SaCS
Scalable computing systems laboratory

EPFL

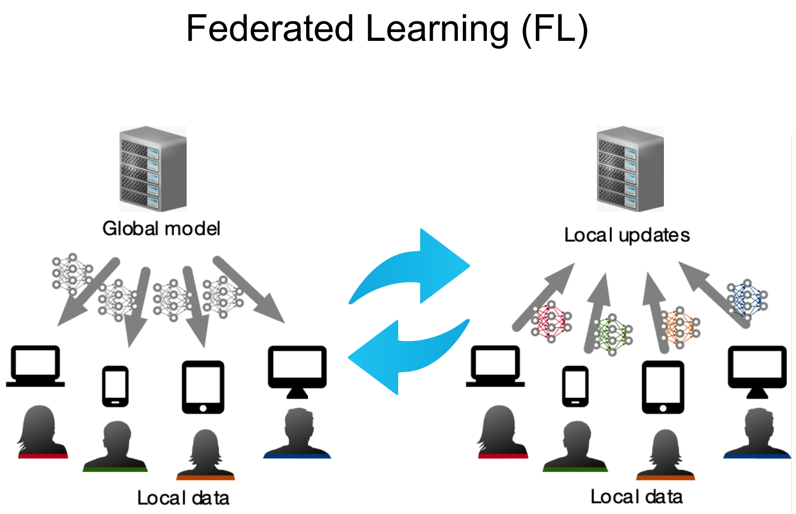
Decentralized Learning made Practical

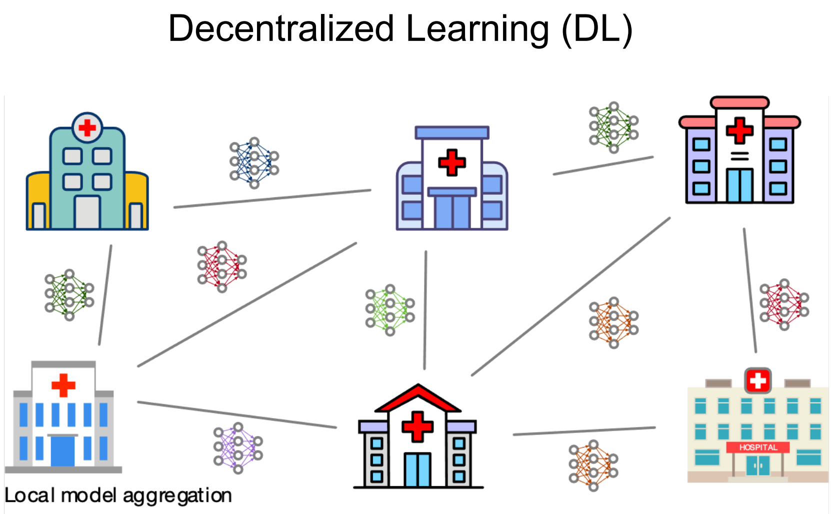
Martijn de Vos
martijn.devos@epfl.ch

In collaboration with Anne-Marie Kermarrec, Rishi Sharma, Akash Dhasade, Johan Pouwelse and Erick Lavoie.

Grenoble, 14.12.2023

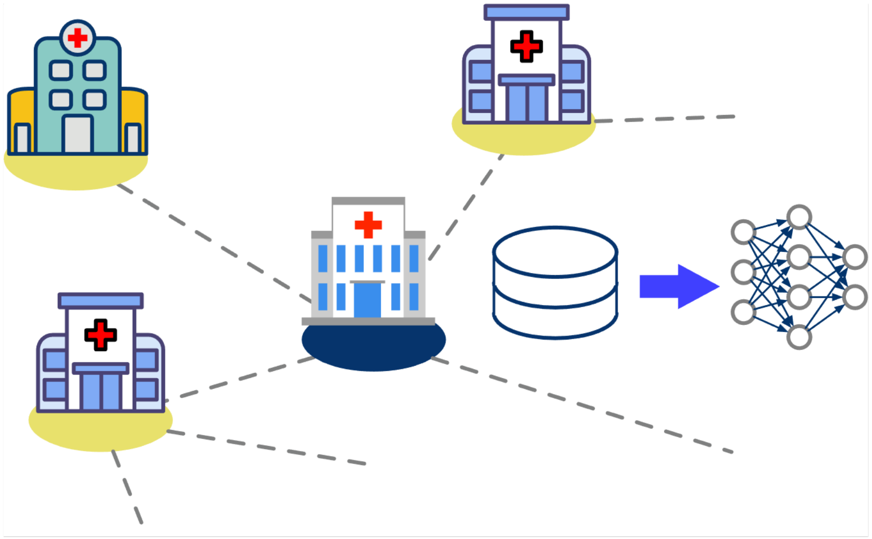
Distributed Learning with Decentralized Data





