

NIKOLI DRYDEN

Clairvoyant Prefetching for Distributed Machine Learning I/O

Clairvoyant Prefetching for Distributed Machine Learning I/O

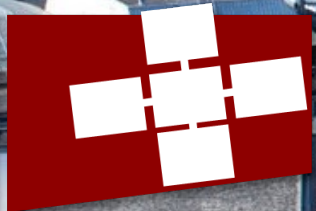
Nikoli Dryden
ndryden@ethz.ch

arXiv:2101.08734

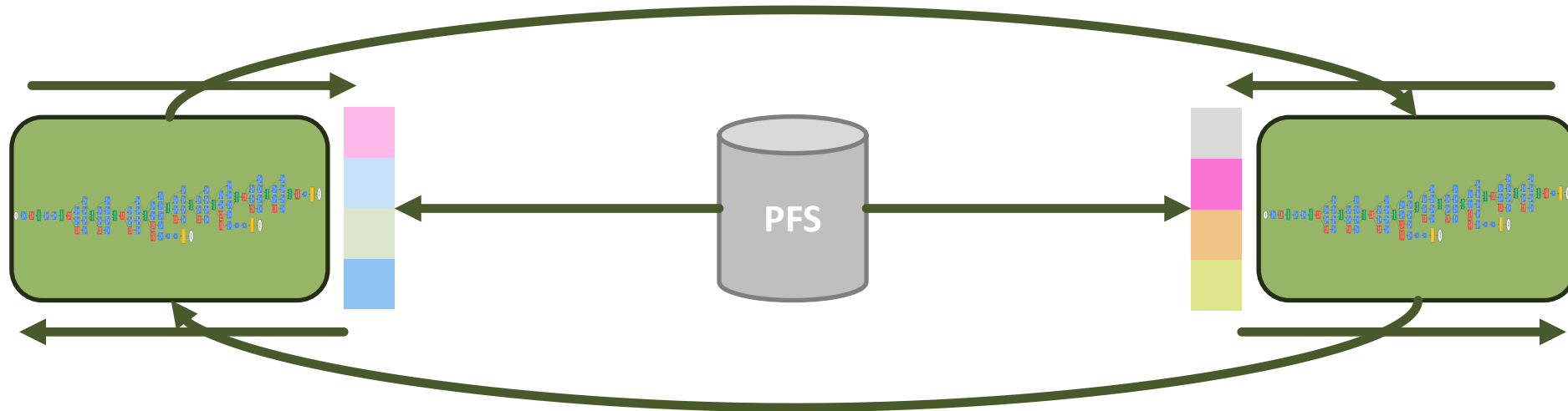
Roman Böhringer
romanboe@student.ethz.ch

Tal Ben-Nun
tal.bennun@inf.ethz.ch

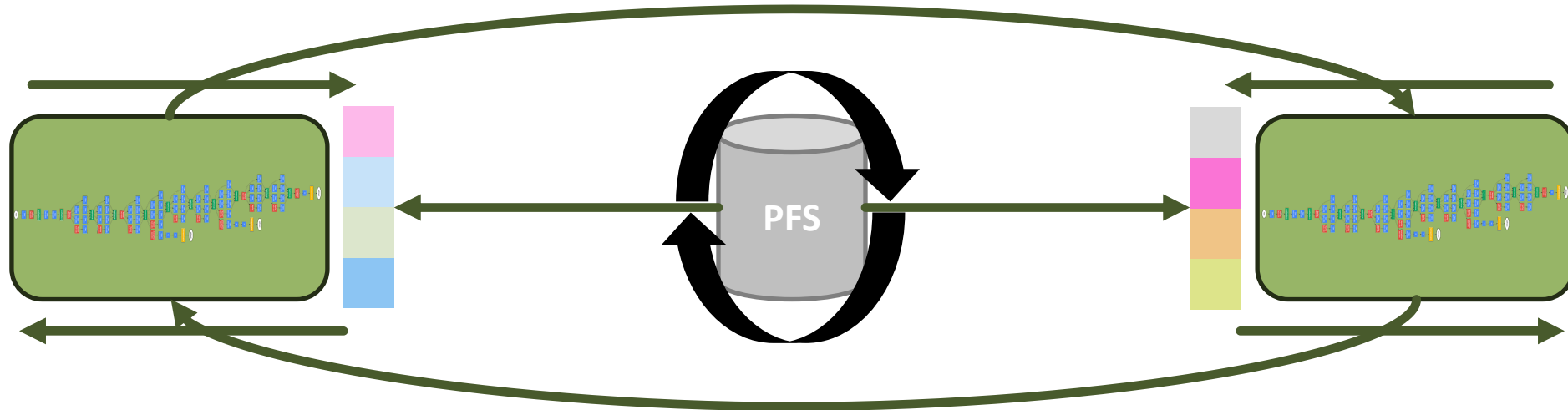
Torsten Hoefler
torsten.hoefler@inf.ethz.ch



