Why globally re-shuffle? Revisiting data shuffling in large scale Deep Learning

Per3S 2023

François Trahay

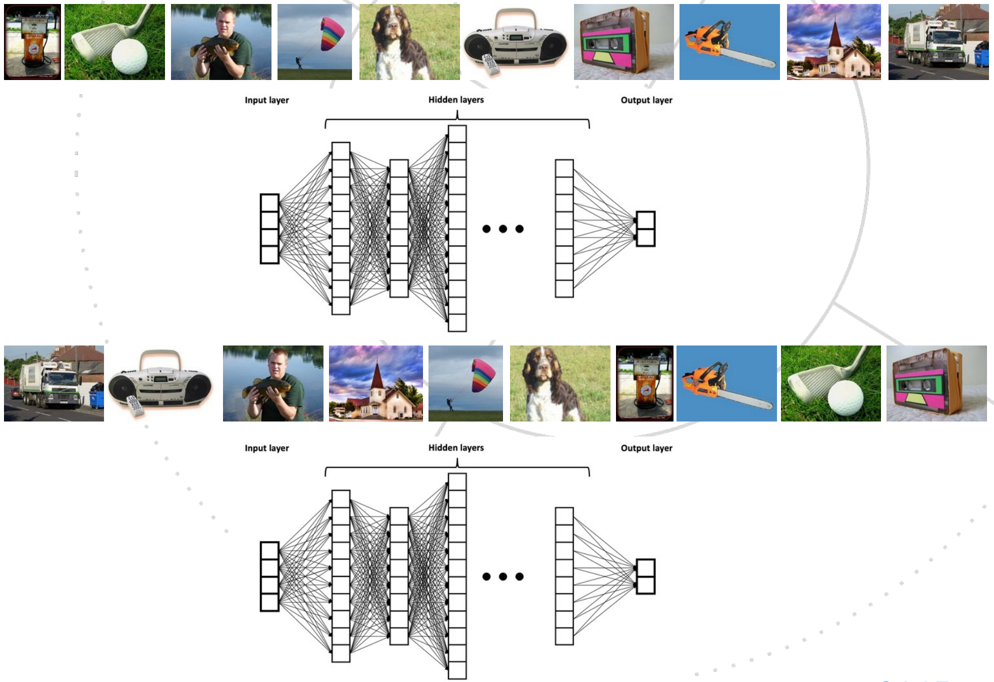
Télécom SudParis, Institut Polytechnique de Paris

ip-paris.fr

Original paper:
T. Truong, F. Trahay, J. Domke, A. Drozd, E. Vatai, J. Liao, M. Wahib, B. Gerofi.
Why Globally Re-shuffle? Revisiting Data Shuffling in Large Scale Deep Learning. IPDPS 2022.

Stochastic Gradient Descent

- **In each training iteration:**
  - Select mini-batch size samples *randomly*
  - Evaluate mini-batch
  - Compute gradients
  - Update model



Large scale machine learning

- **AI data set sizes are increasing rapidly**
  - CIFAR100: 170 MB
  - ImageNet: 140 GB
  - ImageNet21K: 1.8 TB
  - OpenCatalyst: 1.1TB

    → may not fit in memory/on disk

- **AI models size increase too**
  - ResNet18: 11 M parameters
  - ResNet50: 23 M parameters
  - GPT3: 175B parameters
  - → training time increases (eg. ResNet18/ImageNet: 58h, ResNet50/ImageNet: 336 h [1])

    → fitting the model in memory requires dozens of GB of RAM

- **Need for distributed Stochastic Gradient Descent (SGD)**
  - To speed up training
  - To fit the data set in memory

[1] Y. You, Z. Zhang, C.J. Hsieh, J. Demmel, & K. Keutzer. *Imagenet training in minutes.* In ICPP 2018

Distributed SGD

- **In each training iteration, each worker:**
  - Selects **local** mini-batch size samples ***randomly***
  - Evaluates its **local** mini-batch
  - Computes gradients
  - **AllReduce to average gradients across workers**
  - Updates model

