Introduction to Distributed/Federated Machine Learning

Li Ju
PhD student@TDB
li.ju@it.uu.se

The Optimization Problem $\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$

How to solve: first order methods, e.g. gradient descent:

Iteratively, at step t: $\theta_t = \theta_t - \eta \nabla_\theta \sum_{n=1}^{N} \ell(x_n; \theta_{t-1})$

* In practice, we use stochastic gradient descent (sgd):

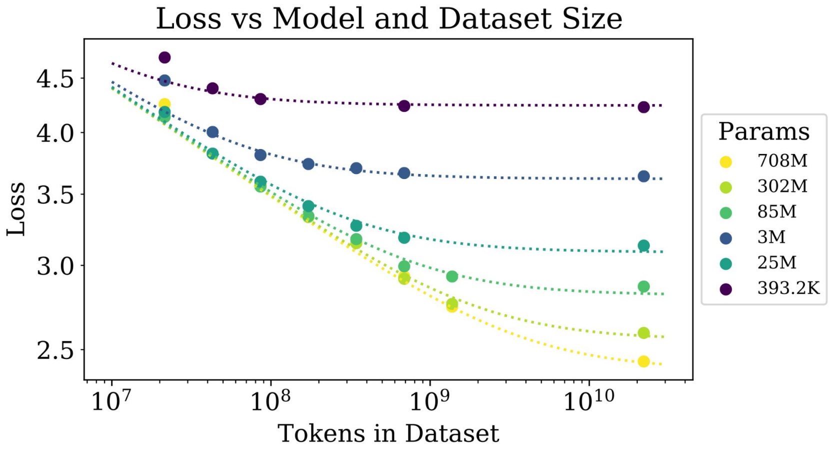
$$\nabla_\theta \sum_{n=1}^{N} \ell(x_n; \theta) \approx \nabla_\theta \sum_{x \in \mathcal{B}} \ell(x; \theta)$$

where $\mathcal{B} \sim \{x_n\}_{n=1}^{N}$

Why Distributed ML

$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

Bigggggger models and datasets bring better performance (Scaling Law)



Organization ⇅	Model ⇅	License ⇅	Parameters (B) ⇅
	o3	Proprietary	-
	Claude 3.7 Sonnet	Proprietary	-
	Grok-3	Proprietary	-
	Grok-3 Mini	Proprietary	-
	o3-mini	Proprietary	-
	o1-pro	Proprietary	-
	o1	Proprietary	-
	Gemini 2.0 Flash Thinking	Proprietary	-
	o1-preview	Proprietary	-
	DeepSeek-R1	Open ⓘ	671
	GPT-4.5	Proprietary	-

Why Distributed ML

$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

Physical infeasibility

GPU RAM required:

~6 TB (batch size: 512)

Best GPU:

Nvidia H100, with 80GB

We need ~ 80 H100

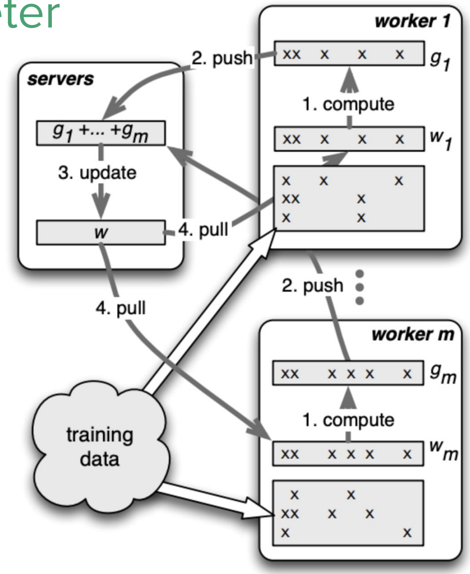
Organization ⇅	Model ⇅	License ⇅	Parameters (B) ⇅
	o3	Proprietary	-
	Claude 3.7 Sonnet	Proprietary	-
	Grok-3	Proprietary	-
	Grok-3 Mini	Proprietary	-
	o3-mini	Proprietary	-
	o1-pro	Proprietary	-
	o1	Proprietary	-
	Gemini 2.0 Flash Thinking	Proprietary	-
	o1-preview	Proprietary	-
	DeepSeek-R1	Open ⓘ	671
	GPT-4.5	Proprietary	-

How to do Distributed ML

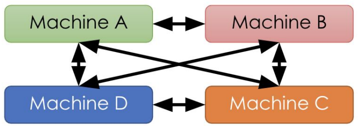
$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

Data parallelization: split batch across M GPUs $\mathcal{B} = \cup_{m=1}^{M} \mathcal{B}_m$