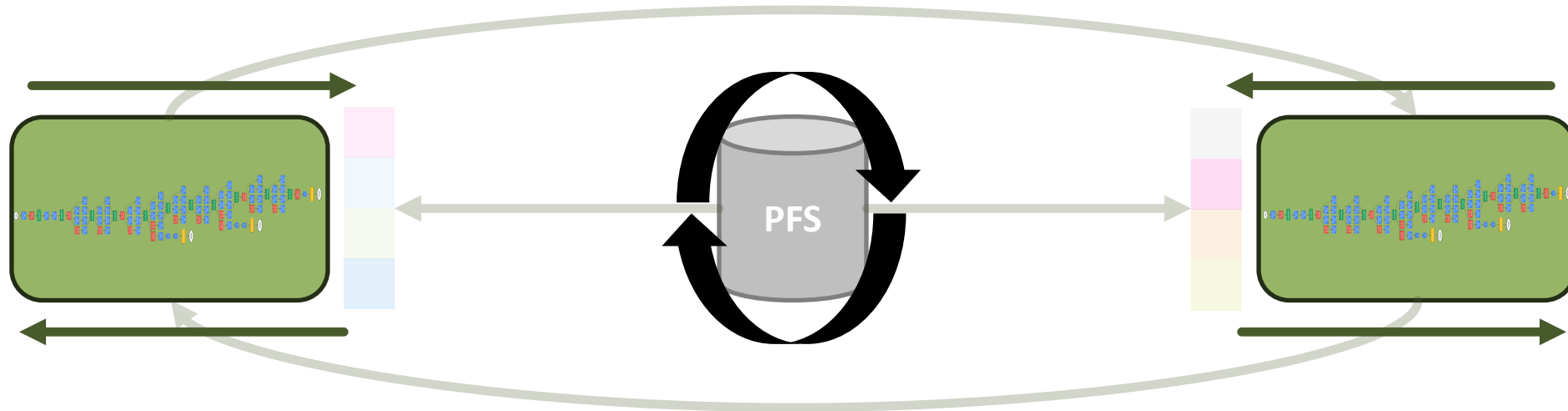
High Performance Training



High Performance Training

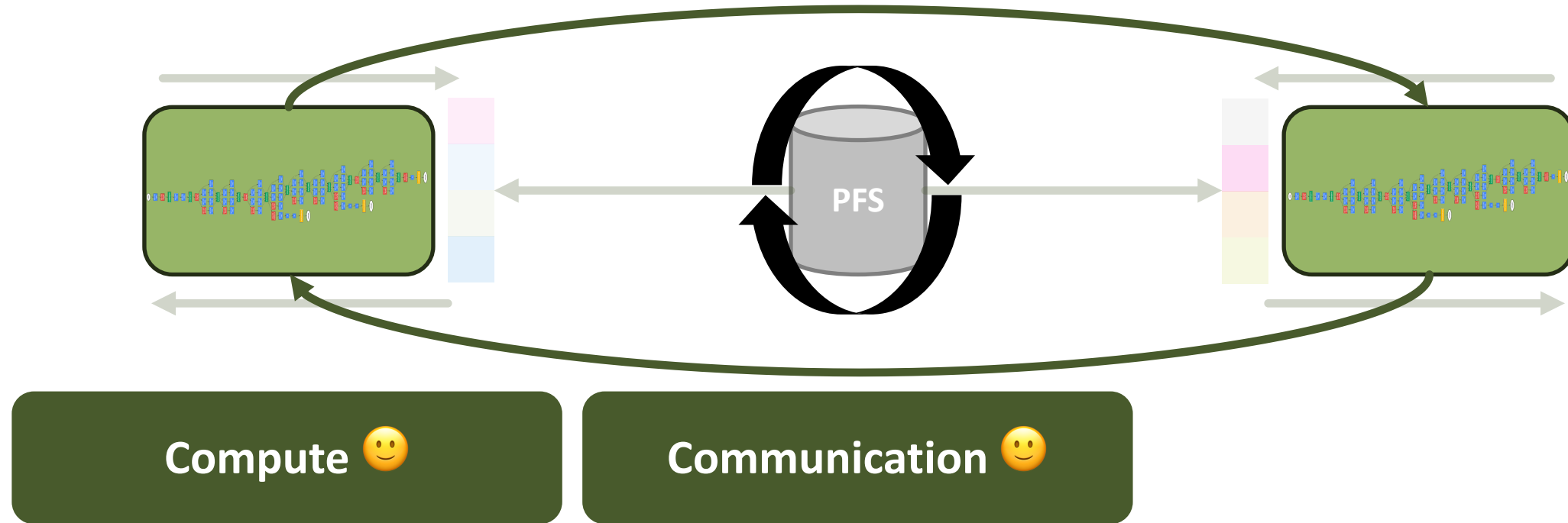


High Performance Training

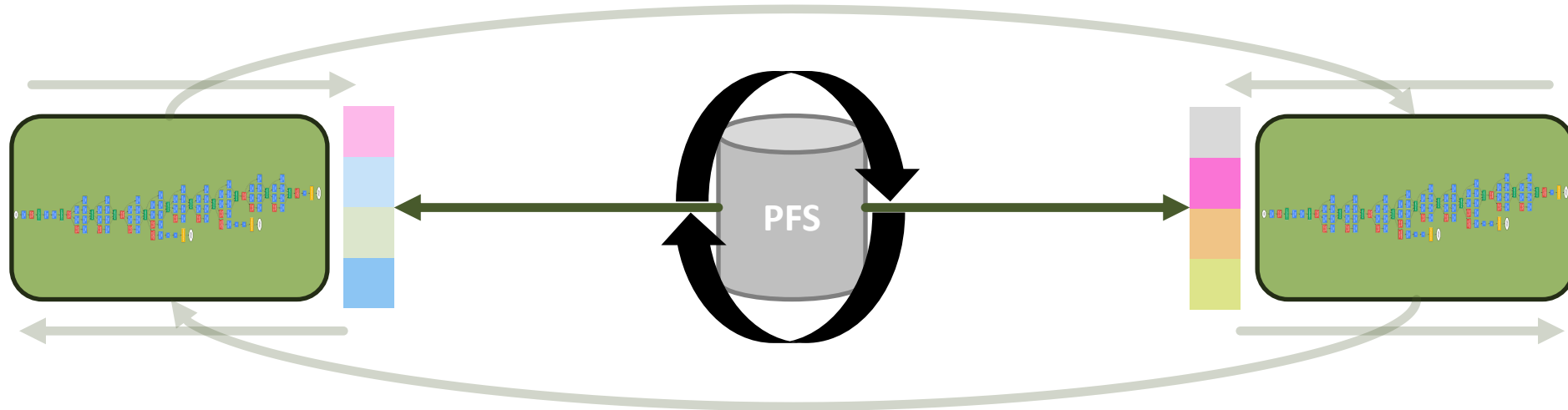


Compute 😊

High Performance Training



High Performance Training

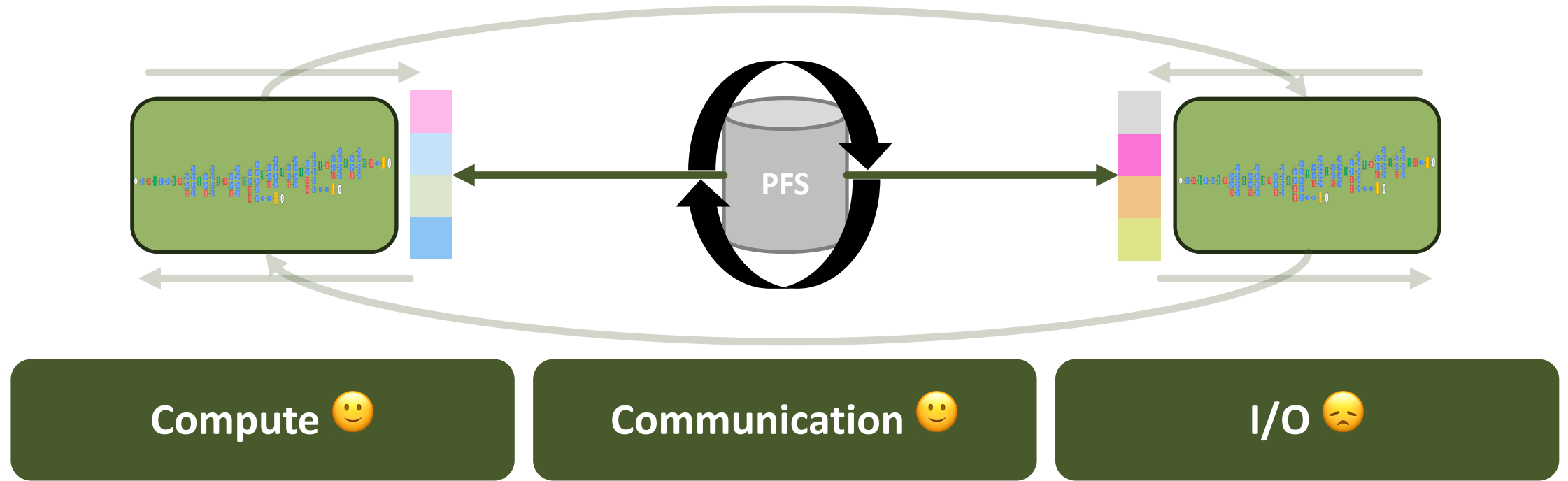


Compute 😊

Communication 😊

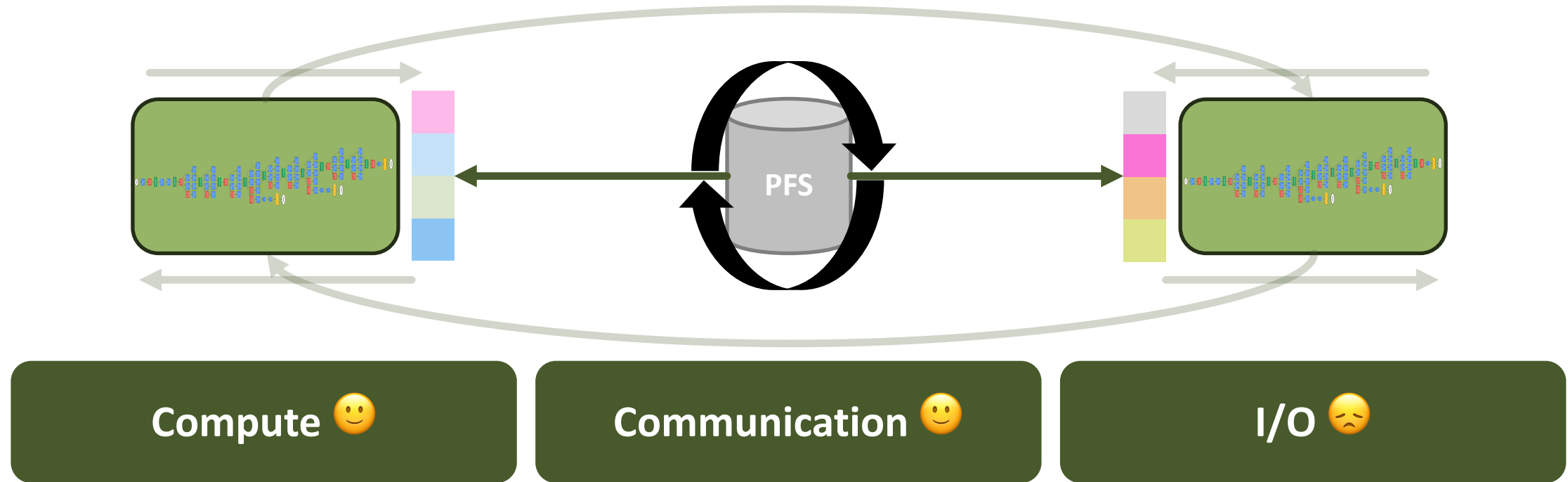
I/O 😞

High Performance Training



I/O overheads up to **85%**!

High Performance Training

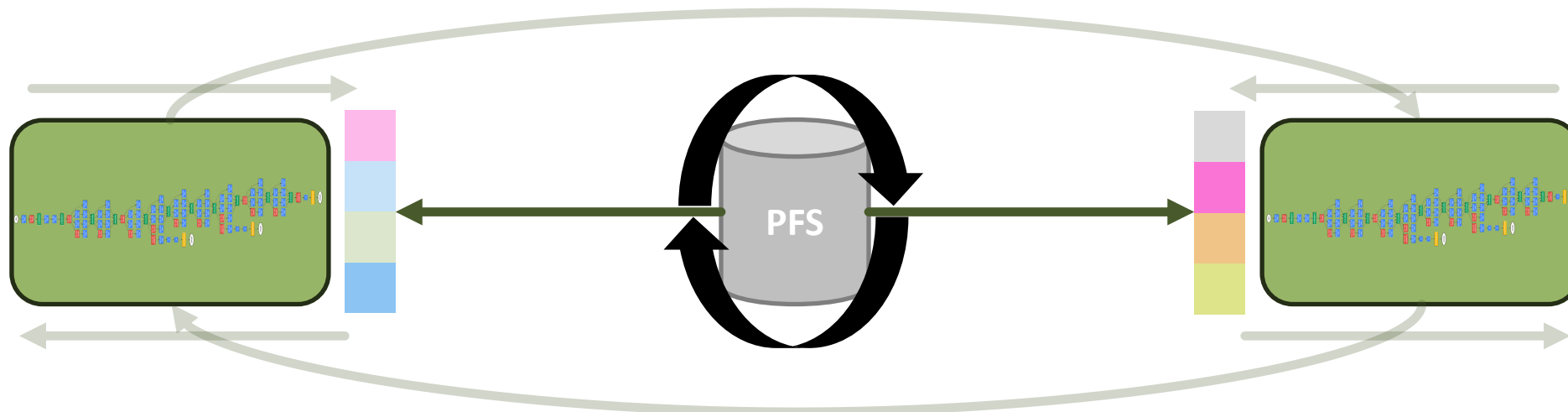


I/O overheads up to **85%**!

