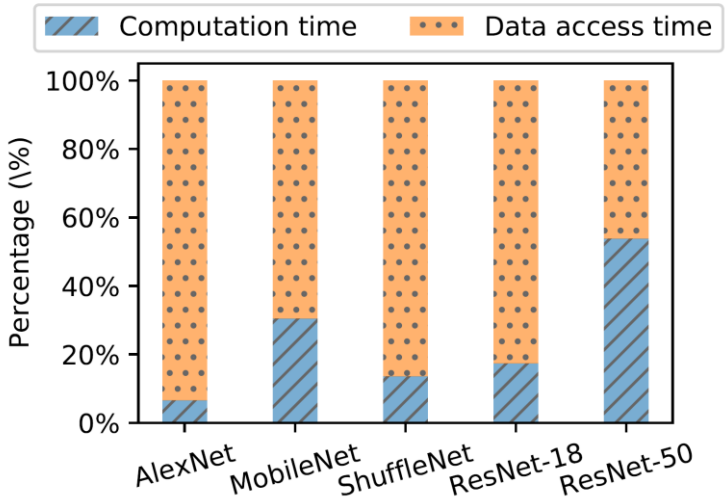


Image size and type distribution on a real-world production cluster:

- Most files are smaller than 128KB

File size distribution and training time breakdown



The data access time takes a significant part in the total training time

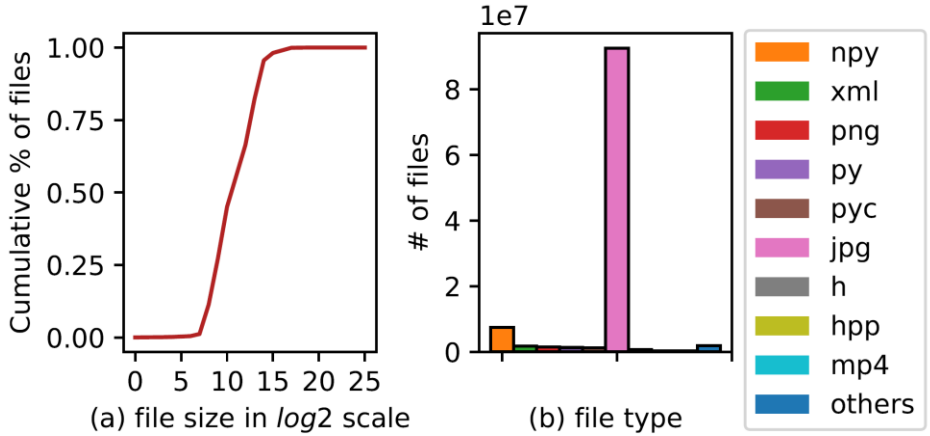


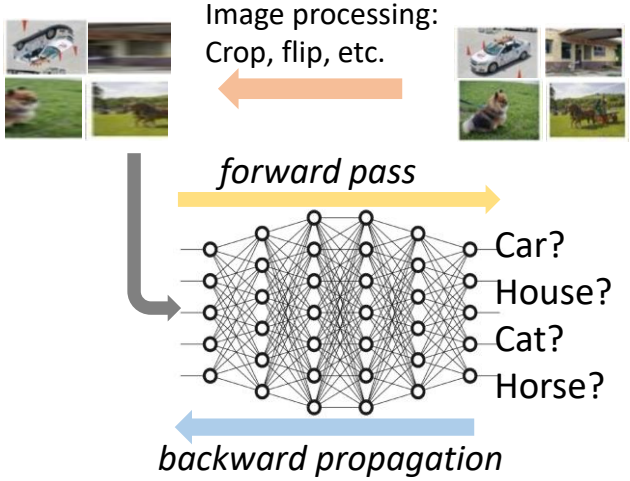
Image size and type distribution on a real-world production cluster:

- Most files are smaller than 128KB

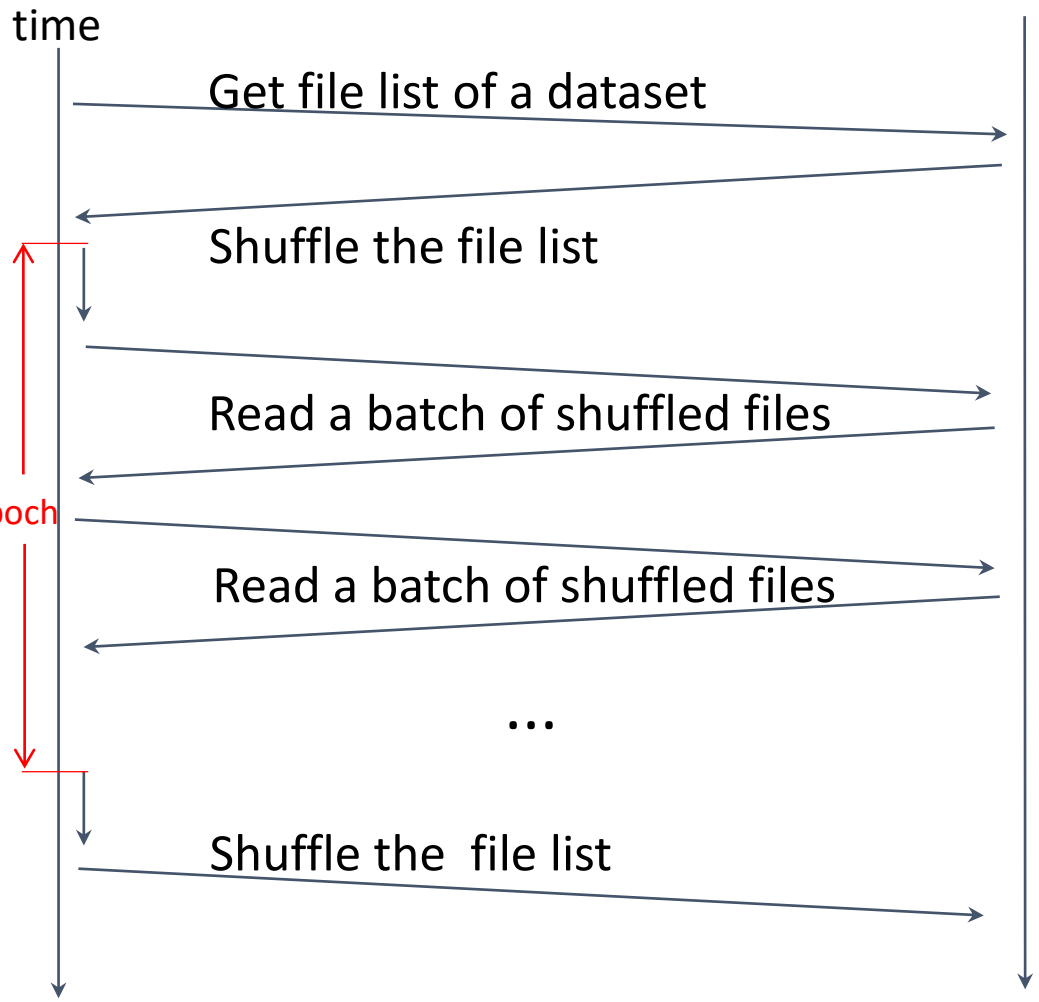
Reduce the data access time!

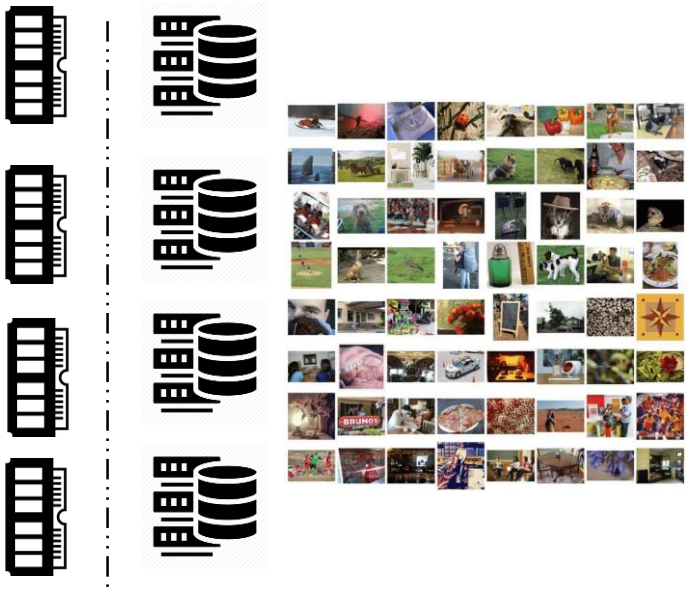
File access procedure in DLT tasks on computer clusters