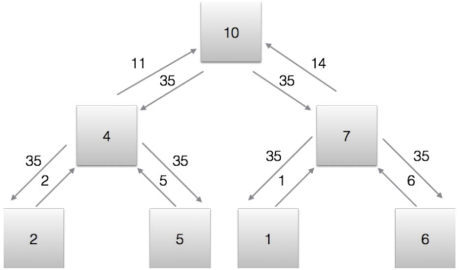
Parameter server



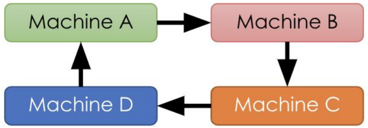
Allreduce



Tree Allreduce



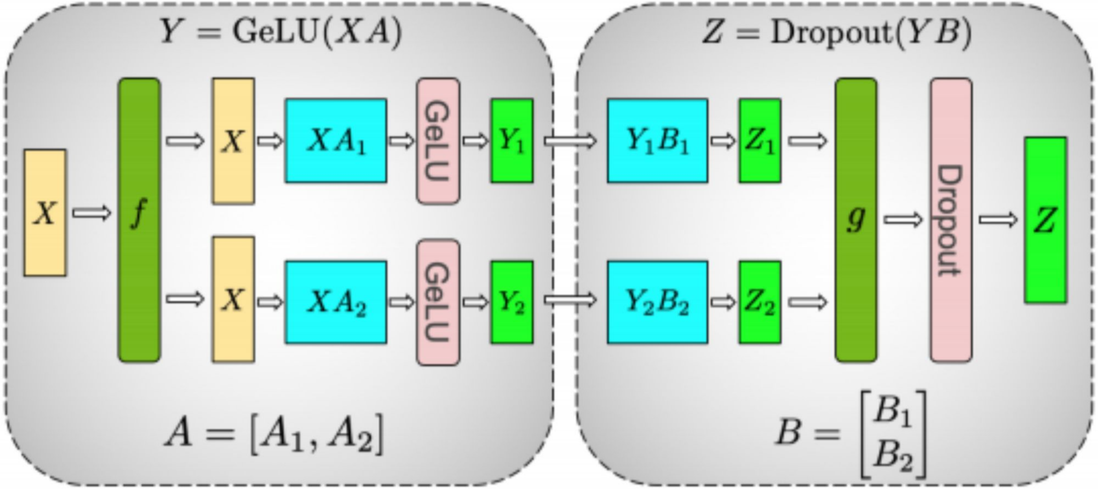
Ring Allreduce



How to do Distributed ML

$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

Model parallelization?



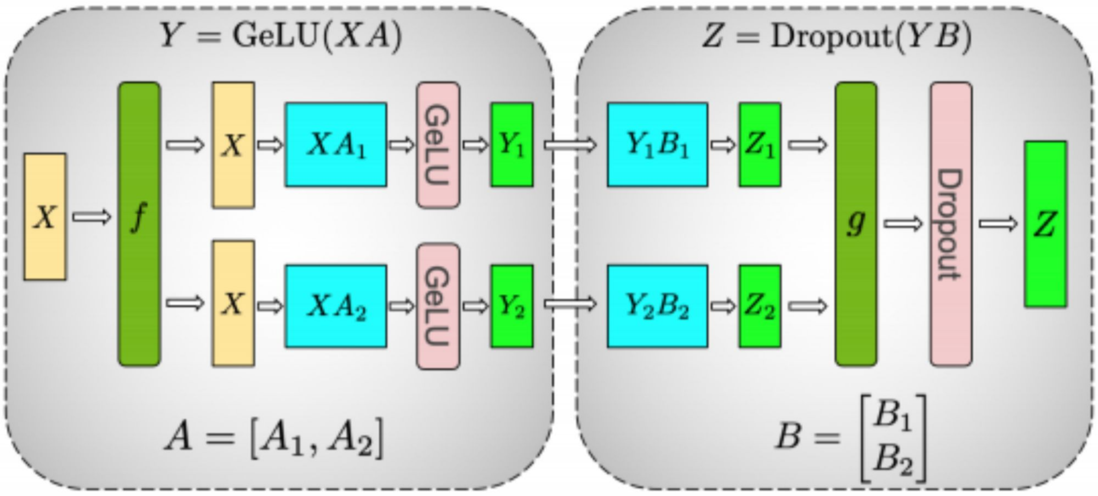
GPU 0

GPU 1

How to do Distributed ML

$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

Model parallelization?



GPU 0

GPU 1

This is NOT model parallelization!