First tier storage on the TOP500 vs. DL data sets

- **Three patterns:**
  - No local storage
  - Compute node local SSD (dark blue in Fig 1)
  - Network attached flash (light blue in Fig 1)

- **Where should I store the data set ?**
  - ~~On local SSDs~~
  - Access data from the parallel file system
  - Split data set among workers and sample locally

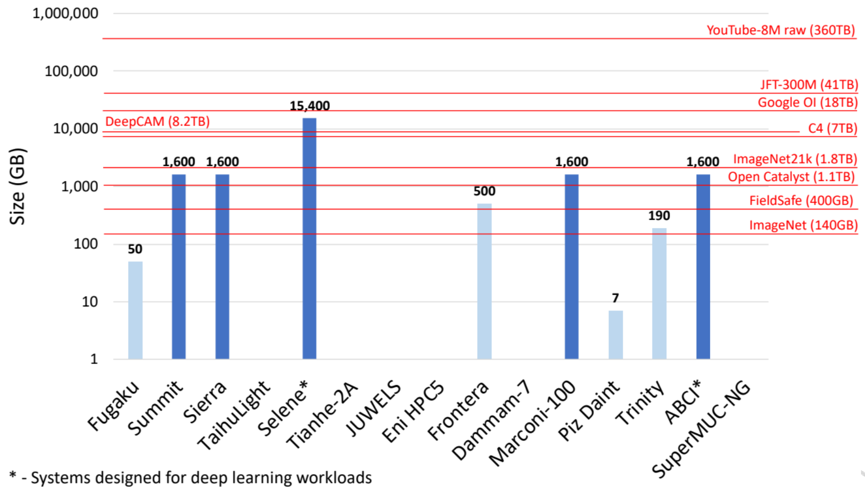


Fig.1 Dedicated node local storage on the fifteen fastest supercomputers from the TOP500 list (21' Jun) vs. DL data set sizes

Accessing the dataset from a parallel filesystem

I/O performance measured during training

- Experiments on ABCI supercomputer
- ResNet-50 on ImageNet-1k running on 64 nodes (256 GPUs)
  - Local: each worker reads from its local SSD (dataset is duplicated)
  - Parallel: workers read from the parallel filesystem (Lustre)
- I/O logged with Darshan DXT

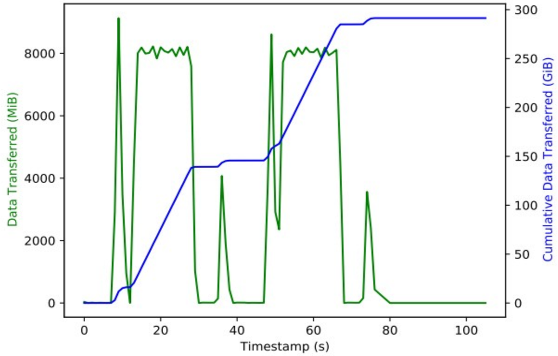
Observations

- Local SSD:
  - Bursts of read at each epoch
  - Read throughput is stable (8GB/s)
- Parallel filesystem:
  - Read throughput is unstable (avg: 3GB/s)
  - Concurrent random read significantly degrade the IO performance
- → training from the local SSD is ~70% faster than from the PFS

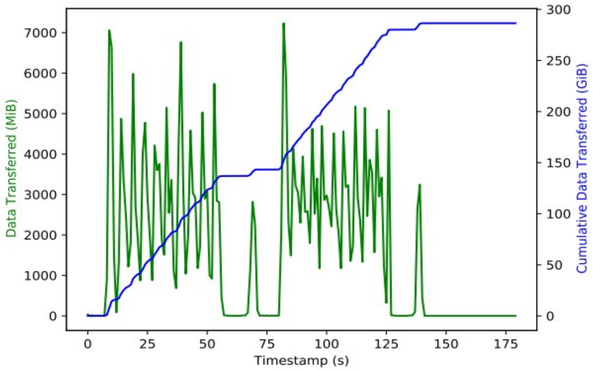
Where should I store the data set ?