mxnet PYTORCH TensorFlow theano Caffe2 Microsoft CNTK



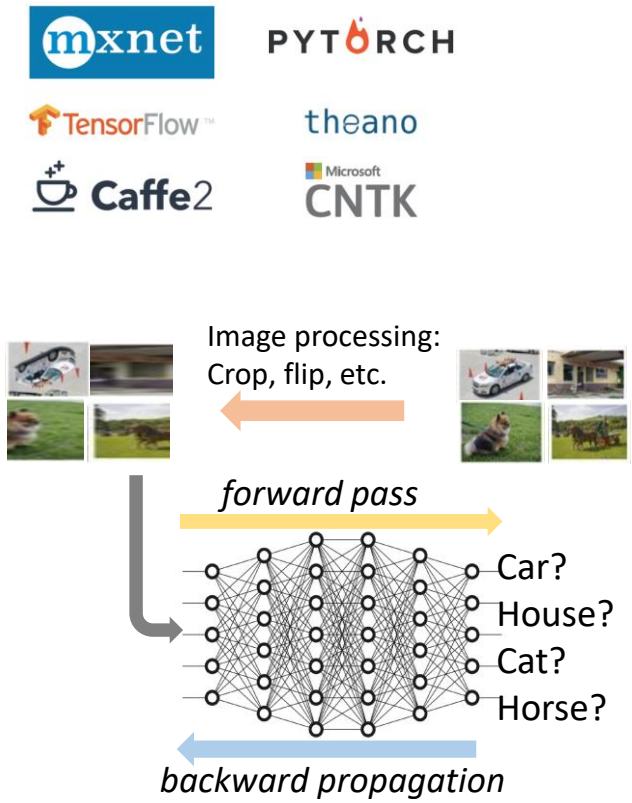
Training framework



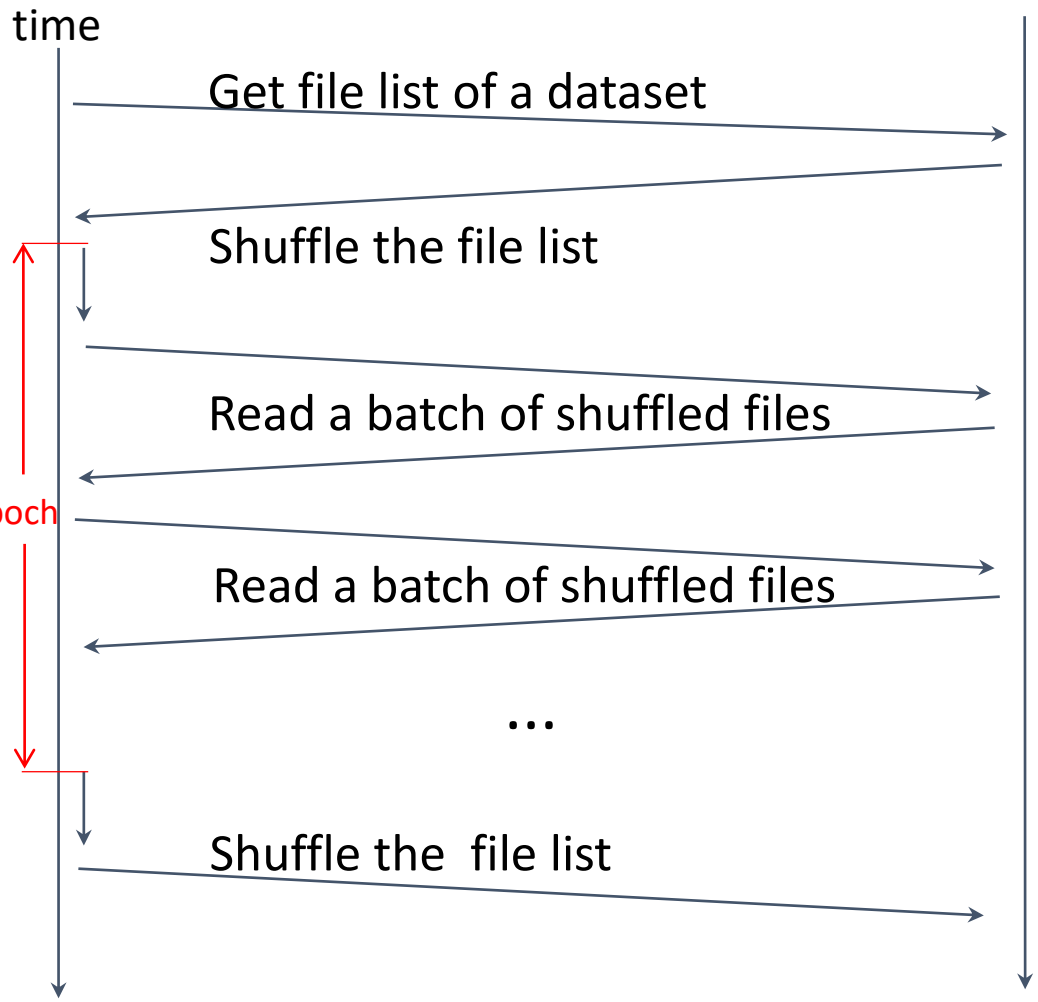


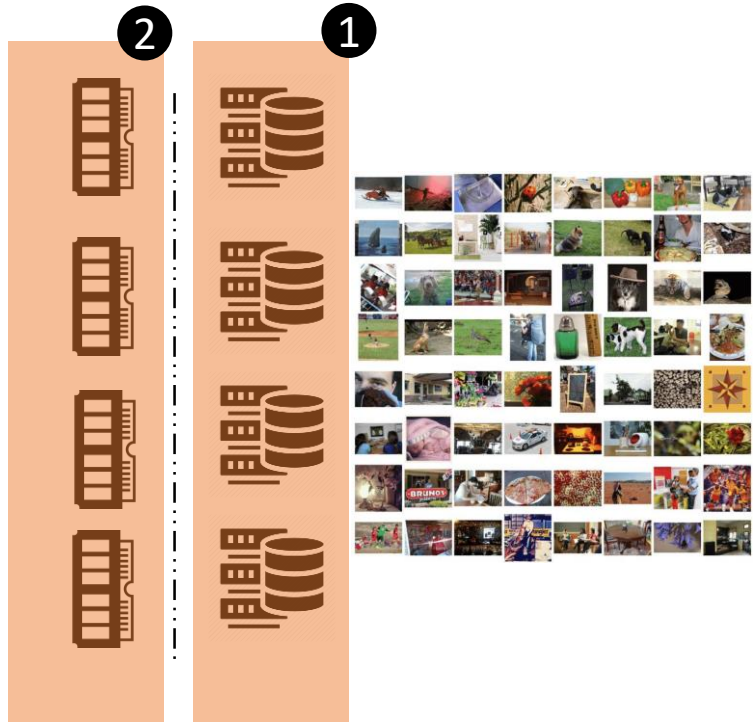
Distributed storage and cache

File access procedure in DLT tasks on computer clusters



Training framework





Distributed storage and cache

Three problems in existing storage and caching systems

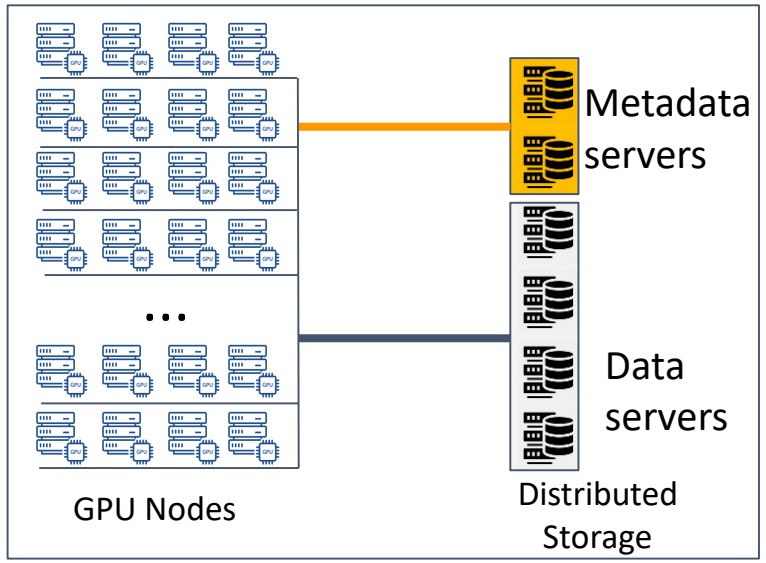
P1: Large number of small files → Metadata access is not scalable on existing systems

P2: Node failure in global cache → Affects all DLT tasks on a cluster & slow to recover

P3: Shuffled file access pattern → Slow read speed

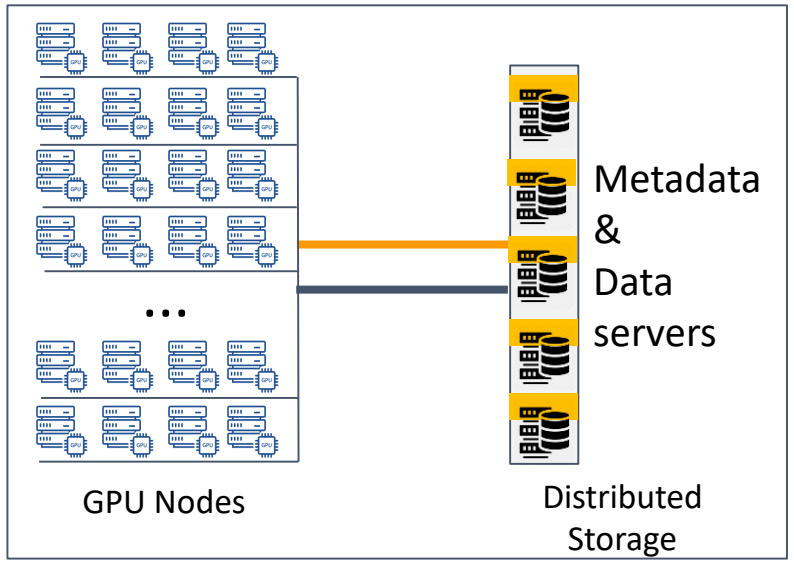
Problem 1: metadata access does not scale on existing systems

Alternative 1

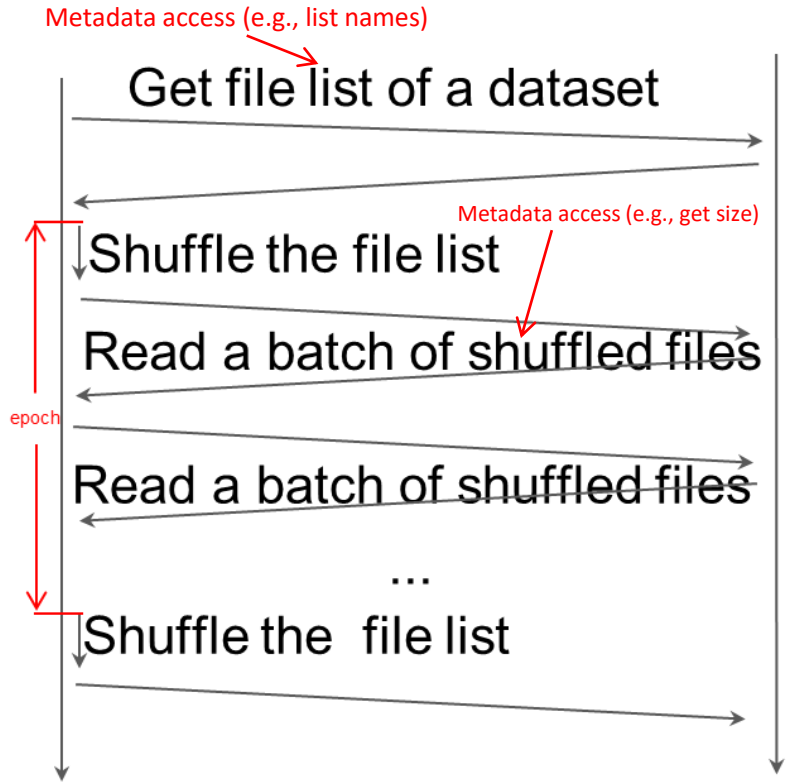


Separated metadata servers and data servers
(e.g., Lustre, GFS)