Example: ResNet-50 on ImageNet-1k

- ImageNet-1k: ~150 GiB, ~1.3M images (average: 115 KiB, range: 508 B – 15 MiB)
- MLPerf on one A100: ~2.9K samples/s → ~333 MiB/s random access
- → **2 SSDs / GPU**
- *2-4x for scientific problems like CosmoFlow*

High Performance Training



Compute 😊

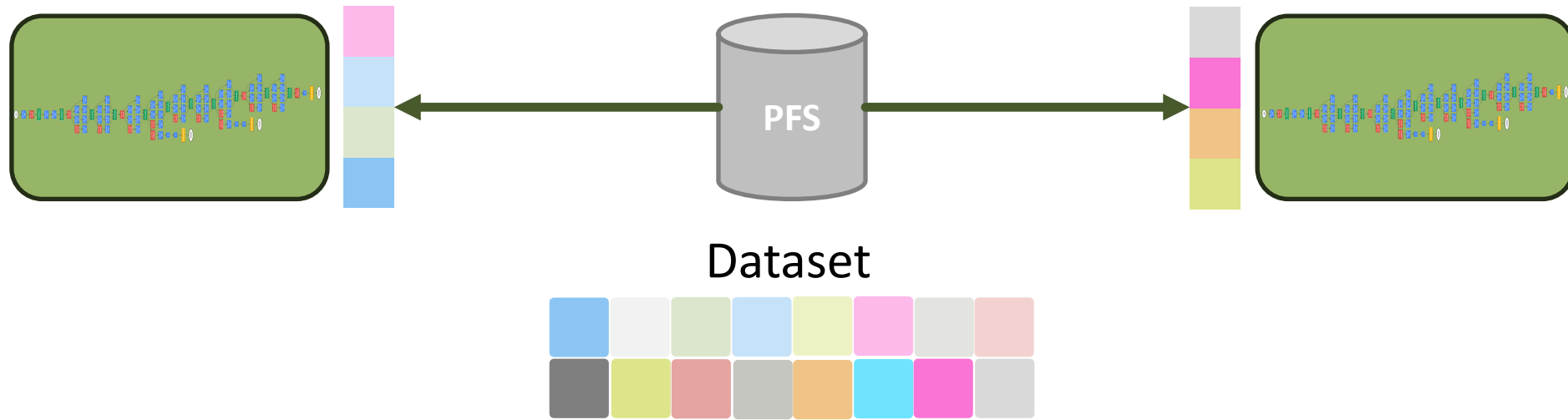
Communication 😊

I/O 😊

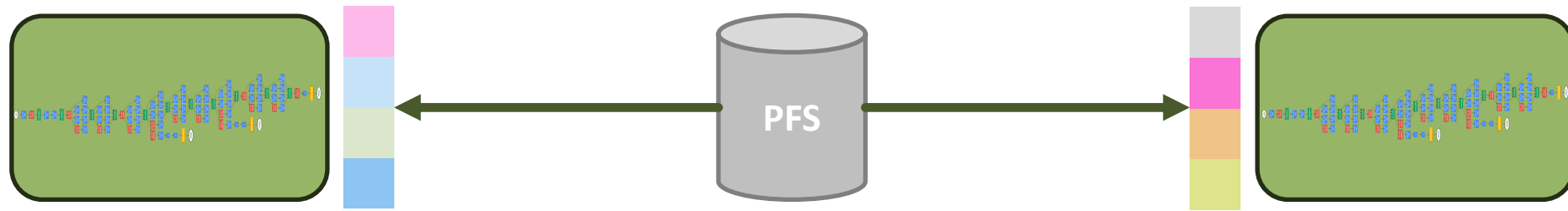
NoPFS: Near-optimal Pre-Fetching System

Up to 5.4x end-to-end training improvements!

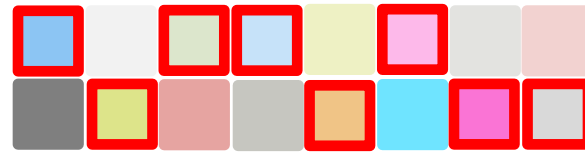
I/O for Machine Learning



I/O for Machine Learning

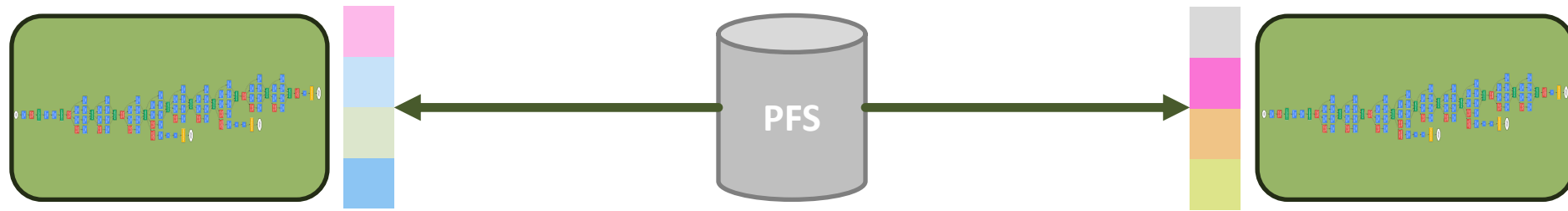


Dataset

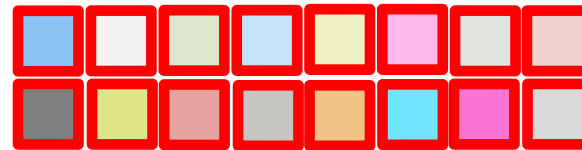


Randomly sample mini-batch

I/O for Machine Learning



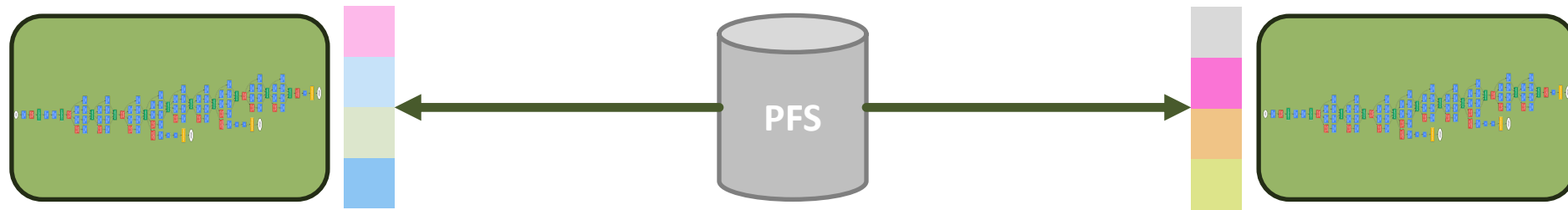
Dataset



Randomly sample mini-batch

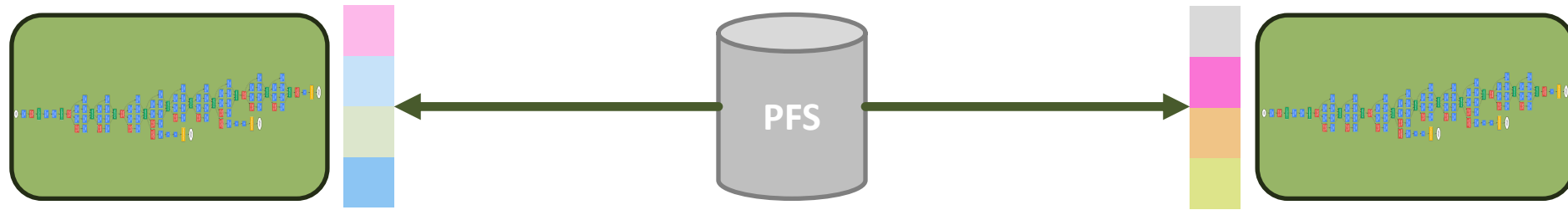
→ Epoch

I/O for Machine Learning



What makes a good I/O framework?

I/O for Machine Learning

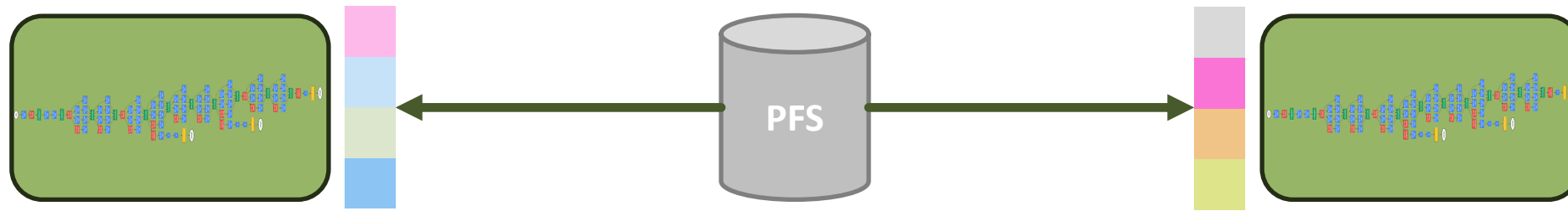


What makes a good I/O framework?

System Scalability



I/O for Machine Learning



What makes a good I/O framework?