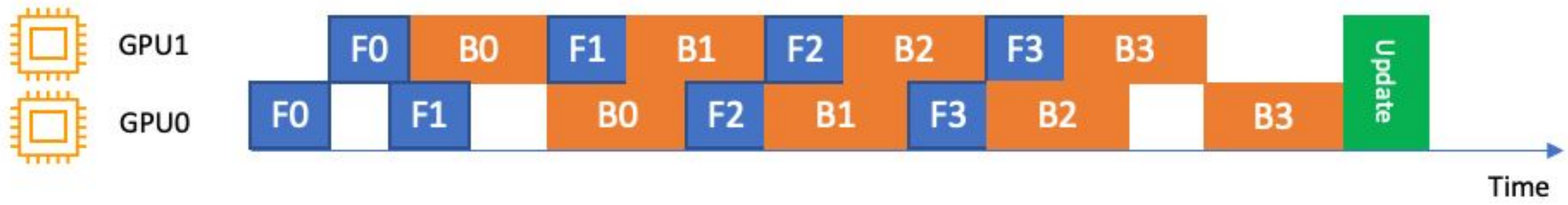
Workloads on GPUs depend on each other...

It is called workload partitioning.

How to do Distributed ML

$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

Pipeline parallelization:



* The batch of data needs to be split into mini batches.

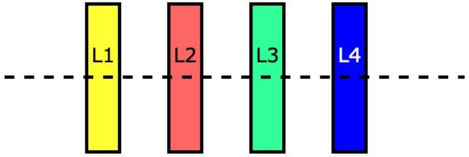
How to do Distributed ML

Model Parallelization:

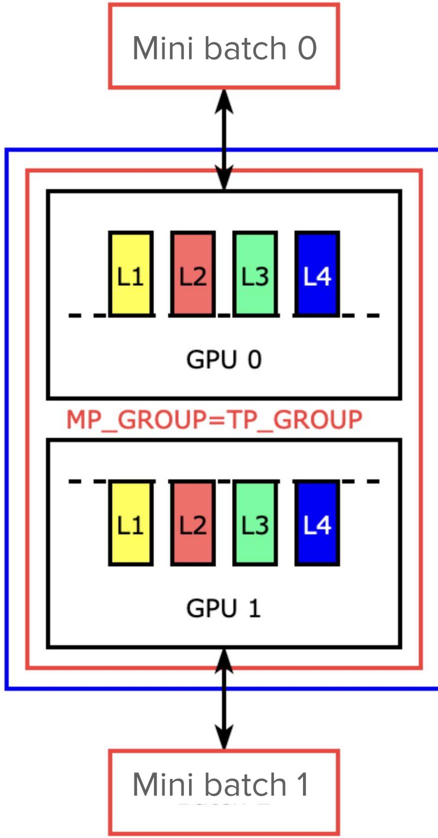
- Zero Redundancy Optimizer (ZeRO)
- Tensor parallelization
- …

ZeRO is the most common approach.

* The batch of data needs to be split into M mini batches.



Sample model with four layers



How to do Distributed ML

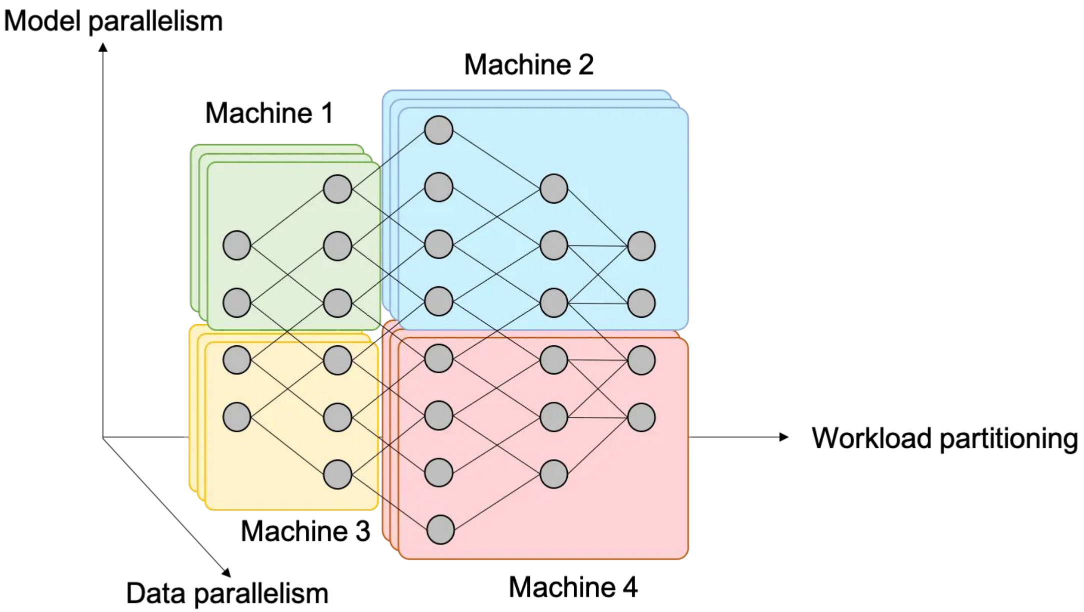
$$\arg_\theta \min \sum_{n=1}^{N} \ell(x_n; \theta)$$