Alternative 2

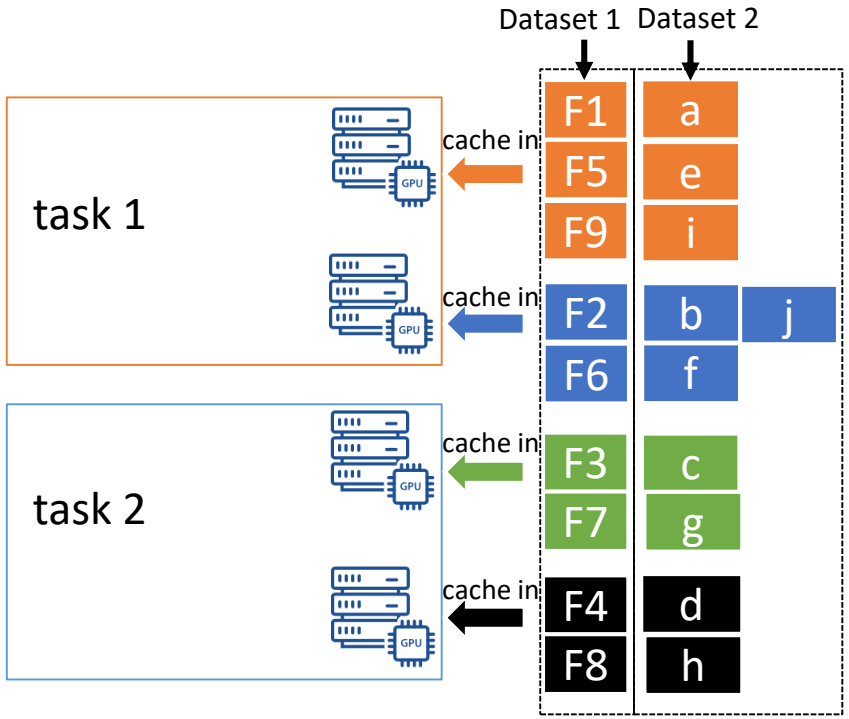


Distribute metadata and data on all storage servers
(e.g., Ceph, GlusterFS)



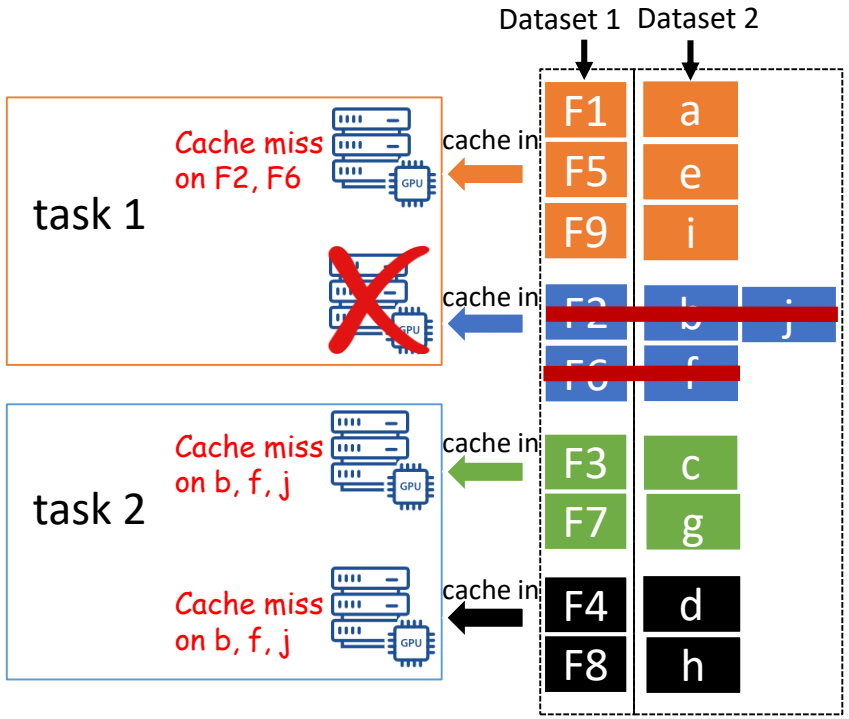
- Existing storage systems have poor scalability on metadata access

Problem 2: global caching systems are vulnerable to node failures



- Task 1 works on dataset 1
- Task 2 works on dataset 2

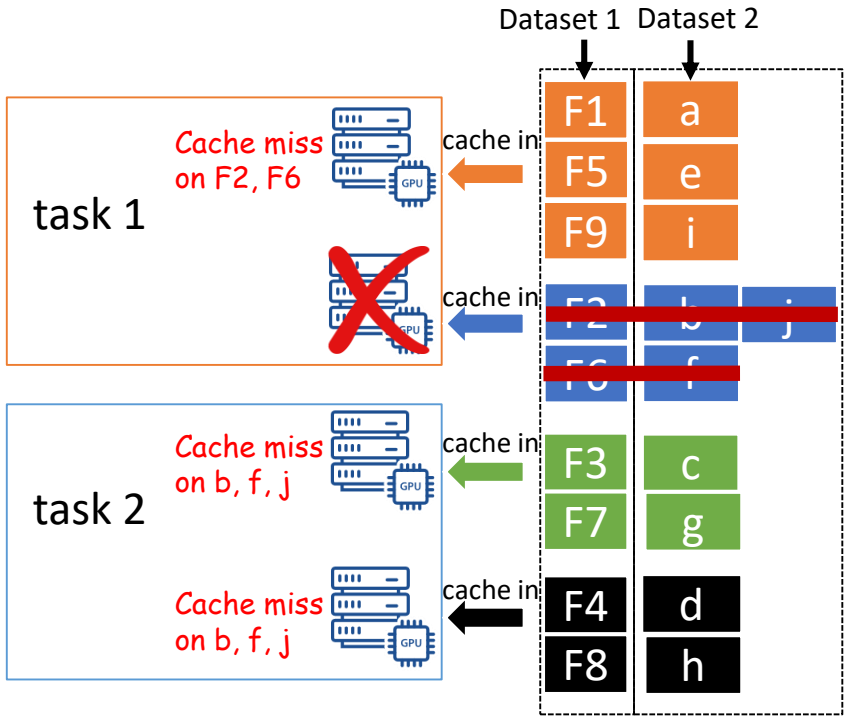
Problem 2: global caching systems are vulnerable to node failures



- Task 1 works on dataset 1
- Task 2 works on dataset 2

- A node failure in task 1 will affect both task 1 and task 2!

Problem 2: global caching systems are vulnerable to node failures

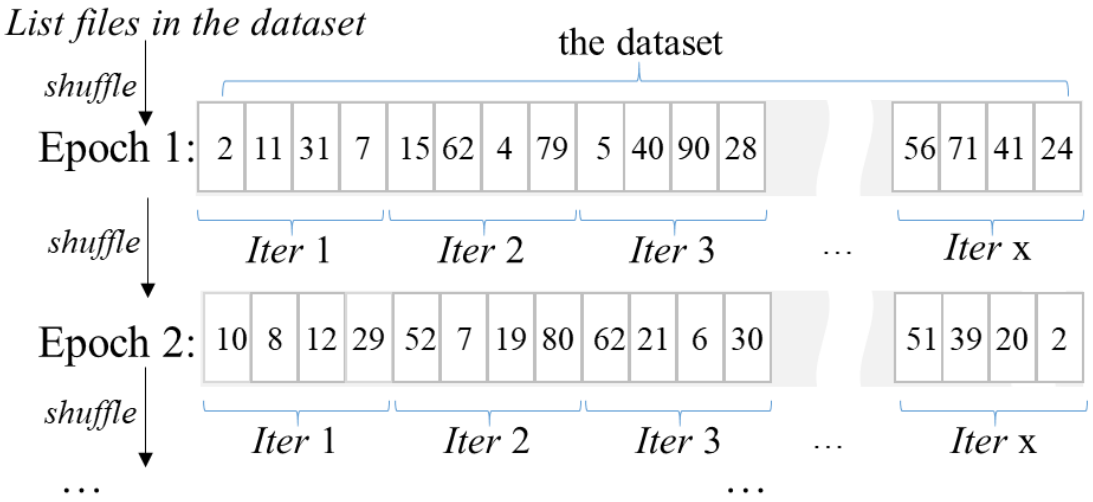


- Task 1 works on dataset 1
- Task 2 works on dataset 2

- A node failure in task 1 will affect both task 1 and task 2!
- The cache node recovery takes a long time due to small file reads!