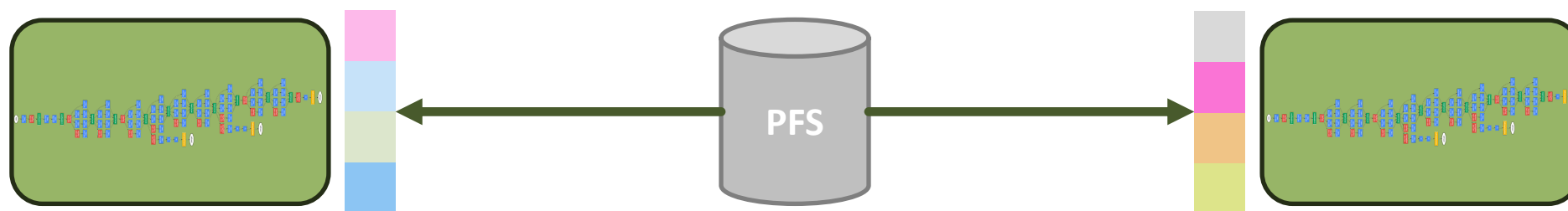
System Scalability



Dataset Scalability



I/O for Machine Learning



What makes a good I/O framework?

System Scalability



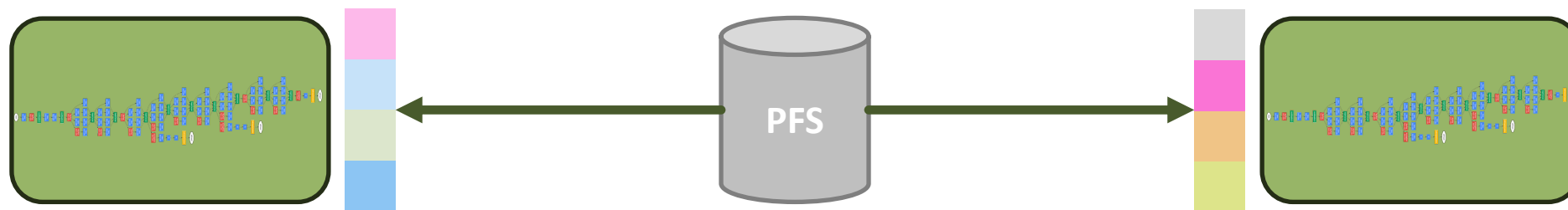
Dataset Scalability



Full Randomization



I/O for Machine Learning



What makes a good I/O framework?

System Scalability



Dataset Scalability



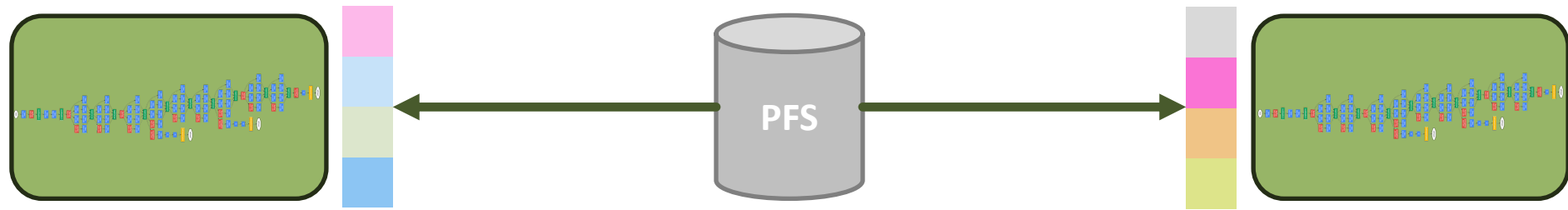
Full Randomization



Hardware Independence



I/O for Machine Learning



What makes a good I/O framework?

System Scalability



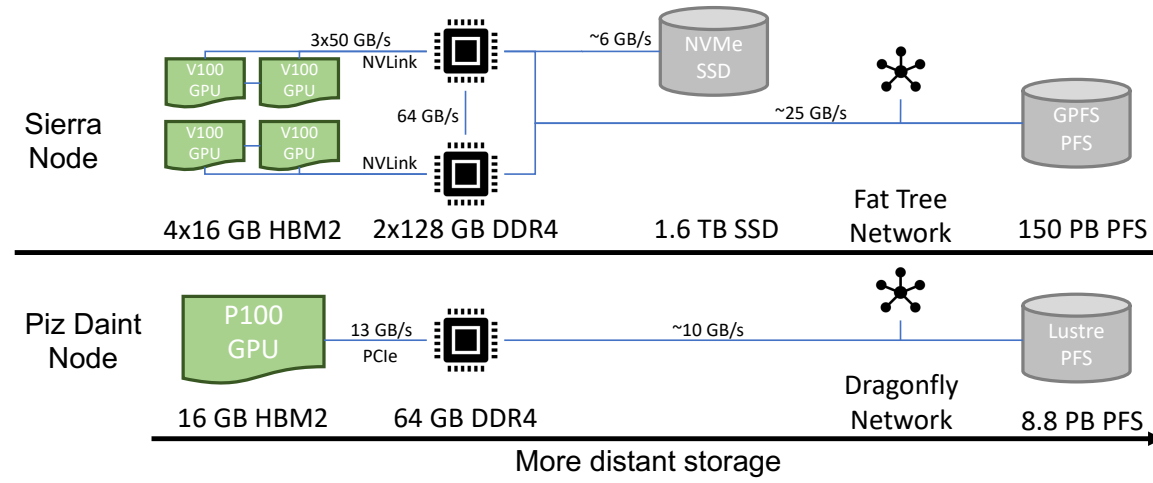
Dataset Scalability



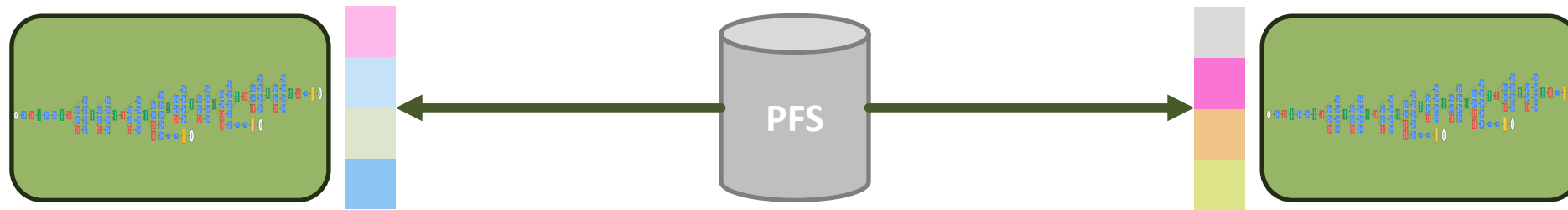
Full Randomization



Hardware Independence



I/O for Machine Learning



What makes a good I/O framework?

System Scalability



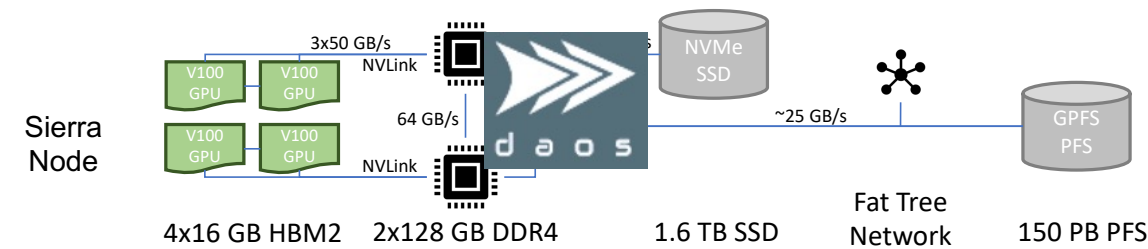
Dataset Scalability



Full Randomization



Hardware Independence



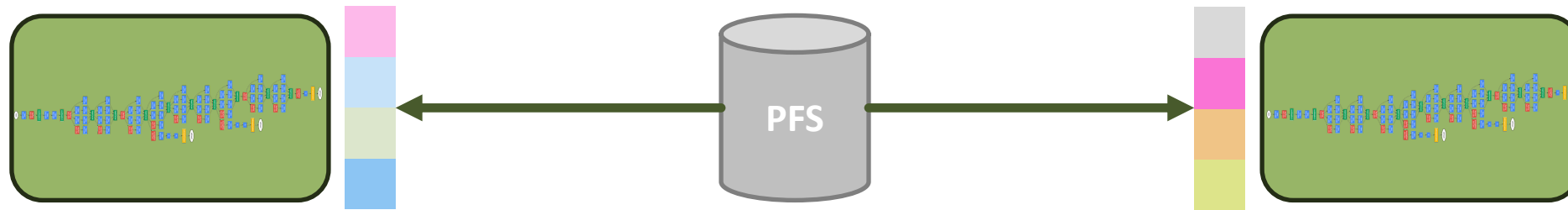
Piz D No

Livermore's El Capitan Supercomputer to Debut HPE Rabbit 'Near-Node' Storage
February 18, 2021

A near-node local storage innovation called Rabbit factored heavily into Lawrence Livermore National Laboratory's decision to select Cray's proposal for its CORAL-2 machine, the lab's first exascale-class supercomputer, El Capitan.

More distant storage

I/O for Machine Learning



What makes a good I/O framework?

System Scalability



Dataset Scalability



Full Randomization



Hardware Independence



Ease of Use

```
dataset = ImageFolder(data_dir, data_transforms)
dsampler = DistributedSampler(dataset, num_replicas=n, rank=rank)
dataloader = DataLoader(dataset, batch_size, sampler=dsampler)
```

Clairvoyant I/O

“Randomly sample mini-batch”

By “random”, we really mean pseudorandomly with a known seed!