3D-parallelization

Data, pipeline and model

parallelization are orthogonal to

each other:



Backend

Taking PyTorch as an example:

MPI: CPU

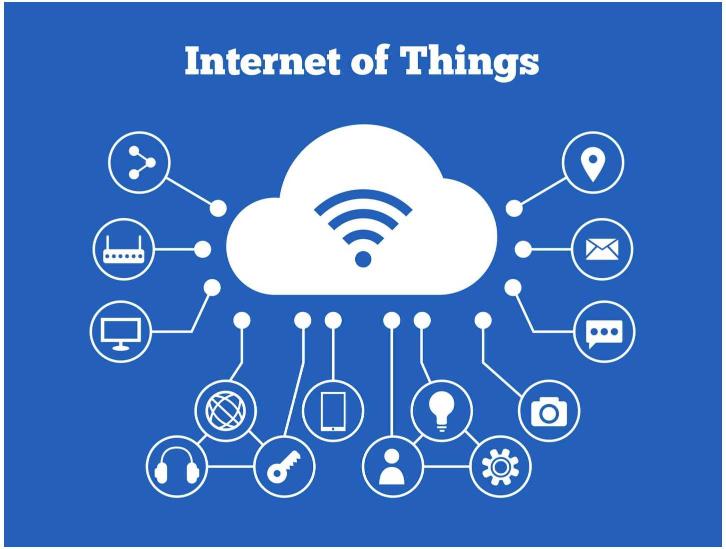
NCCL: Nvidia GPU

GLOO: Both (partially)

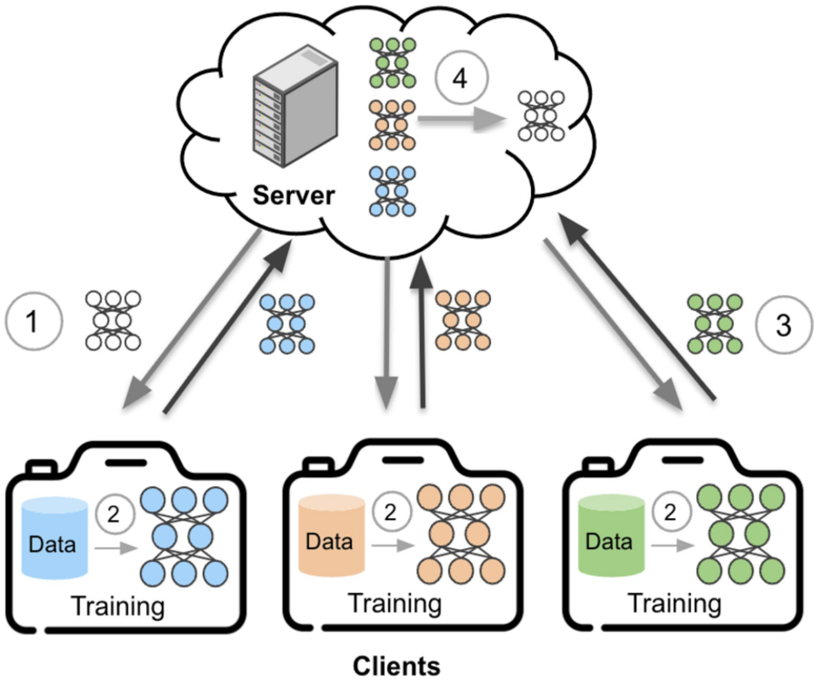
Backend	`gloo`		`mpi`		`nccl`	
Device	CPU	GPU	CPU	GPU	CPU	GPU
send	✓	✘	✓	?	✘	✓
recv	✓	✘	✓	?	✘	✓
broadcast	✓	✓	✓	?	✘	✓
all_reduce	✓	✓	✓	?	✘	✓
reduce	✓	✘	✓	?	✘	✓
all_gather	✓	✘	✓	?	✘	✓
gather	✓	✘	✓	?	✘	✓