- ~~On local SSDs~~
- ~~Access data from the parallel file system~~
- Split data set among workers and sample locally

Local SSD



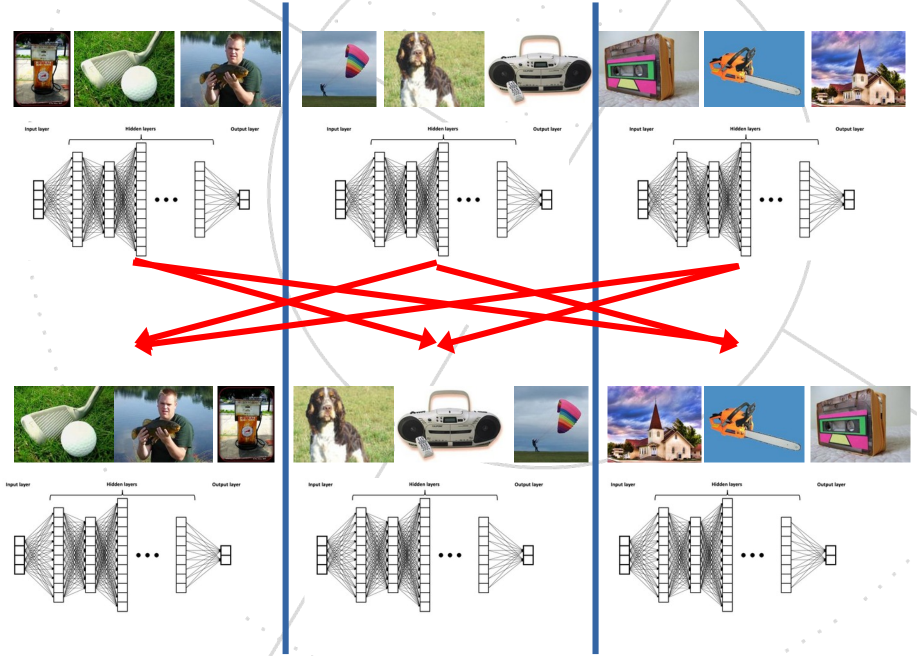
Parallel filesystem



Local shuffling strategy

- **In each training iteration, each worker:**
  - Selects local mini-batch size samples ***randomly*** from **local samples**
  - Evaluates its local mini-batch
  - Computes gradients
  - AllReduce to average gradients across workers
  - Updates model

- Each worker trains on a (small) subset of the dataset
  → Fast (local) I/O
  → What is the impact on accuracy of training from a few samples ?



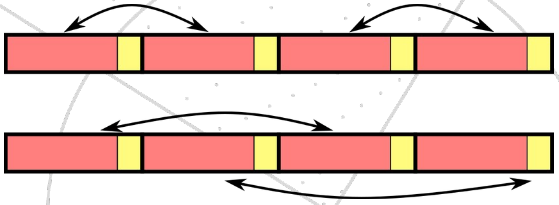
Local-partial shuffling

- **Proposal:** **keeping samples mostly local**
  - Exchange a portion Q of local samples with another node
    - Q=0 → local shuffling
    - Q=1 → global shufflling

- **Executions steps:**
  - N samples initially distributed among M workers
  - Randomly pick Q x N/M samples (global partition)
  - Exchange of samples between random pairs of workers

- **Implementation in PyTorch**
  - Replacement to DistributedSampler()

- **To overlap forward and backward paths:**
  - Use non-blocking MPI calls (i.e., MPI_Isend/recv())



Training code with global shuffling

```
train_dataset = ImageFolder(train_dir, transformations)
train_sampler = DistributedSampler(train_dataset, size, rank)
train_loader = DataLoader(train_dataset, batch_size=b, train_sampler)
```

Training code with (Partial) Local Shuffling

```
train_dataset = PLS.ImageFolder(train_dir, class_file, transformations)
train_sampler = DistributedSampler(train_dataset, size, rank=rank)
train_loader = DataLoader(train_dataset, batch_size=b, train_sampler)
scheduler = PLS.Scheduler(train_dataset, batch_size=b, fraction=Q)
...
train(epoch):
    scheduler.scheduling(epoch)
    ...... # Training loop here
    send_req, recv_req = scheduler.communicate() # Non-blocking exchange
    scheduler.synchronize(send_req, recv_req) # Wait to finish exchange
    scheduler.clean_local_storage() # Remove exchanged samples on the storage
```

Fig. 3: Global vs. partial local sampling in PyTorch.