We know the *exact* access pattern of *every* worker

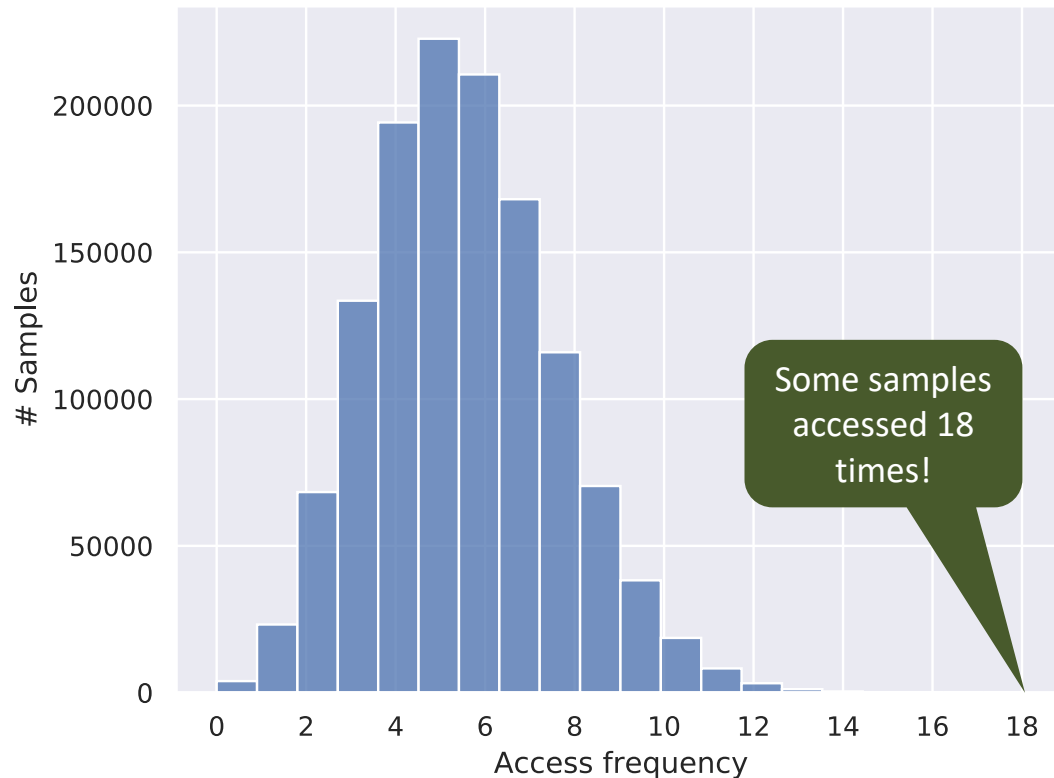


We can exploit clairvoyance to optimize (distributed) I/O

NoPFS is a hierarchical, distributed cache and prefetcher
that knows the future

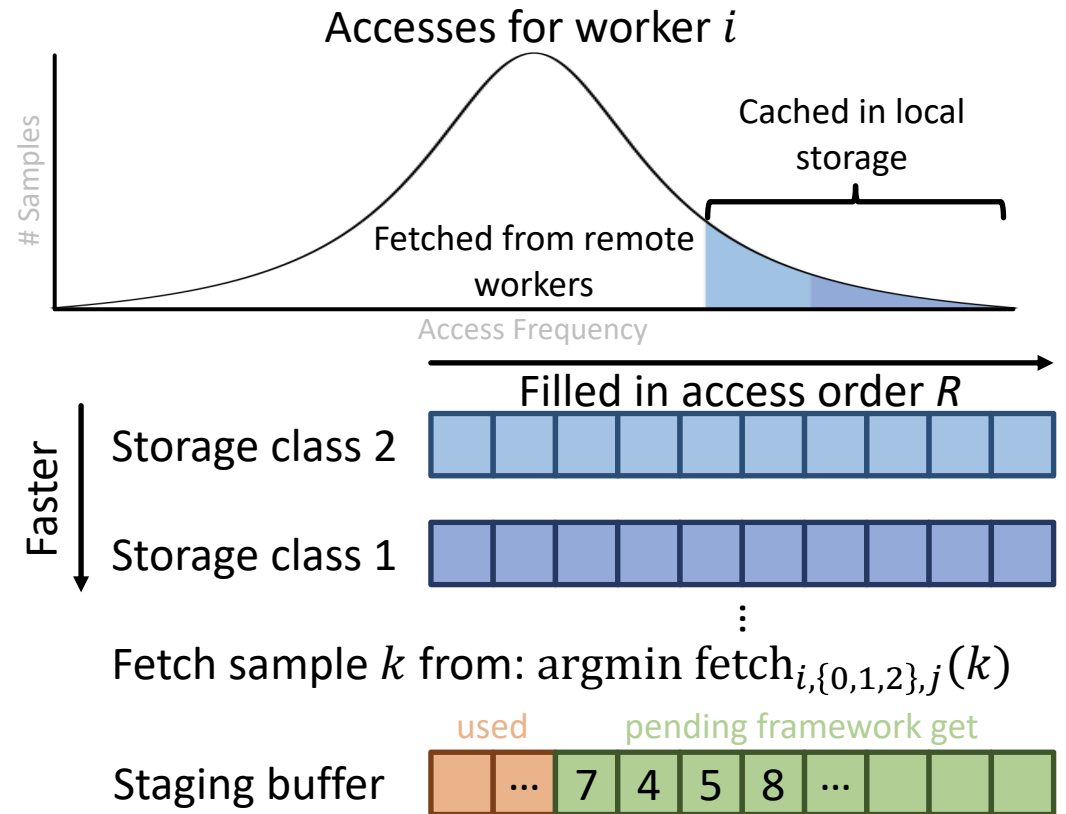
Clairvoyant Prefetching and Caching

Single-process access distribution
ImageNet-1k, 16 processes, 90 epochs

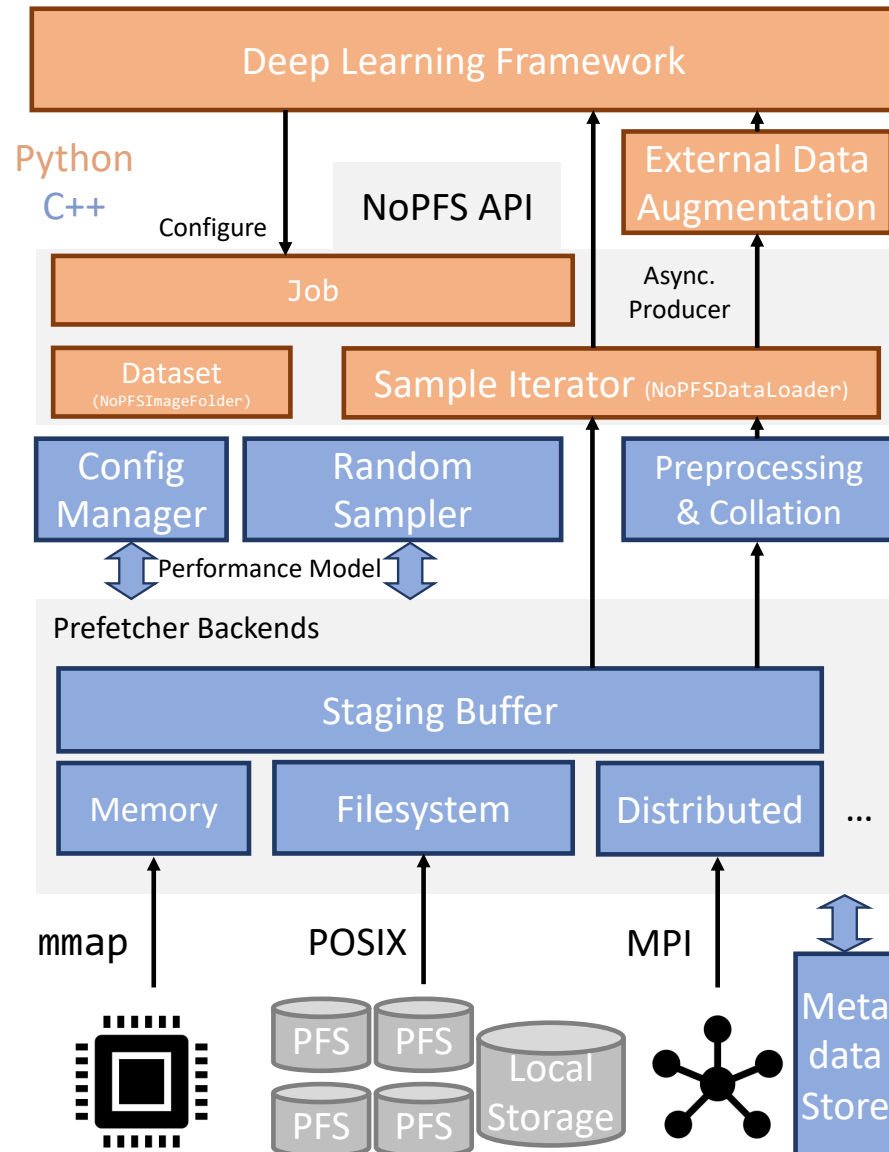


LEMMA 1. If a worker accesses a sample $\lceil (1 + \delta) \frac{E}{N} \rceil$ times (resp. $\lfloor (1 - \delta) \frac{E}{N} \rfloor$ times), at least one other worker will access the sample at most $\lceil (\frac{N-1-\delta}{N-1}) \frac{E}{N} \rceil$ (resp. at least $\lfloor (\frac{N-1+\delta}{N-1}) \frac{E}{N} \rfloor$) times.