Why Federated ML

Intrinsic distributed nature + privacy concern



How to do Federated ML



I. Initialize a seed model

II. Iteratively:

1. Broadcast model params
2. Training locally
3. Collecting local models
4. Aggregating and updating model

How to do Federated ML

Algorithm 1 Federated Averaging (FedAvg)

procedure FedAvg $(\mathbf{x}^{(0,0)}, \eta)$

for $r = 0, \ldots, R-1$ **do**

  on client for $m \in [M]$ **in parallel do**

    $\mathbf{x}_m^{(r,0)} \leftarrow \mathbf{x}^{(r,0)}$ $\triangleright$ broadcast current iterate

    for $k = 0, \ldots, K-1$ **do**

      $\xi_m^{(r,k)} \sim \mathcal{D}_m$

      $\mathbf{g}_m^{(r,k)} \leftarrow \nabla f(\mathbf{x}_m^{(r,k)}; \xi_m^{(r,k)})$

      $\mathbf{x}_m^{(r,k+1)} \leftarrow \mathbf{x}_m^{(r,k)} - \eta \cdot \mathbf{g}_m^{(r,k)}$ $\triangleright$ client update

  $\mathbf{x}^{(r+1,0)} \leftarrow \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m^{(r,K)}$ $\triangleright$ server averaging

K=1: Parameter-server data parallelization / Distributed SGD

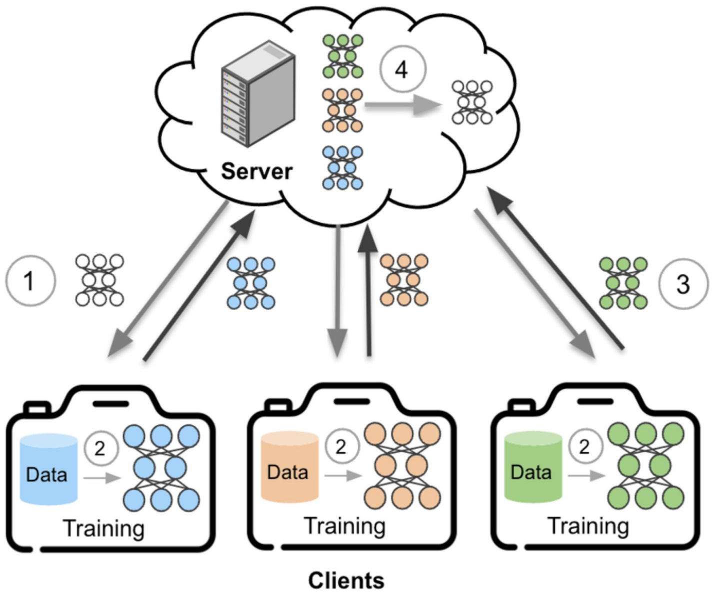
K>>1: Federated learning

Why FML is significant

Intrinsic mathematical issues: Distributed

SGD is equivalent to SGD, but FedAvg is not:

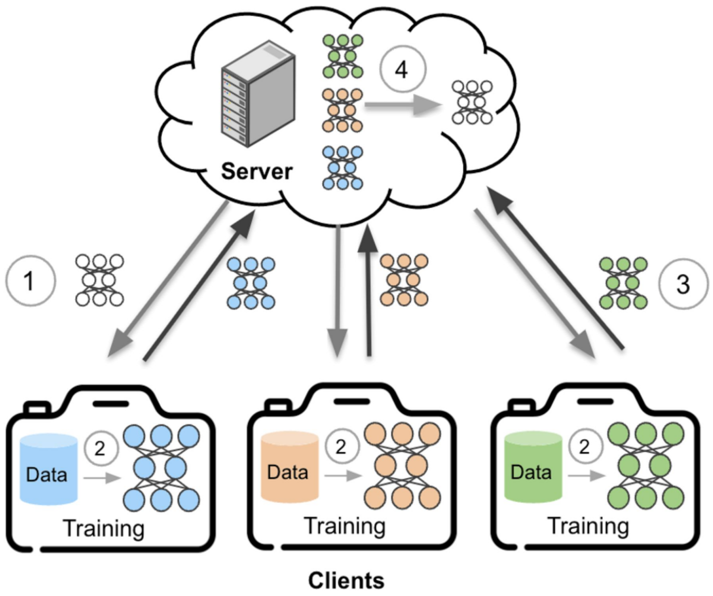
- Different local steps
- Size heterogeneity of local data
- Statistical heterogeneity of local data
- Convergence analysis
- ...



Why FML is significant

Practical/engineering issues:

- Extremely bottlenecked by communication
- Straggler problem
- Asynchronized FML
- Personalization
- Machine Unlearning (quitting participant)
- ...



Any Questions?