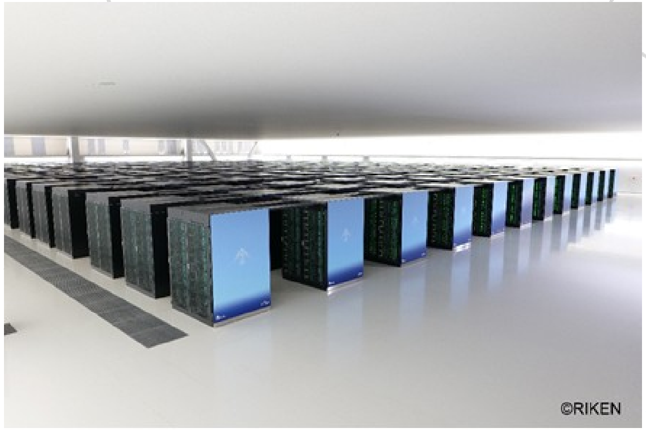
Experiments

- **What is the impact of shuffling strategies**
  - **on accuracy ?**
  - **on performance ?**

Evaluation Platforms

- **ABCI (#16 on Top500)**
  - 1,088 compute nodes (CN)
  - 2 Intel Xeon Gold + 4 NVIDIA V100 per node
  - Infiniband EDR
  - 1.6 TB SSD local per CN
  - 4 workers per CN

- **Fugaku (#1 on Top500)**
  - 158,976 CNs
  - Fujitsu A64FX CPU (48 cores, 4 NUMA domains) per node
  - 1.6 TB SSD shared among 16 nodes
  - 2 workers per CN



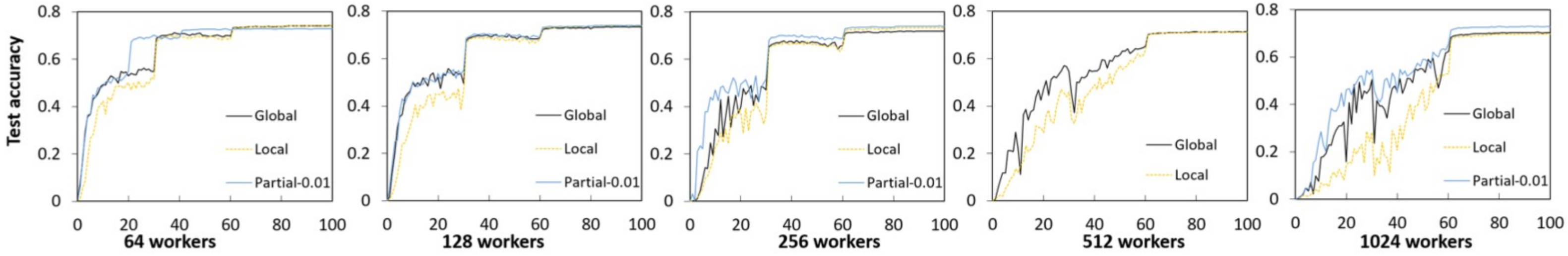
Models, Data Sets and Configurations

- **Evaluated models/datasets**
  - ResNet50 with ImageNet-1k
  - DenseNet with ImageNet-1k
  - WideResNet-28-10 with CIFAR100
  - ResNet50 (pretrained) with StanfordCar dataset
    - Pre-trained, transfer learning scenario
  - ResNet50 with ImageNet-50 (ABCI)
    - ImageNet-50 is a subset of ImageNet-1K with only 50 classes
  - Inception-v4 with CIFAR100 (ABCI)
  - ImageNet-21k with Resnet50
  - DeepCAM
- **Evaluated configuration**
  - **Local** sampling
  - **Global** sampling
  - **Partial-0.x**: local-partial shuffling with an x% exchange