Problem 3: shuffled access of small files is slow

file access pattern in DLT tasks



read speed comparison on different read unit size

File Size(KB)	Bandwidth(MB)	Files/Second	4K-IOPS
1	33.54	34353.45	8588.36
4	128.28	32841.47	32841.47
16	464.44	29724.48	118897.92
64	1317.04	21072.64	337162.24
256	2725.93	10903.72	697838.08
1024	3104.26	3104.26	794690.56
4096	3197.68	799.42	818606.08

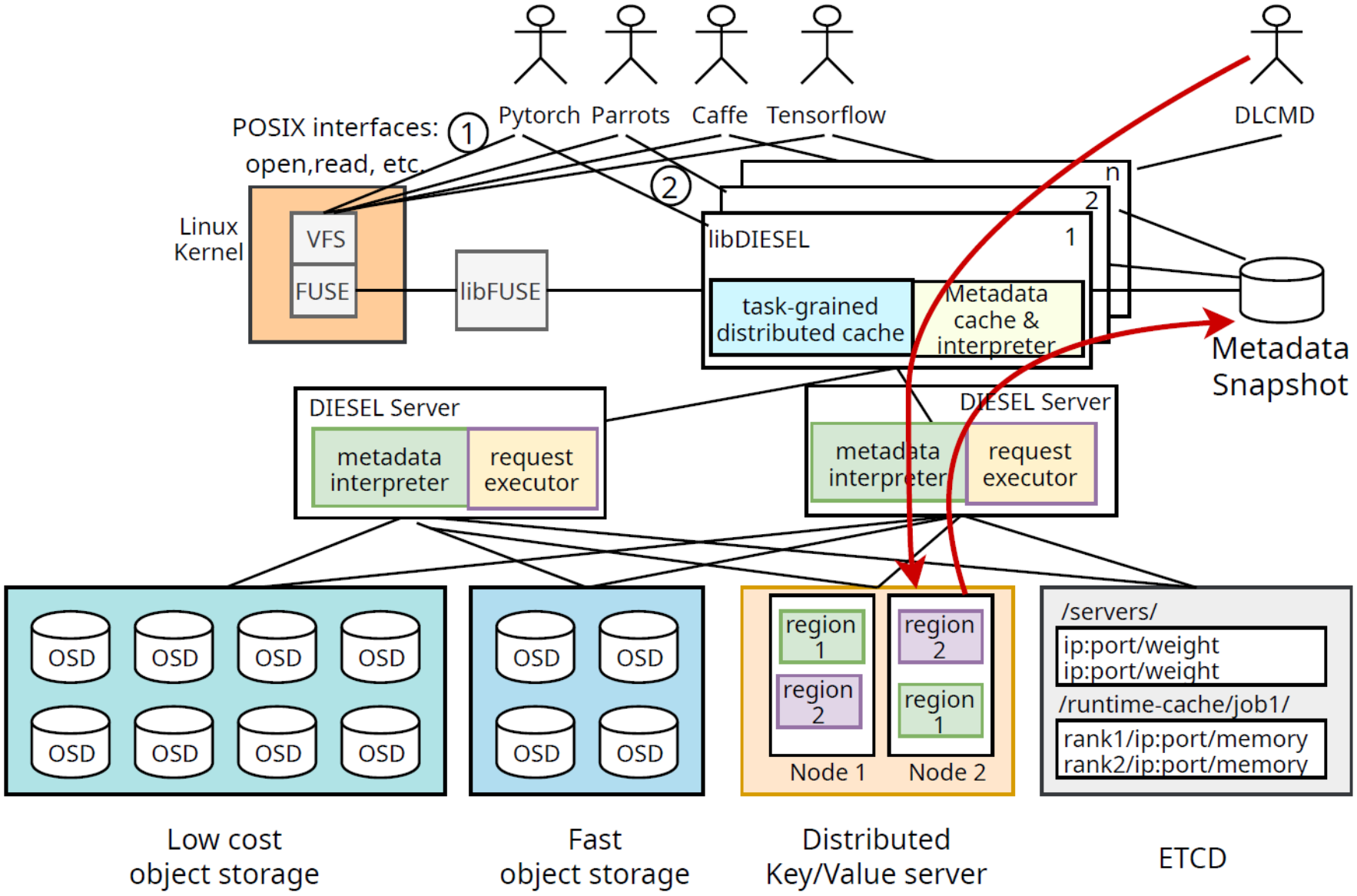
~25x

- The shuffled access pattern on small files hurts the read performance a lot

Proposed solutions in DIESEL

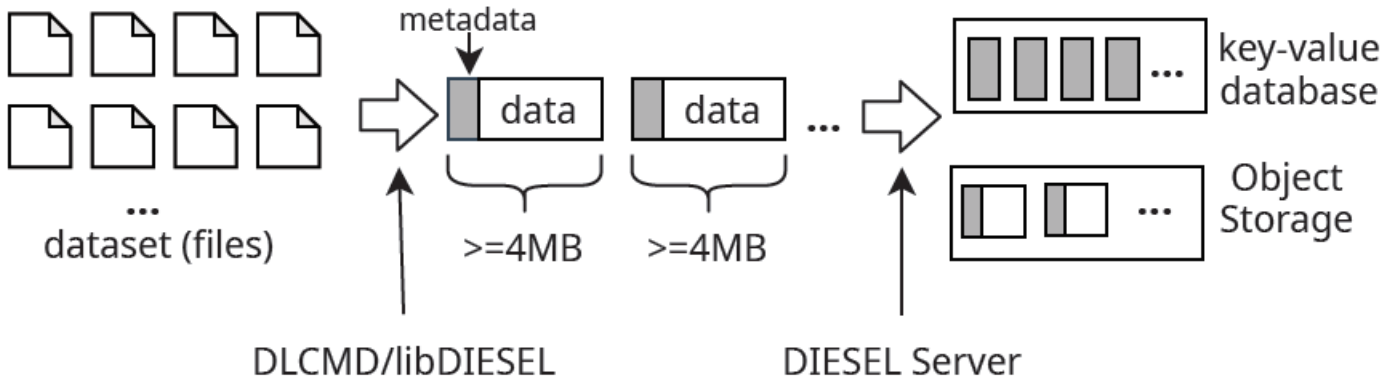
P1: Large number of small files	→	Metadata access is not scalable on existing systems	→ Distributed in-memory metadata server & metadata snapshot ❶
P2: Node failure in global cache	→	Affects all DLT tasks & slow to recover	→ Task-grained caching system ❷
P3: Shuffled file access pattern	→	Slow read speed	→ Chunk-based shuffle method ❸

DIESEL overview



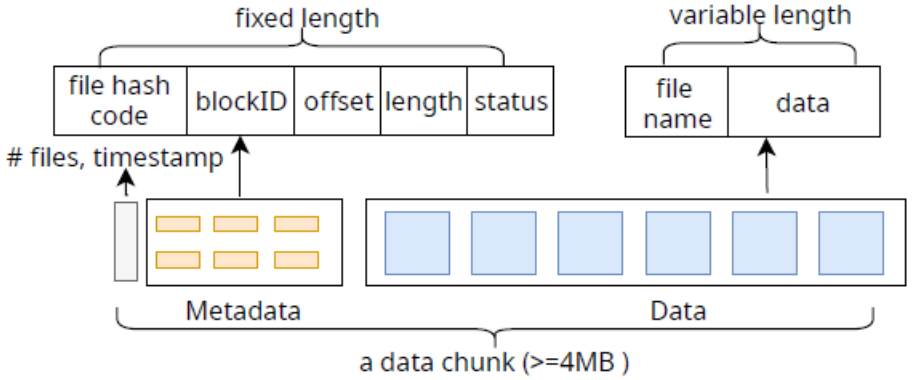
- Distributed in-memory key/value server as metadata server
- Metadata snapshot
- Task-grained distributed cache
- POSIX-compliant interface

The first step: write files into DIESEL

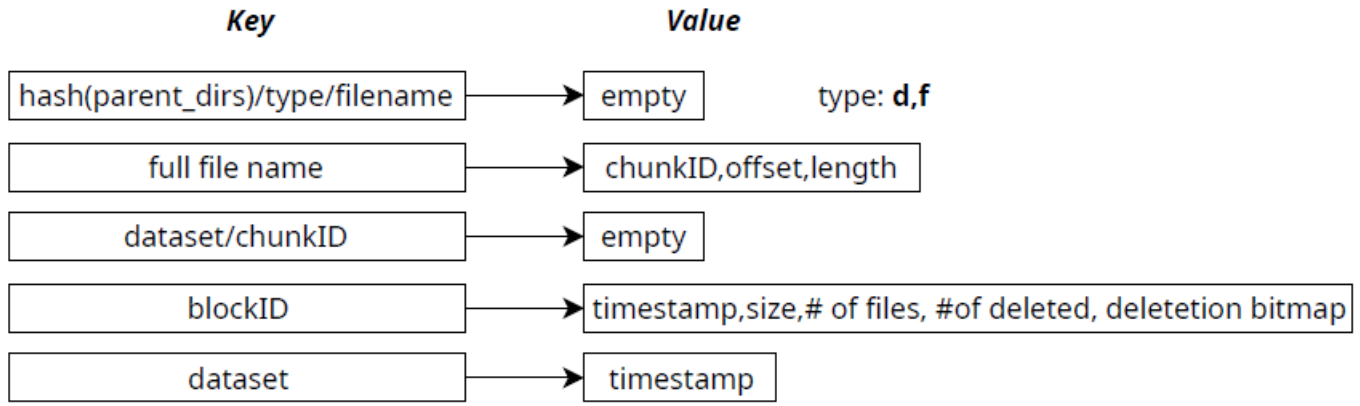


- Files are merged into large chunks
- Metadata is saved in the head of each chunk as well as in an in-memory key/value server