PRNG seed \rightarrow Access stream $R = (\dots, 7, 4, 5, 8, \dots)$



NoPFS



Loading ImageNet:

PyTorch:

```
dataset = ImageFolder(data_dir, data_transforms)
dsampler = DistributedSampler(dataset, num_replicas=n, rank=rank)
dataloader = DataLoader(dataset, batch_size, sampler=dsampler)
```

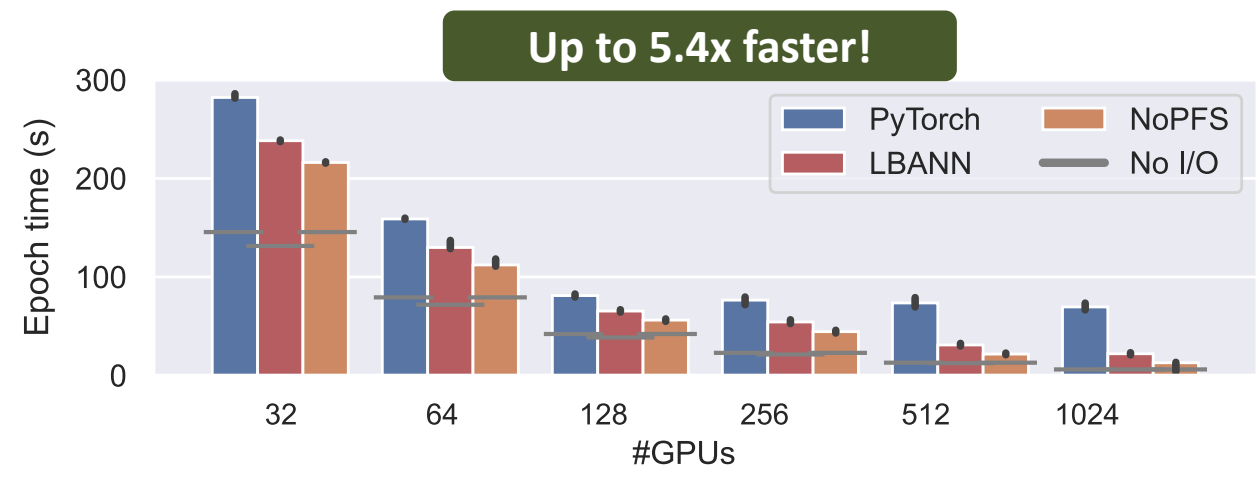
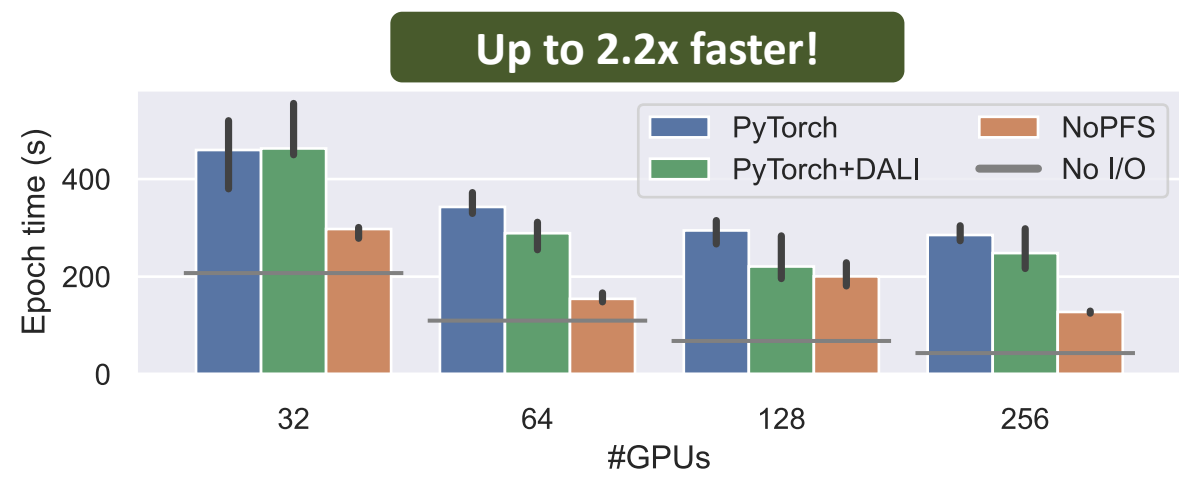
NoPFS:

```
job = Job(data_dir, batch_size, num_epochs, 'uniform', drop_last)
dataset = NoPFSImageFolder(data_dir, job, data_transforms)
dataloader = NoPFSDataLoader(dataset)
```

Performance



Runtime per epoch

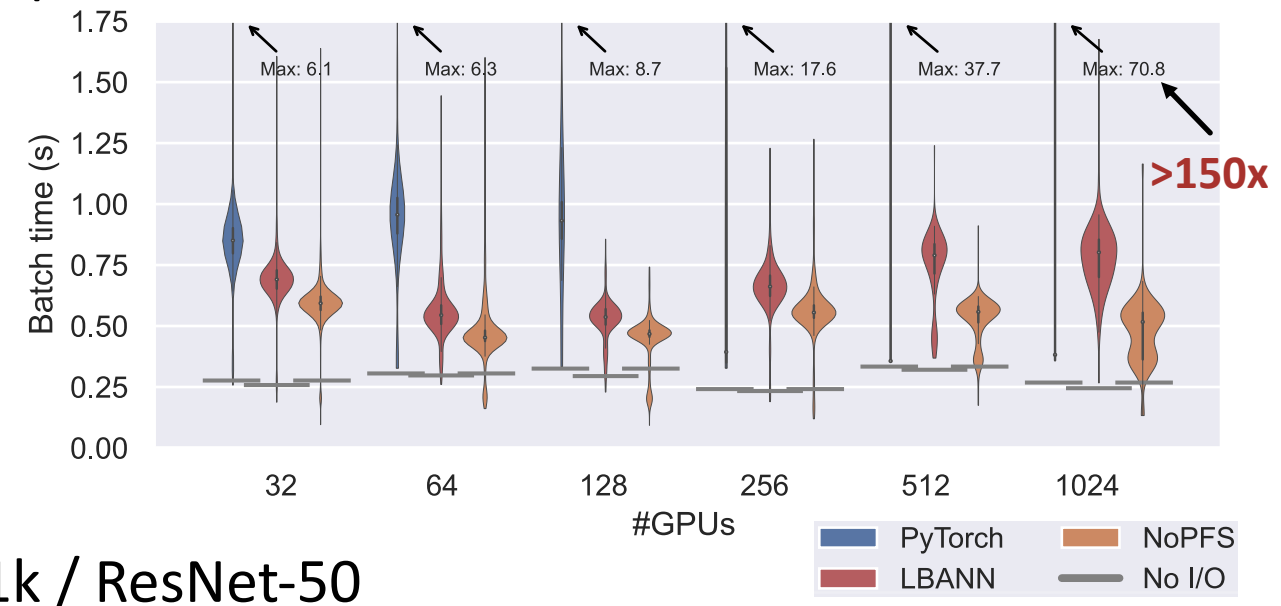
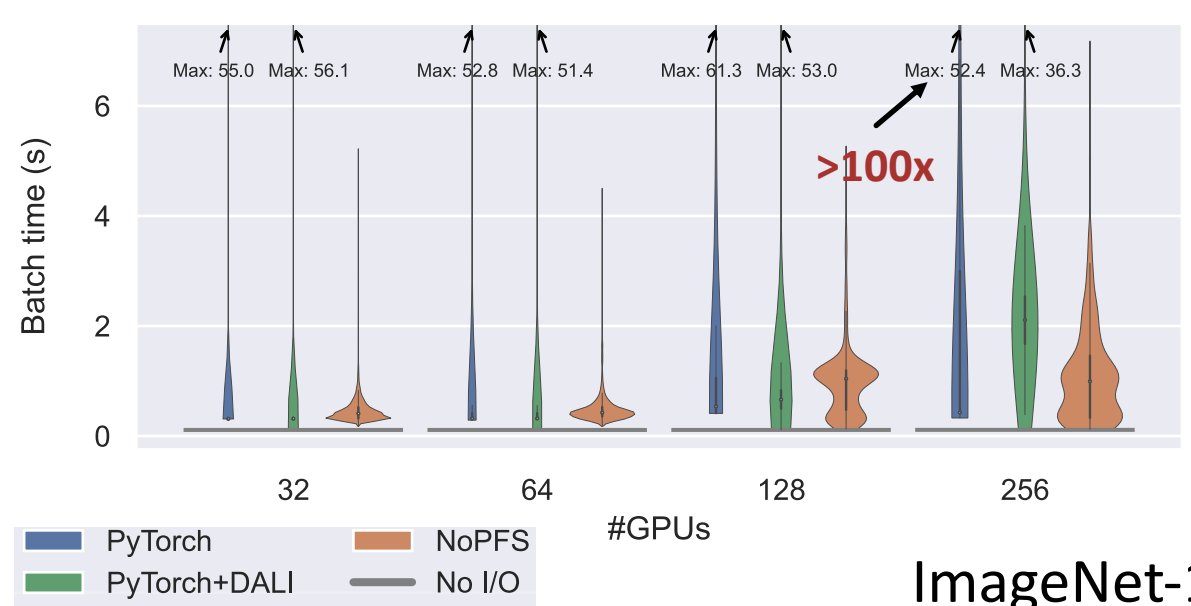