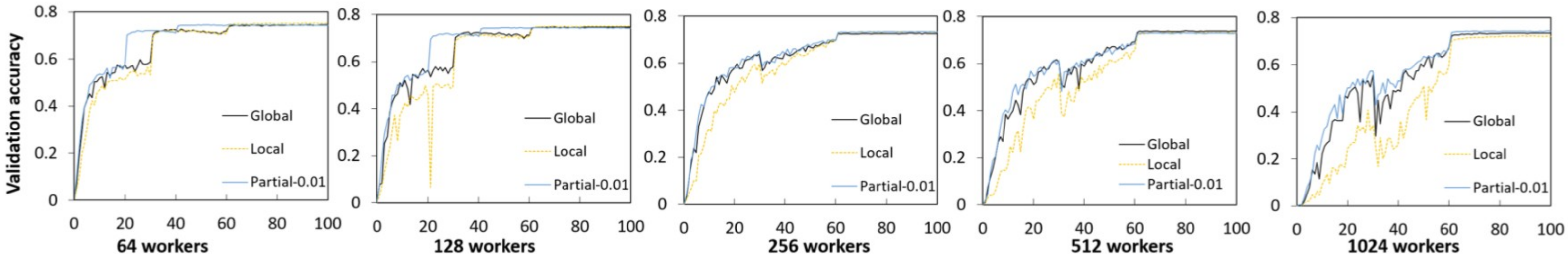
TABLE I: Datasets and Models Used in Experiments (*)Trained on a subset of the original dataset. (**) Use pre-trained model.

Model	Dataset	#Samples	Size
Resnet50 [26]	ImageNet-1K [9]	1.2M	~ 140GB
Densenet161 [27]			
Resnet50 [26]	ImageNet-50(*) [9]	~65K	~ 2GB
WideResNet-28-10 [28]	CIFAR-100 [29]	50K	~160 MB
Inceptionv4 [30]			
Resnet50 (**) [26]	Standford Cars [31]	8144	~ 934 MB
Resnet50 [26]	ImageNet-21K(*) [9]	~ 9.3M	~1.1 TB
DeepCAM [26]	DeepCAM [1]	~ 122K	~8.2 TB

Local Shuffling is sufficient



(a) ResNet50 with ImageNet-1K

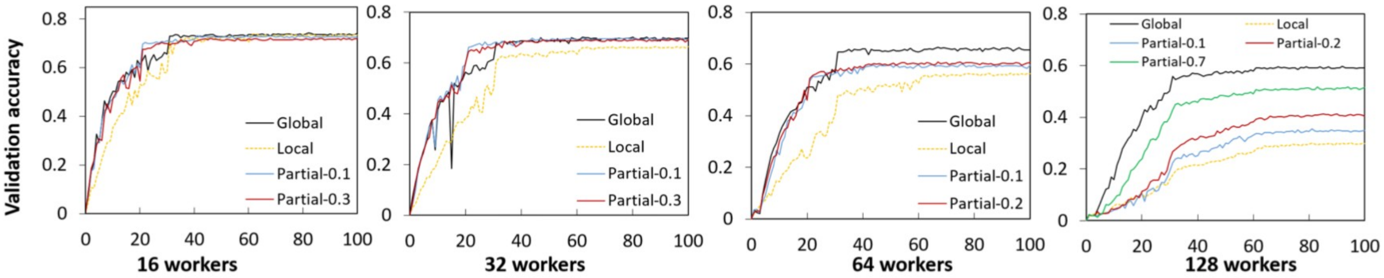


(b) DenseNet with ImageNet-1K

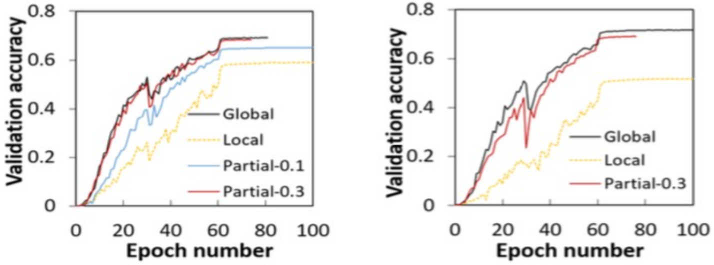
- Both local shuffling and global achieve the same validation accuracy
- Local partial shuffling provides almost identical accuracy trajectory with global sampling