Data stay where it is generated. Learning happens by model exchange.

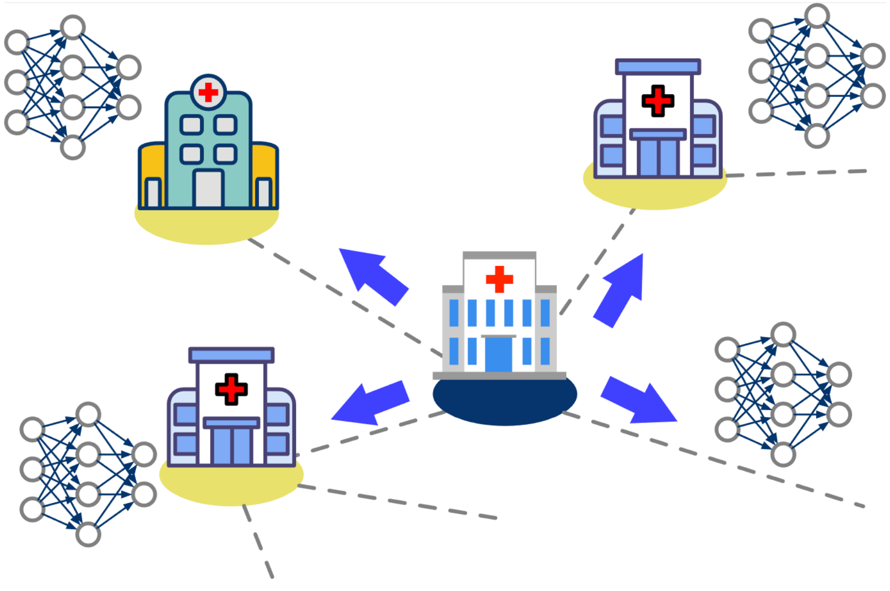
Decentralized Learning



Train

Share

Decentralized Learning



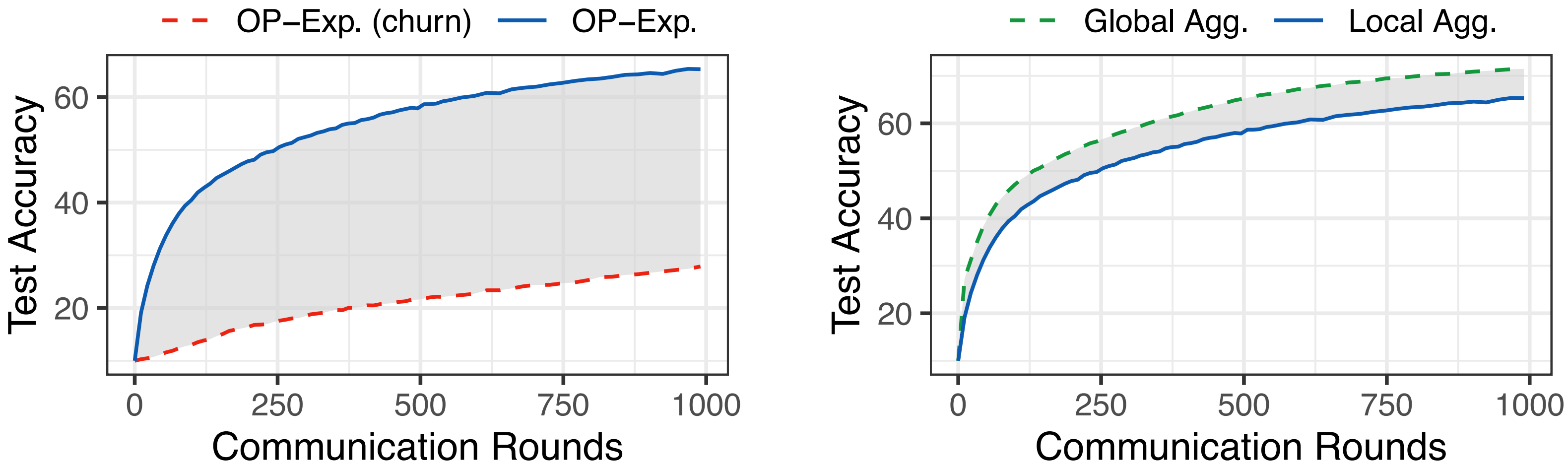
Receive

⇩

Aggregate

Improving Decentralized Learning

1. **Churn** impacts convergence
2. **Global aggregation** improves convergence
3. Not all nodes need to work each round (theory in [1])



[1] Liu, Ziwei, et al. "Decentralized stochastic optimization with client sampling." *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*. 2022.

The Design of Plexus

High-level algorithm

1. Each round, a unique subset of online nodes, or a **sample**, train the model.
2. These nodes send their trained model to an **aggregator.**
3. The aggregator aggregates models and sends the model to the next sample.

Resembles Federated Learning, but without server.

Three Technical Challenges (TCs)

TC1: How can nodes derive samples?

TC2: How can Plexus avoid selecting offline nodes during sampling?

TC3: How can Plexus ensure system progression when nodes go offline during training or aggregation?

TC1: How can nodes derive samples?

- Important that different nodes derive the same sample.
- Each peer stores all peer IDs in a **local view**.
- Each peer samples peer for the next round using its population view.
- Aggregator selected like this as well, based on bandwidth.

TC1: How can nodes derive samples?

Algorithm 1 Sampling by node i where k denotes the round number and s is the requested sample size.