ImageNet-1k / ResNet-50

Performance

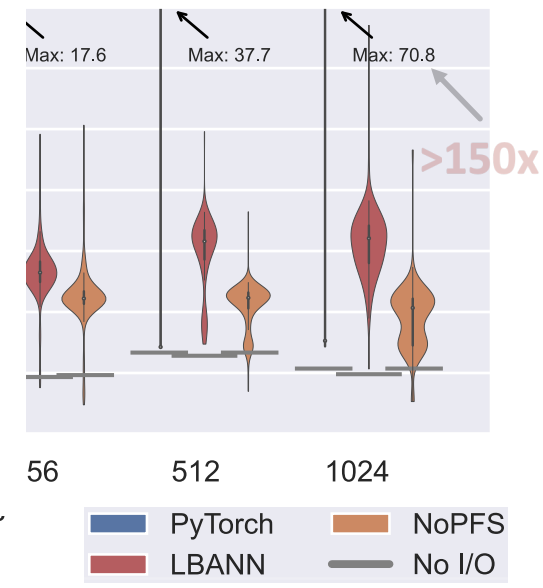
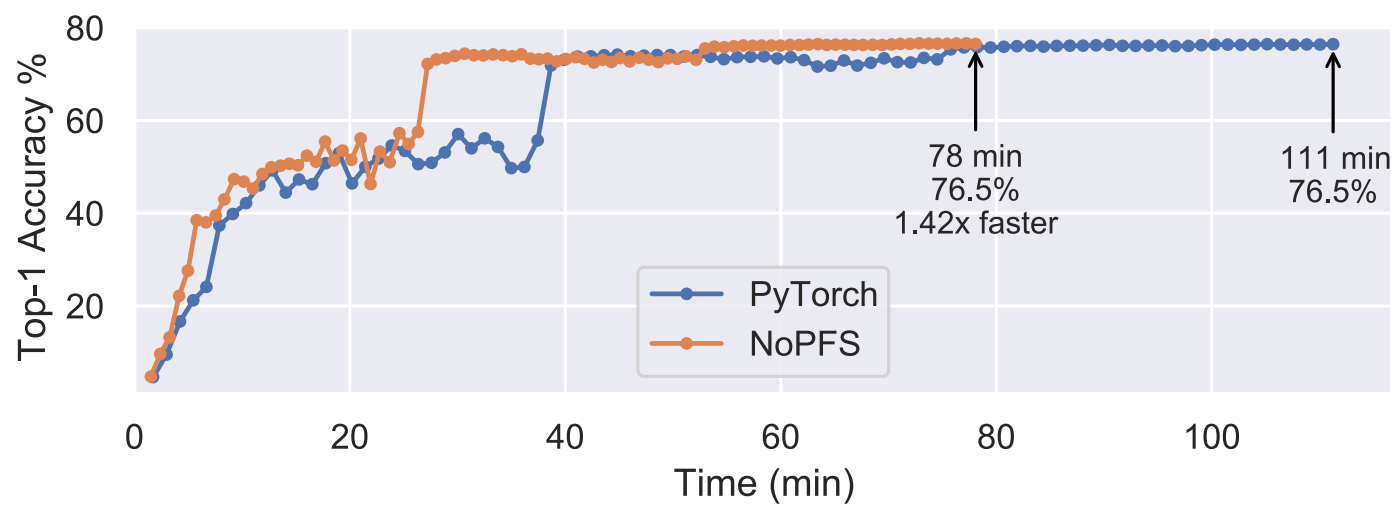
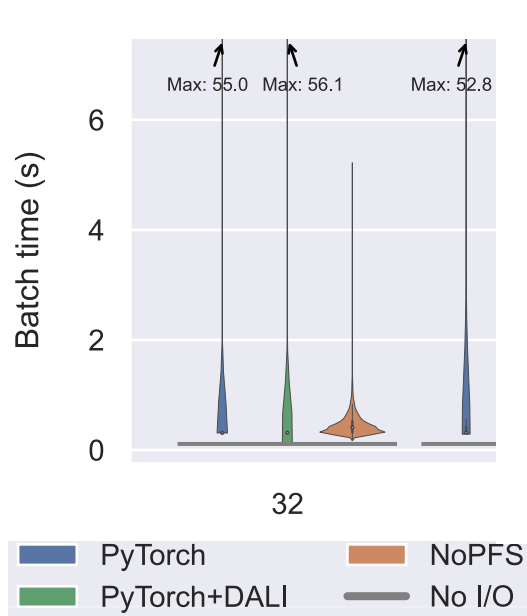


Runtime per batch



ImageNet-1k / ResNet-50

Performance

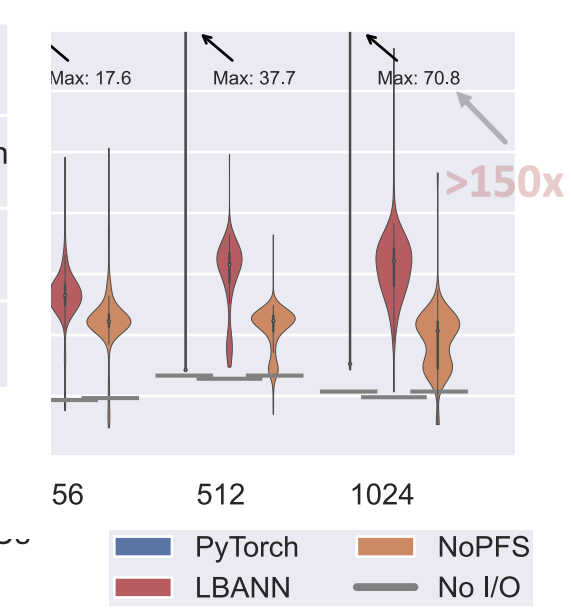
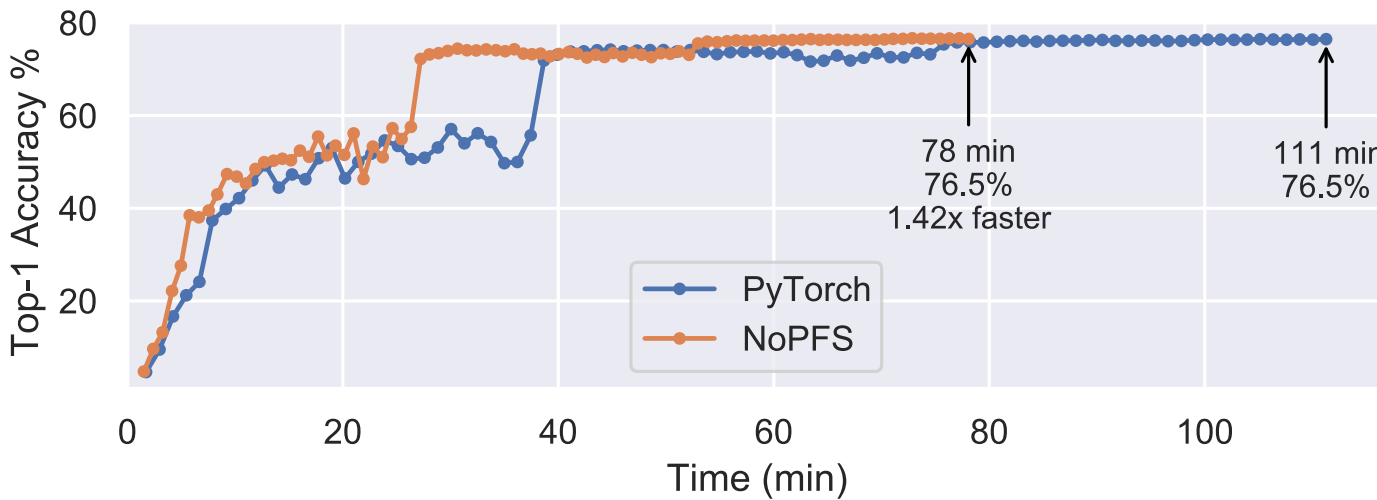
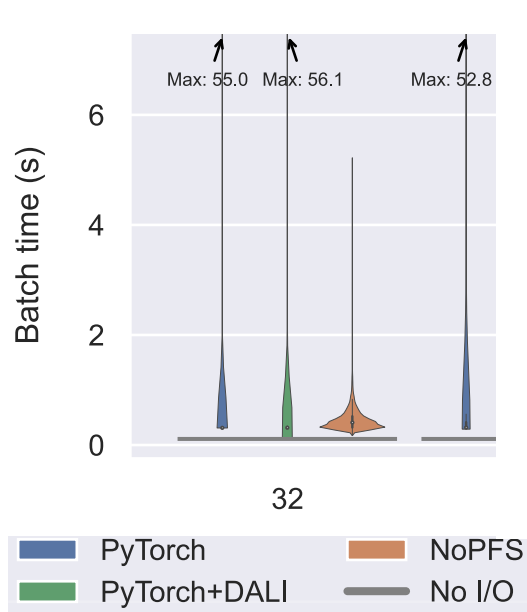