  → **local-partial shuffling could reduce the run time**

Partial Local Shuffling can improve Accuracy



(a) ResNet50 with ImageNet50 (ImageNet subset that includes 50 classes)



(a) 2048 workers (b) 4096 workers

ResNet50 with ImagetNet-1K on Fugaku

- For some models/datasets, local shuffling degrades the accuracy
- Partial local shuffling maintains the same accuracy as global shuffling

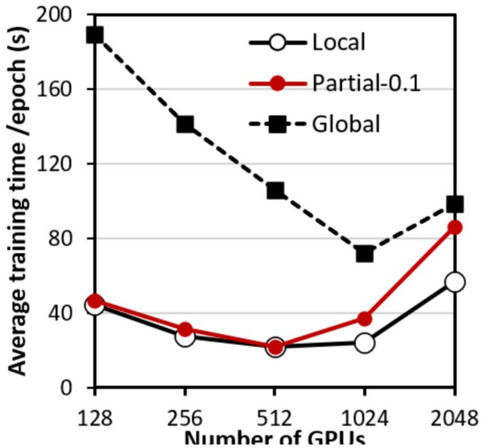
- Partial local shuffling reduces the need for local storage
  - Partial-0.01 : Each of the 4096 workers store only ~**0.03%** of the data set locally

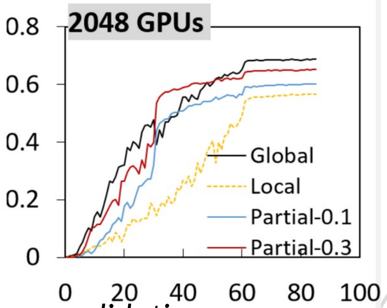
→ Achieving good accuracy for datasets that don't fit on the local storage

Performance analysis

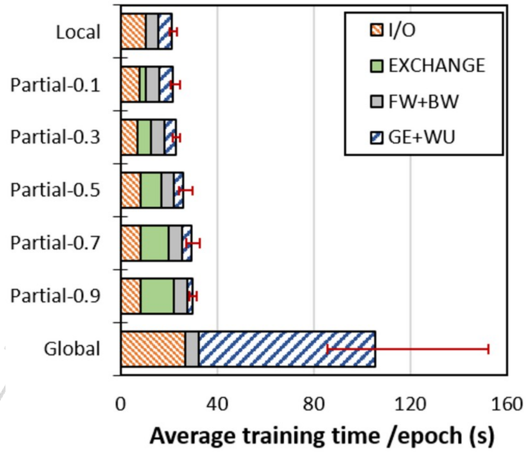
- **Resnet50 on ImageNet1k:**
  - Good scalability for up to 1k GPUs, performance drop on 2k
  - Still significant improvement compared to global out of PFS

- **Partial-local** shuffling
  - Cost of sample exchange increases as Q increases
- **Global** shuffling
  - Cost of I/O is high
  - High variability due to I/O contention





validation accuracy



Summary and Takeaways

- **Global shuffling in distributed DL has severe I/O implications**

- **Impact of randomization on accuracy is unclear**
  - Only significant for large number of workers ?
- **Built a library that enables experimentation with the proposed partial local shuffling**
  - Often local shuffling provides similar accuracy as global
  - In some cases partial shuffling can improve accuracy and approximate that of global shuffling (with the fraction of the cost of running out of PFS)

- **Future work:**
  - Redundant data storage to avoid communication in partial shuffling
  - Cost model for dynamically determining the right fraction (Q) to shuffle
  - More experiments, more datasets (e.g., OpenCatalyst)
  - (ongoing) select the important samples to exchange → almost local shuffling with high accuracy