Metadata storage in DIESEL



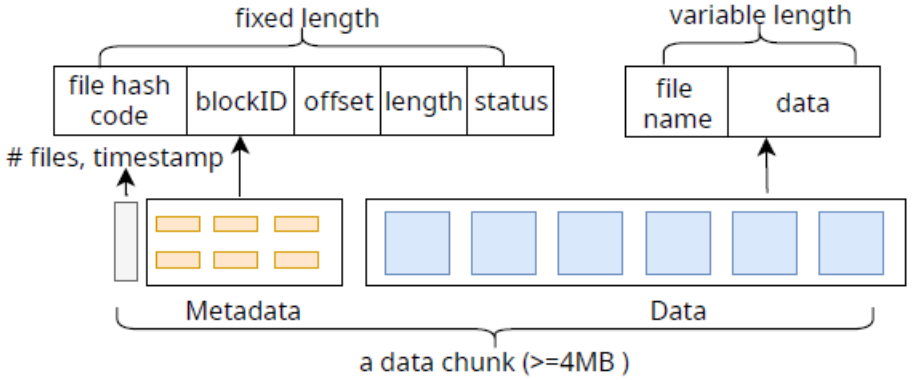
(a) Data chunk layout in DIESEL



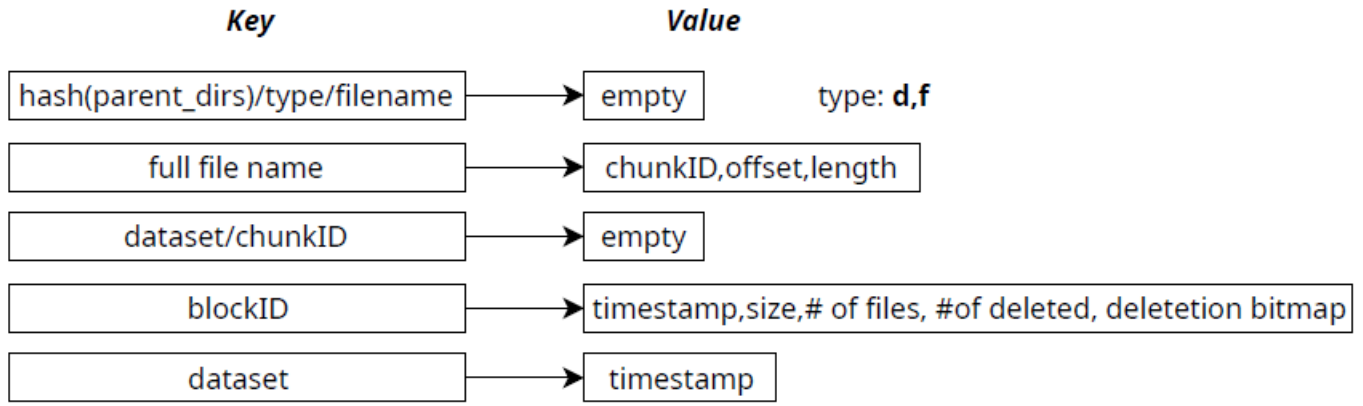
(b) Metadata design in key-value database

Why need to store metadata with data chunks?

Metadata storage in DIESEL



(a) Data chunk layout in DIESEL

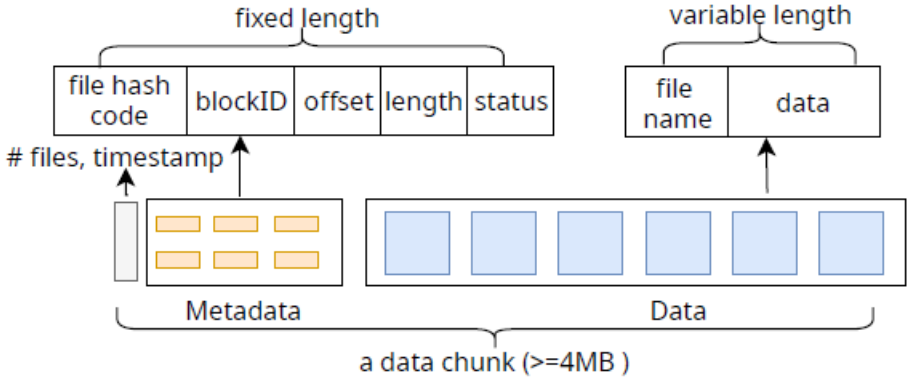


(b) Metadata design in key-value database

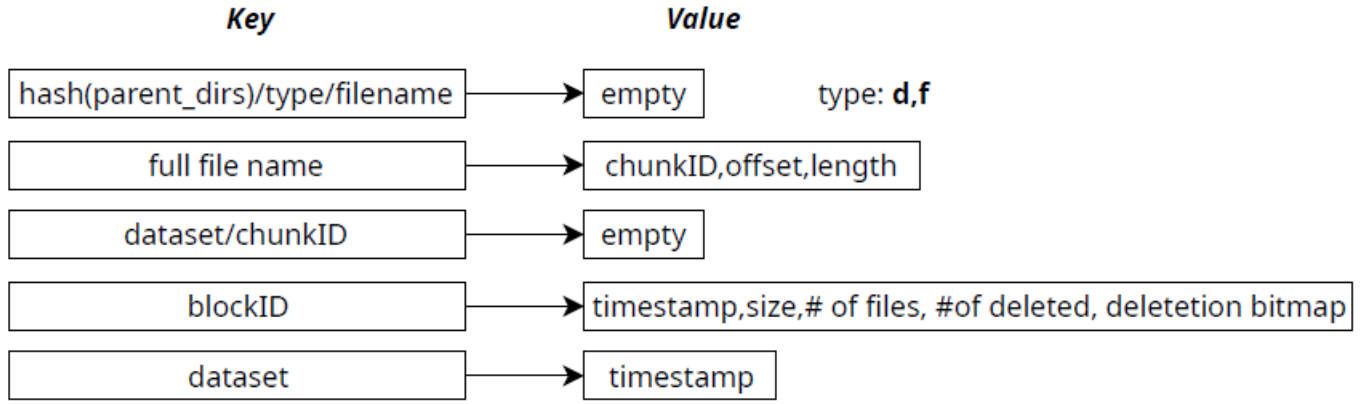
Why need to store metadata with data chunks?

The in-memory key/value server may **fail**:

Metadata storage in DIESEL



(a) Data chunk layout in DIESEL



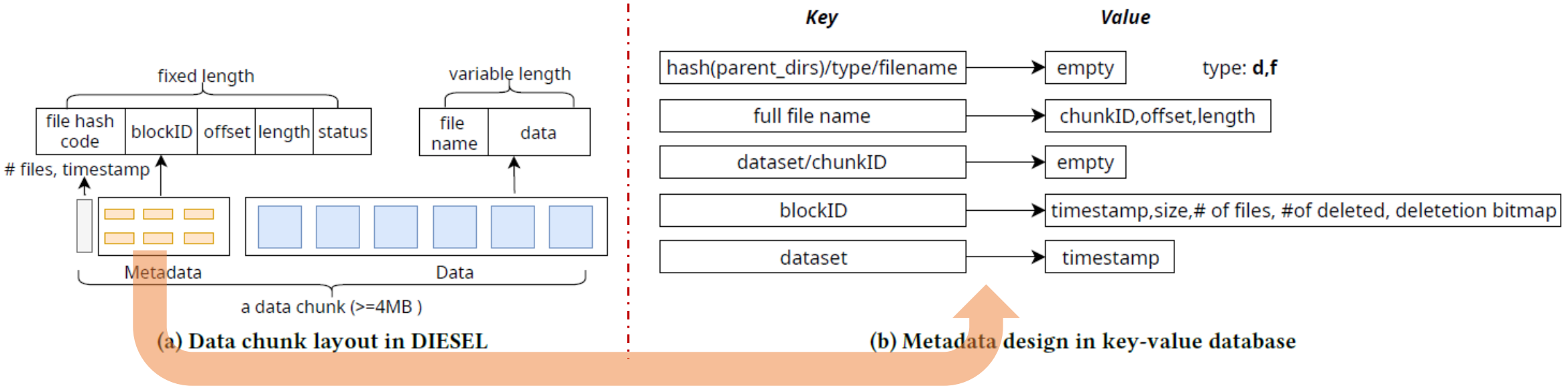
(b) Metadata design in key-value database

Why need to store metadata with data chunks?

The in-memory key/value server may **fail**:

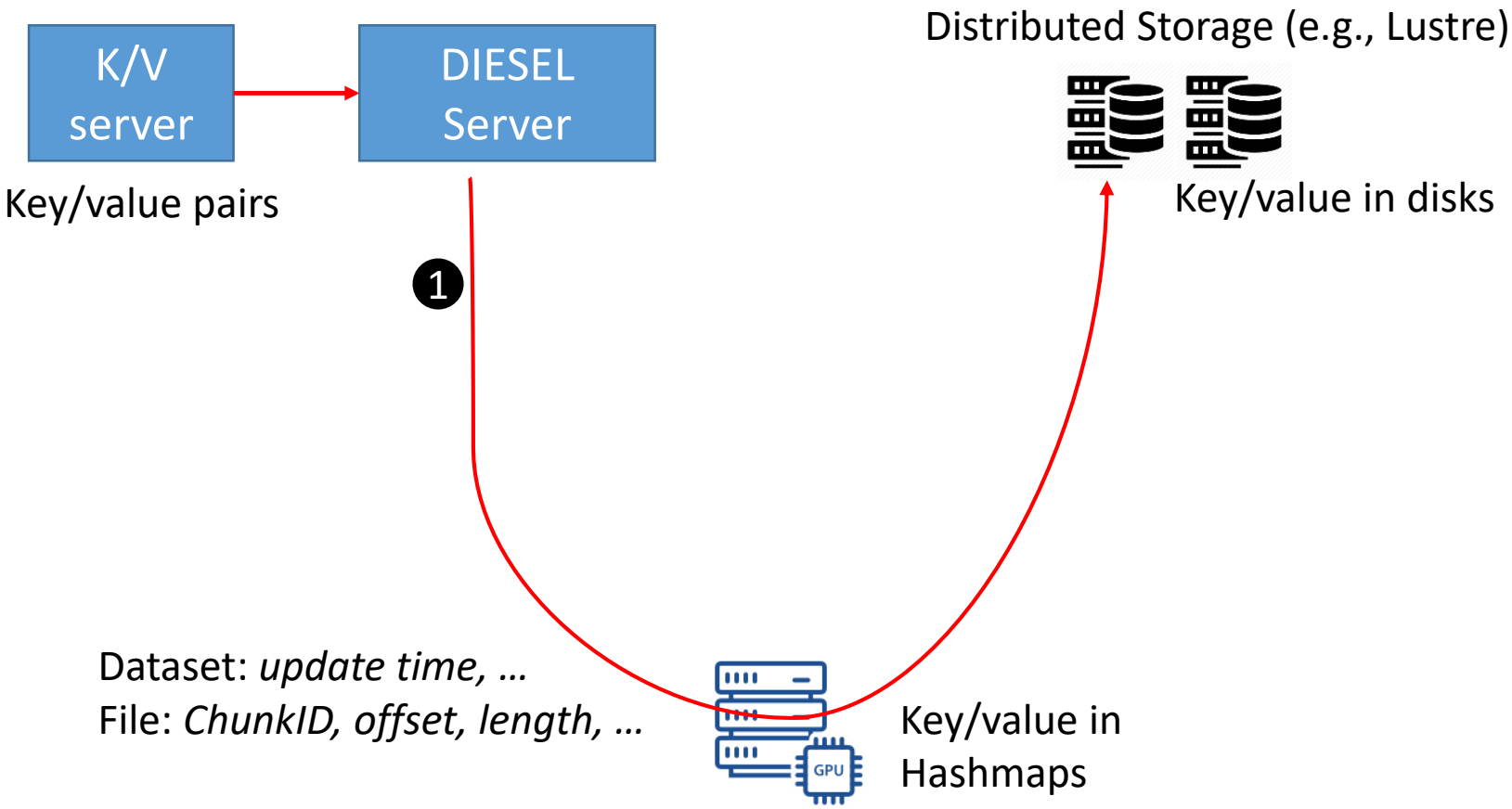
- Lost recently written entries
- Lost all entries due to power failure