ImageNet-1k / ResNet-50

Performance



NoPFS improves performance and reduces noise across systems and scales

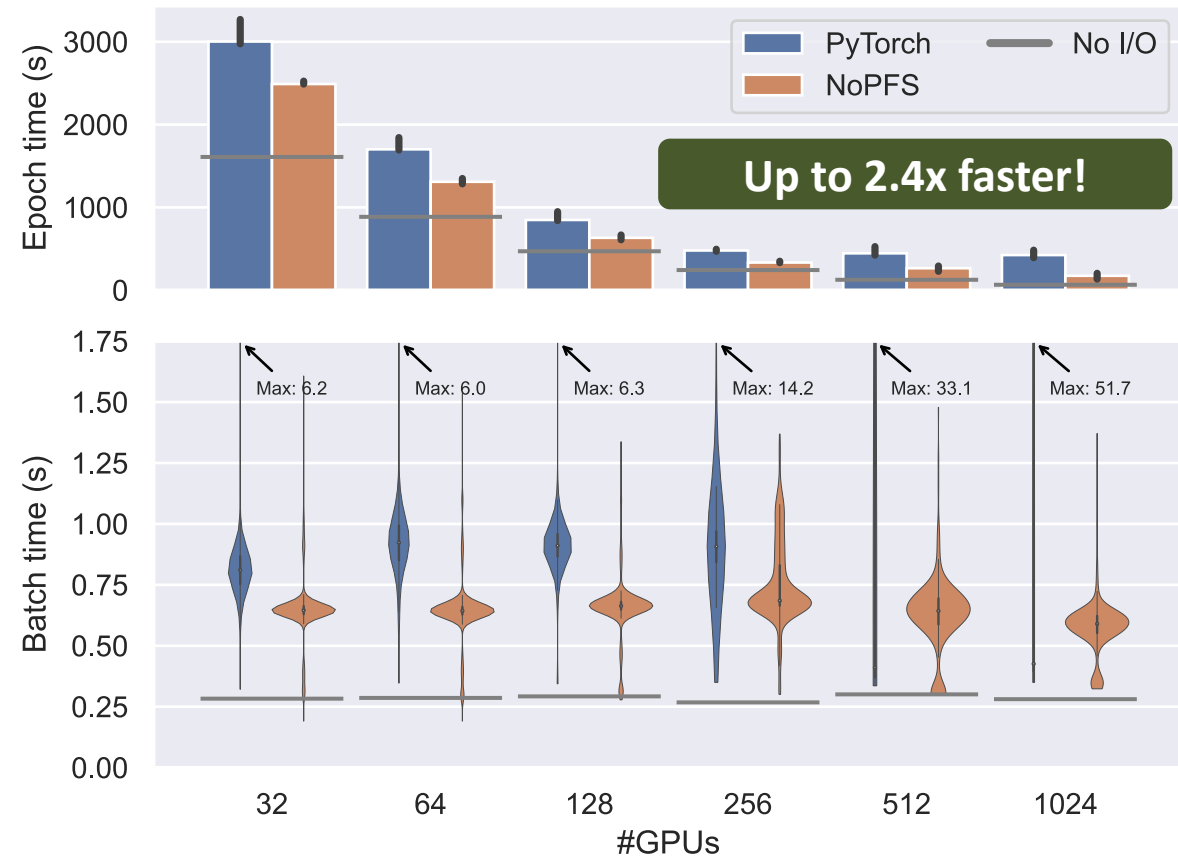


ImageNet-1k / ResNet-50

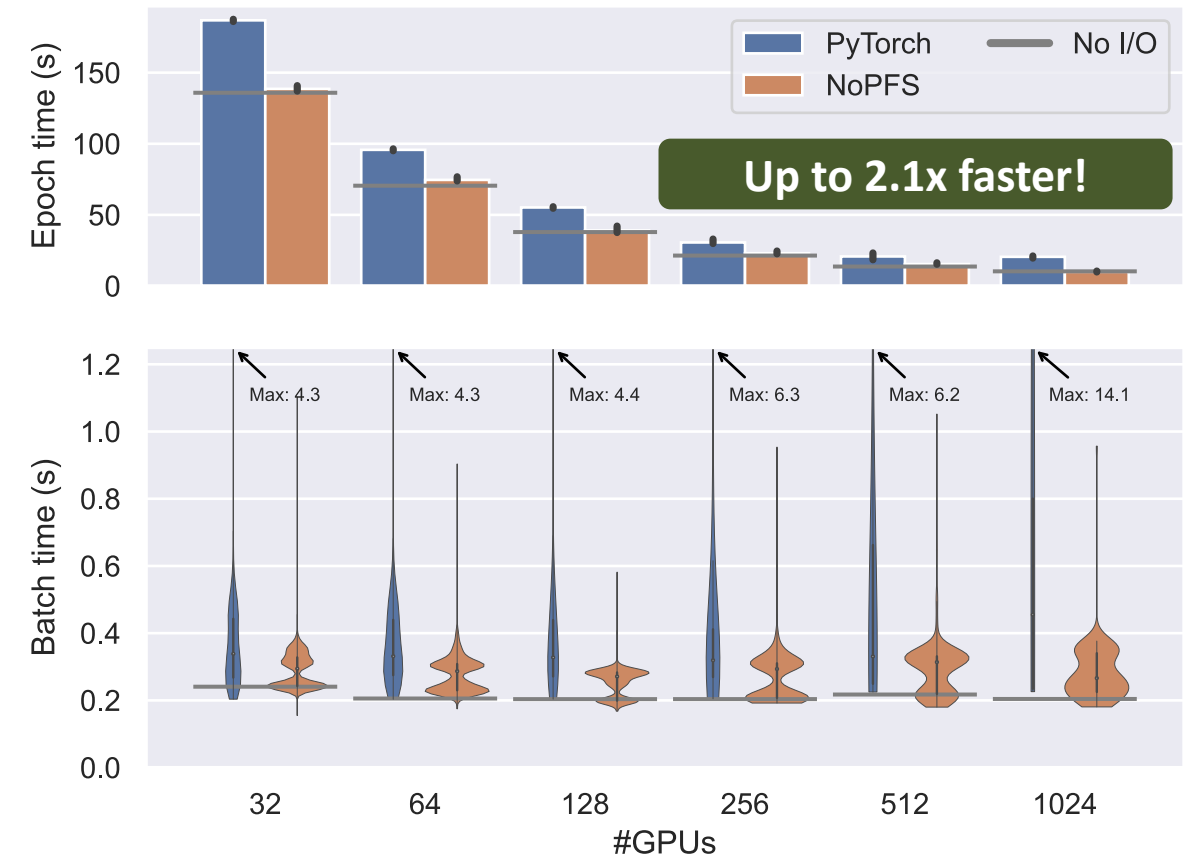
Performance: Going Bigger



ResNet-50 / ImageNet-22k (1.5 TB)



CosmoFlow (4 TB)



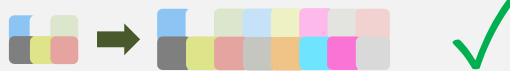
Conclusions

NoPFS is a hierarchical, distributed cache and prefetcher that knows the future

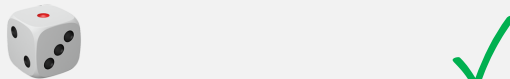
System Scalability



Dataset Scalability



Full Randomization

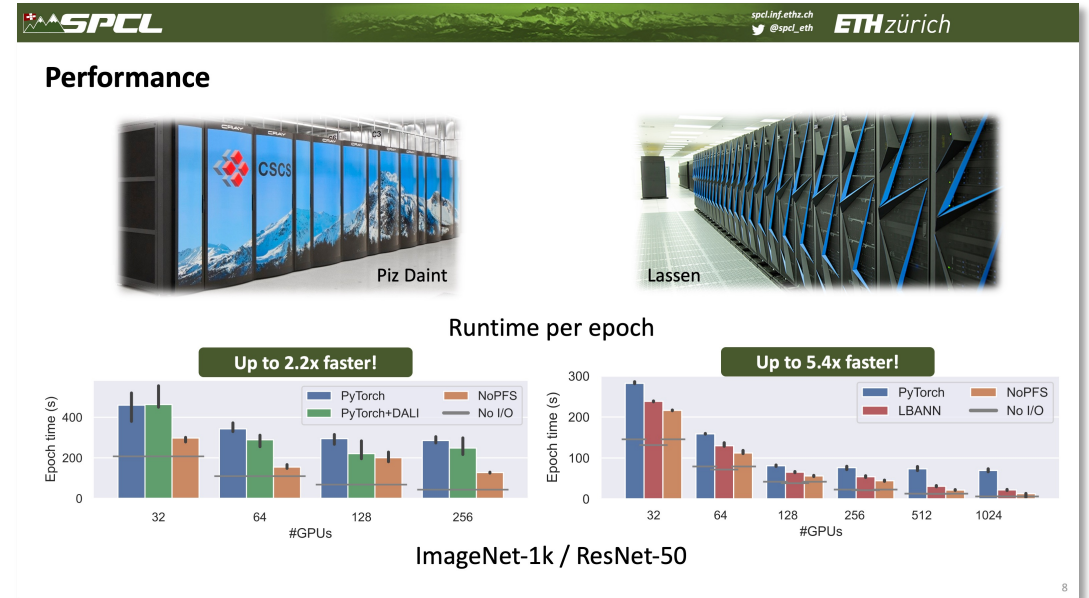


Hardware Independence



Ease of Use

```
dataset = ImageFolder(data_dir, data_transforms)
dsampler = DistributedSampler(dataset, num_replicas=n, rank=rank)
dataloader = DataLoader(dataset, batch_size, sampler=dsampler)
```



<https://github.com/spcl/nopfs>