Reconstruct key/value pairs from data chunks



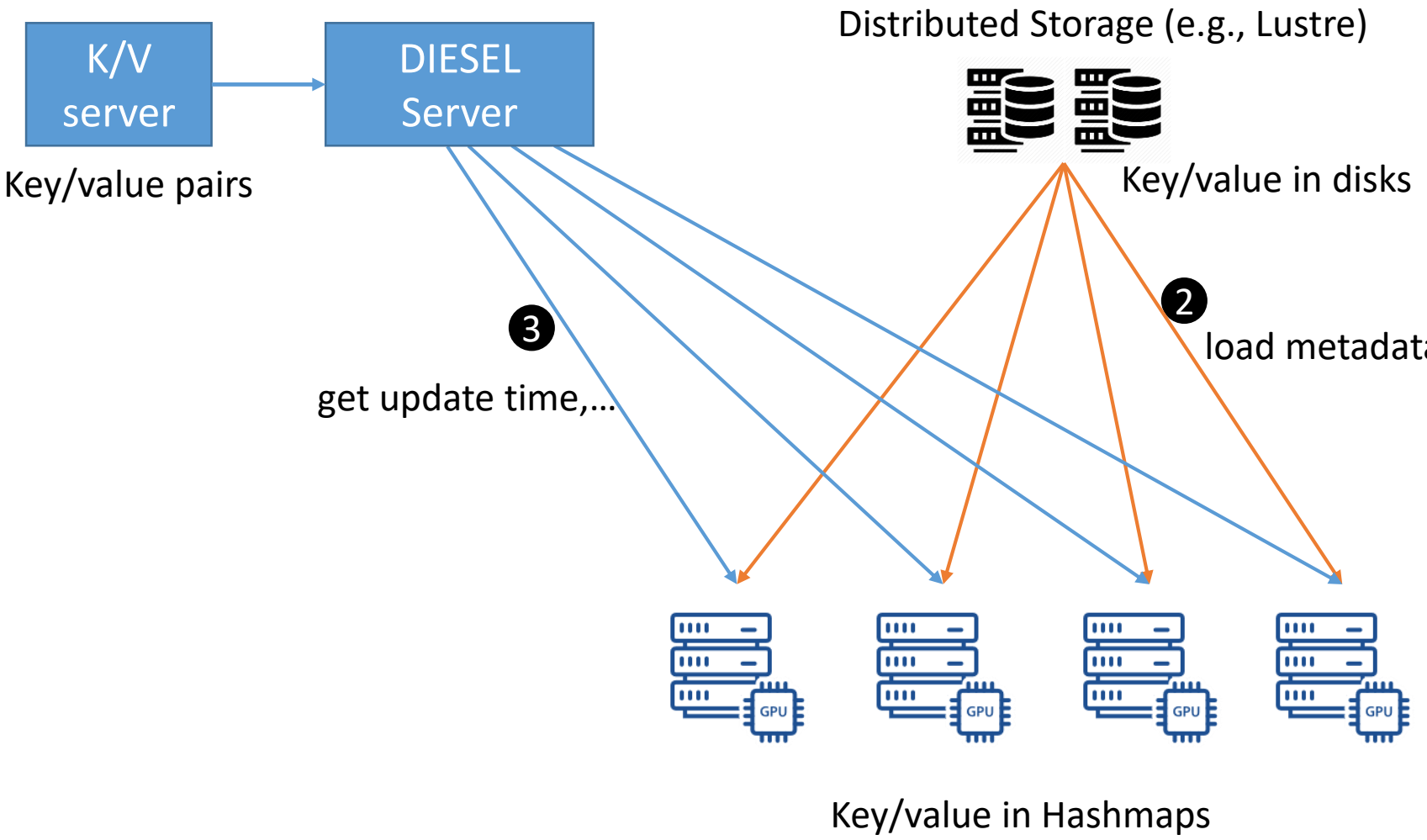
Reconstruct key/value pairs from data chunks

Metadata snapshot – download from DIESEL



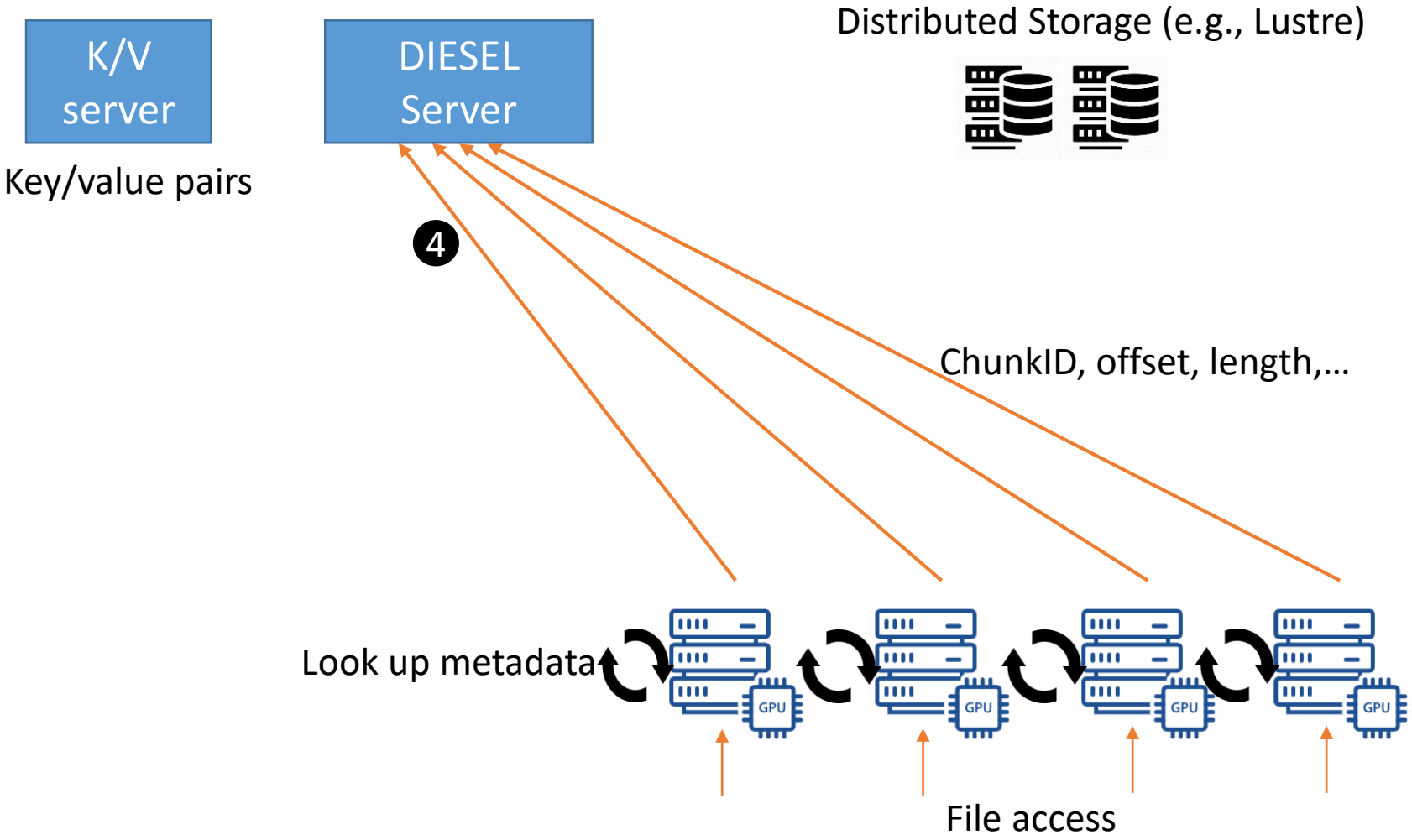
- Get metadata of a dataset
- Save metadata to a disk file on distributed storage

Metadata snapshot – load from disks



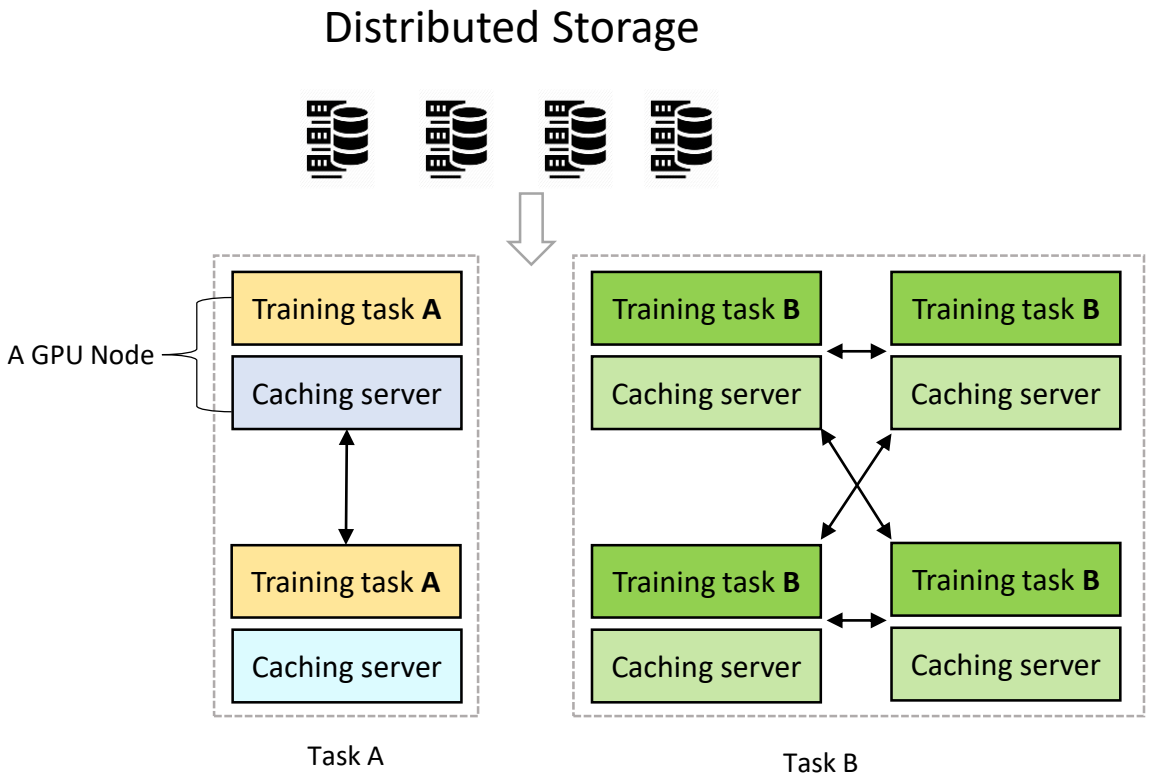
- Load metadata from disk file
- Check the update timestamp

Metadata snapshot – bypass the metadata server to retrieve files



- Lookup metadata locally, bypass the metadata server
- Read data chunks

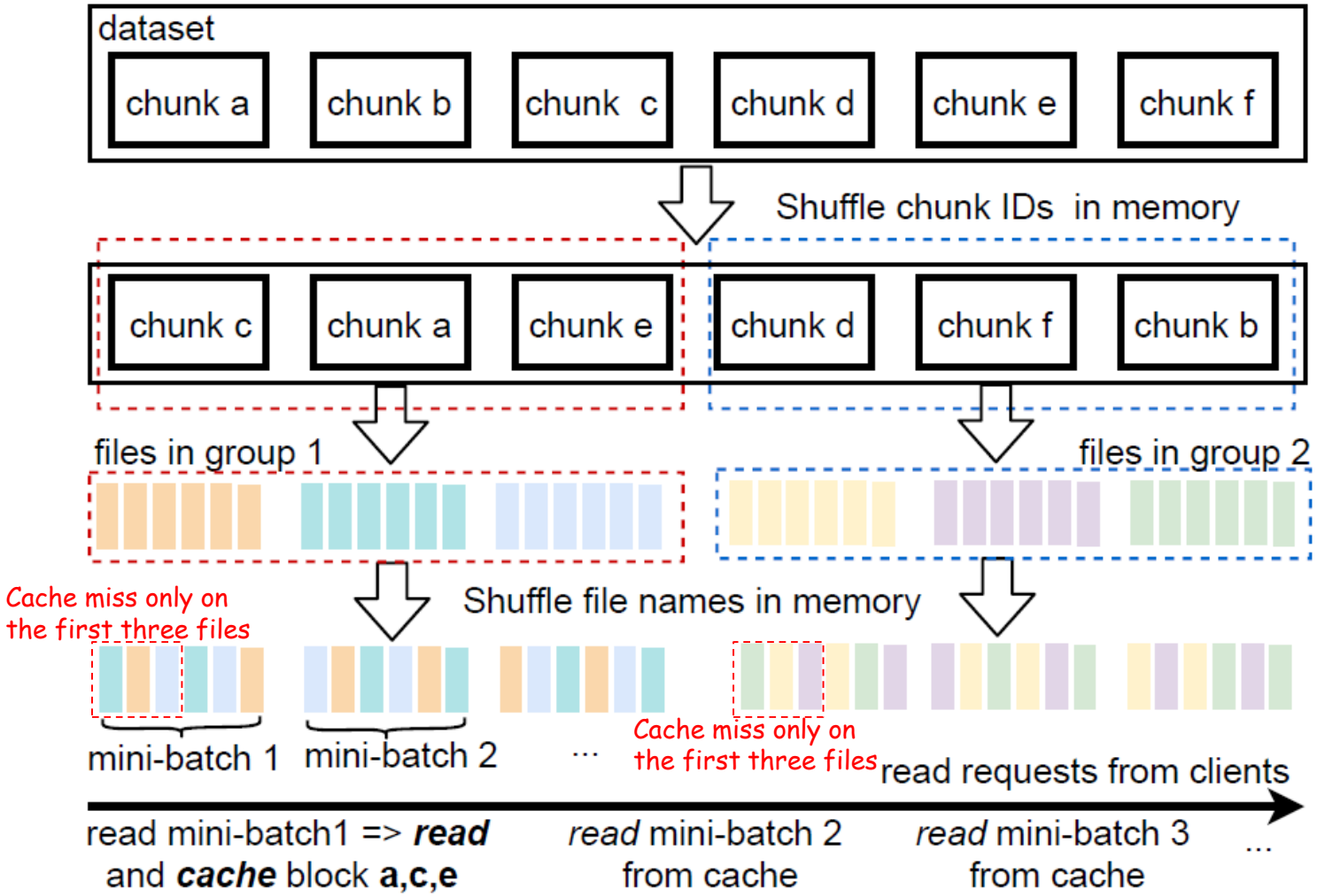
Task-grained distributed caching system



DIESEL deploys a task-grained distributed cache across the GPU nodes of a DLT task:

- Isolate node failure
- Reduce # of network connections
- Lifetime follows the DLT task

Chunk-based shuffle method



- In DLT tasks, the file access order does not matter, as long as it is random
- DIESEL generates a shuffled file list
- Convert individual file reads into large chunk reads
- Small memory footprint

Experimental Setup

	Storage Machine	Test Machine
# Machines	6	10
OS	CentOS 7.6 Kernel 3.10.0-957	
Network	100 Gbps Infiniband	
CPU	Intel(R) Xeon(R) Gold 6148 (20c40t) ×2	Intel(R) Xeon(R) Gold 6130 (16c32t) ×2
GPU	-	Tesla V100 SXM2 32GB ×8
Memory	512 GB	256 GB
Disk	NVME SSD 3.8T ×6	-
Software	Lustre 2.12.2	Memcached 1.4.15, FUSE 3.4.2, Twemproxy 0.4.1, Redis 5.0.5, Intel MPI 2017, PyTorch 1.1.0

Dataset:

- ImageNet-1K (1.28million images, ~150GB)

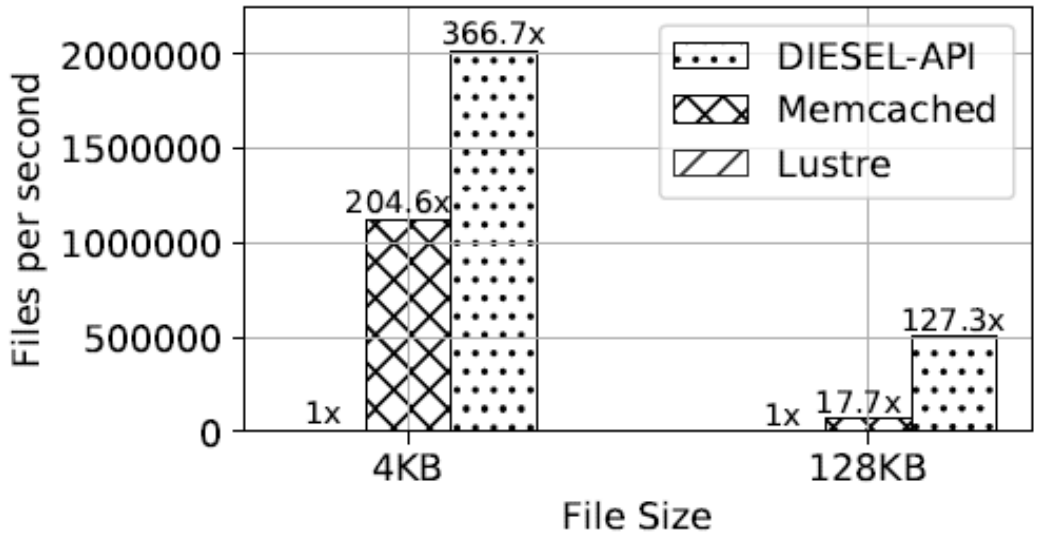
Framework:

- PyTorch

Models:

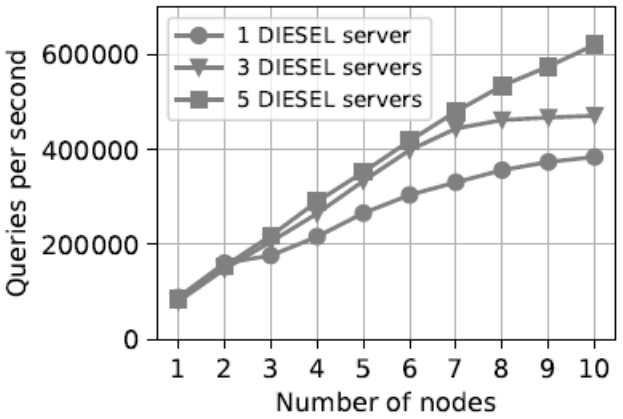
- AlexNet
- VGG-11
- ResNet-18
- ResNet-50

Evaluation on file writing

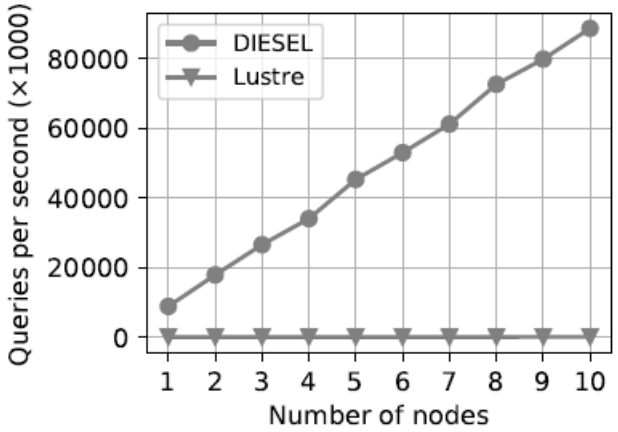


- DIESEL is faster than the Lustre and Memcached on file writing
- On 4KB file size, DIESEL is about 200x and 360x faster than the Memcached and Lustre, respectively
- On 128KB file size, DIESEL is about 17x and 120x faster than the Memcached and Lustre, respectively

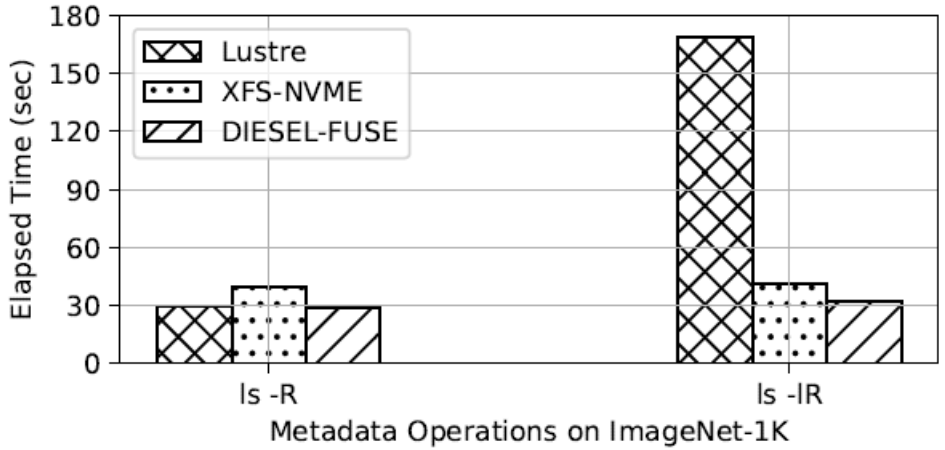
Evaluation on metadata access and metadata snapshot



(a) Metadata performance with 1,3 and 5 DIESEL servers



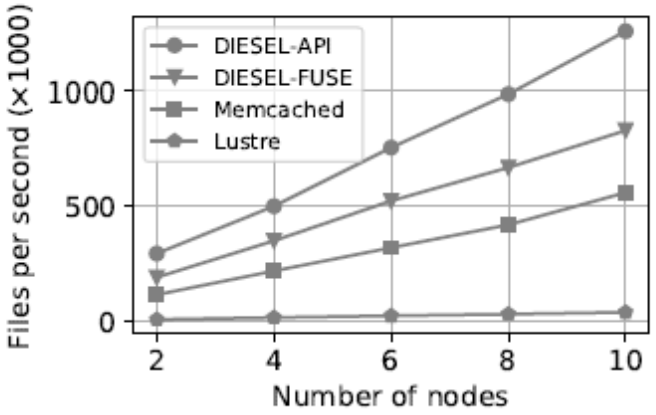
(b) Metadata performance with the metadata snapshot enabled



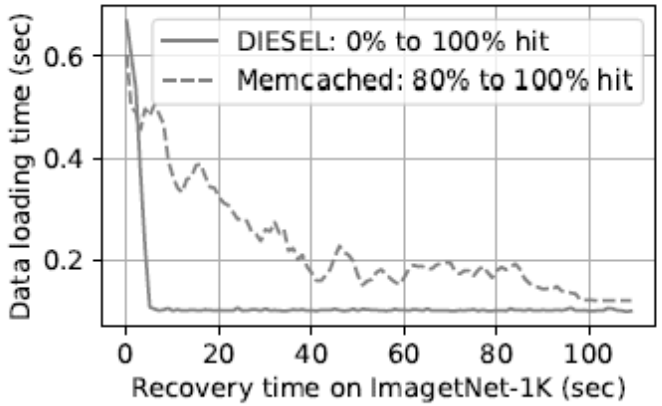
(c) Comparison of elapsed time on metadata operations

- Increasing the number of DIESEL server will increase the metadata access performance when the metadata snapshot is disabled
- With the metadata snapshot enabled, the metadata access throughput increases linearly with the number of workers
- DIESEL is faster than Lustre and XFS-NVME on metadata query response time

Evaluation on task-grained distributed cache



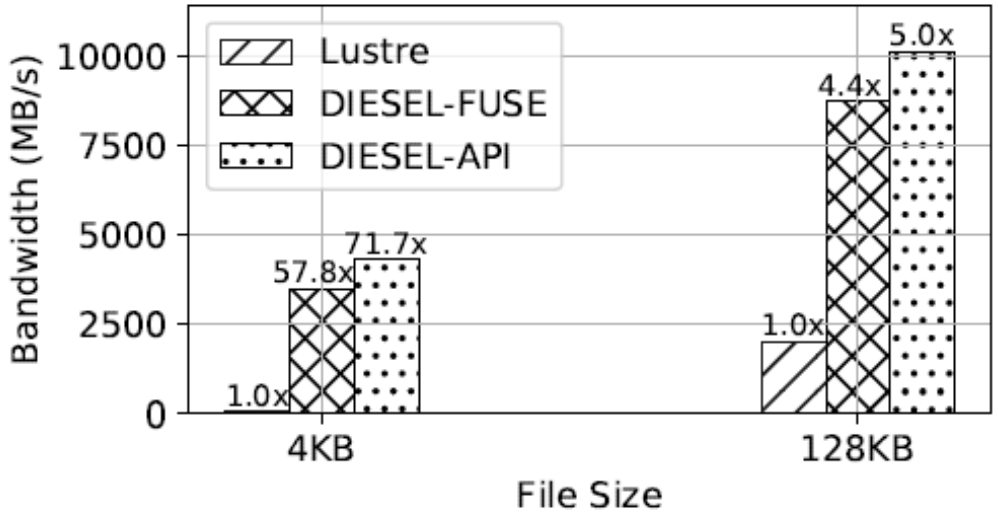
(a) Read performance comparison on 4KB files



(b) Data loading time with recovery time

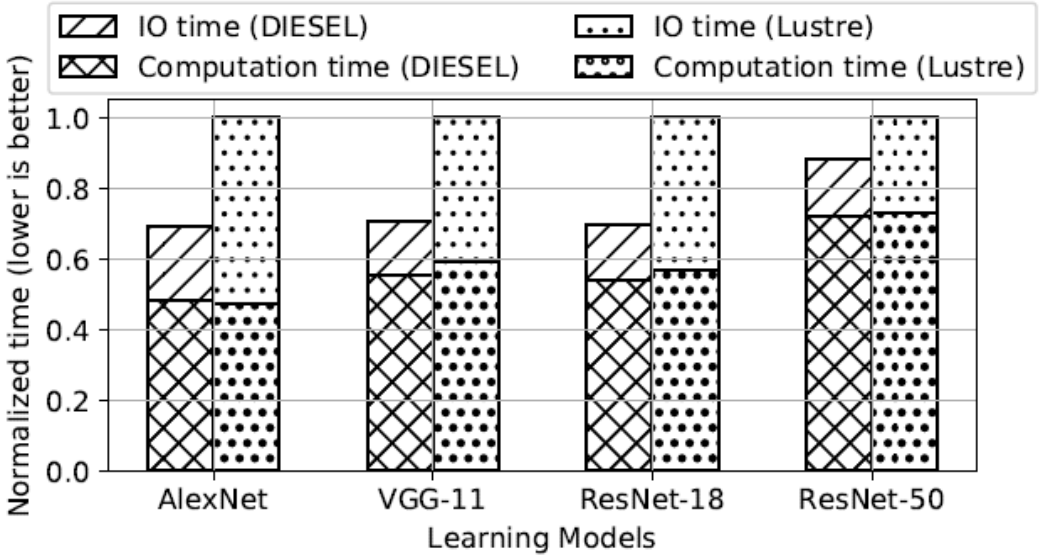
- Task-grained distributed cache achieves better performance than existing global in-memory caching system
- The task-grained distributed cache's "Cold-booting" time is shorter than Memcached's node recovery time

Evaluation on chunk-based shuffle method



- Chunk-based shuffle method has higher read bandwidth than the Lustre filesystem
- On 4KB file reads, DIESEL is more than 50x faster than the Lustre
- On 128KB file reads, DIESEL is more than 4x faster than the Lustre

Evaluation on real-world DLT tasks



- DIESEL reduces about 15%-27% time in end-to-end training tasks

Summary

DIESEL is a storage and caching system co-designed for DLT tasks:

- Efficient metadata management
  - Distributed in-memory key/value database
  - Metadata snapshot mechanism
- Task-grained distributed caching system isolates node failures
- Chunk-based shuffle method converts shuffled small file reads into large chunk reads
- Demonstrated efficiency in real-word DLT tasks

Q & A

Our Research Group：Rapids@HKUST
https://github.com/RapidsAtHKUST