Require: Ping timeout Δt_p

procedure $\textsc{Sample}(k, s)$
    /* $\textsc{Actives}()$ are the online nodes in local views */
    $H \leftarrow \textsc{sort}([\textsc{hash}(j + k)$ **for** j **in** $\textsc{Actives}()])$
    $C \leftarrow [j$ **for** h_j **in** $H]$ ▷ Candidate identifiers
    return the first s in C that answer a ping within Δt_p

TC2: How can Plexus avoid selecting offline nodes during sampling?

- Nodes send a join or leave message to other random nodes.
- Nodes keep track of the membership status of other nodes in their local view.

- Local views are gossiped and merged between nodes.

ID	Seq. no.	Status
3b9f8	2	LEAVE
u7nk3	3	JOIN
a2o8g	2	JOIN

Local view of node a

+

ID	Seq. no.	Status
3b9f8	2	LEAVE
u7nk3	4	LEAVE
a2o8g	1	LEAVE

Local view of node b

ID	Seq. no.	Status
3b9f8	2	LEAVE
u7nk3	4	LEAVE
a2o8g	2	JOIN

After merge

TC3: How can Plexus ensure system progression when nodes go offline during training or aggregation?

<u>Participant failure</u>

- Aggregator proceeds when
  - Received $f < s$ trained models
  - After some aggregation timeout

<u>Aggregator failure</u>

- Aggregator sends ACK message to previous participants when finished.
- Participants await ACK message
  - Retry with another aggregator after some timeout.

Experiment Setup (1/2)

- Implemented Plexus in Python 3 using PyTorch.
- Evaluation on the DAS6 compute cluster.

- Metrics:
  1. Time-to-accuracy
  2. Communication-to-accuracy
  3. Training-resources-to-accuracy

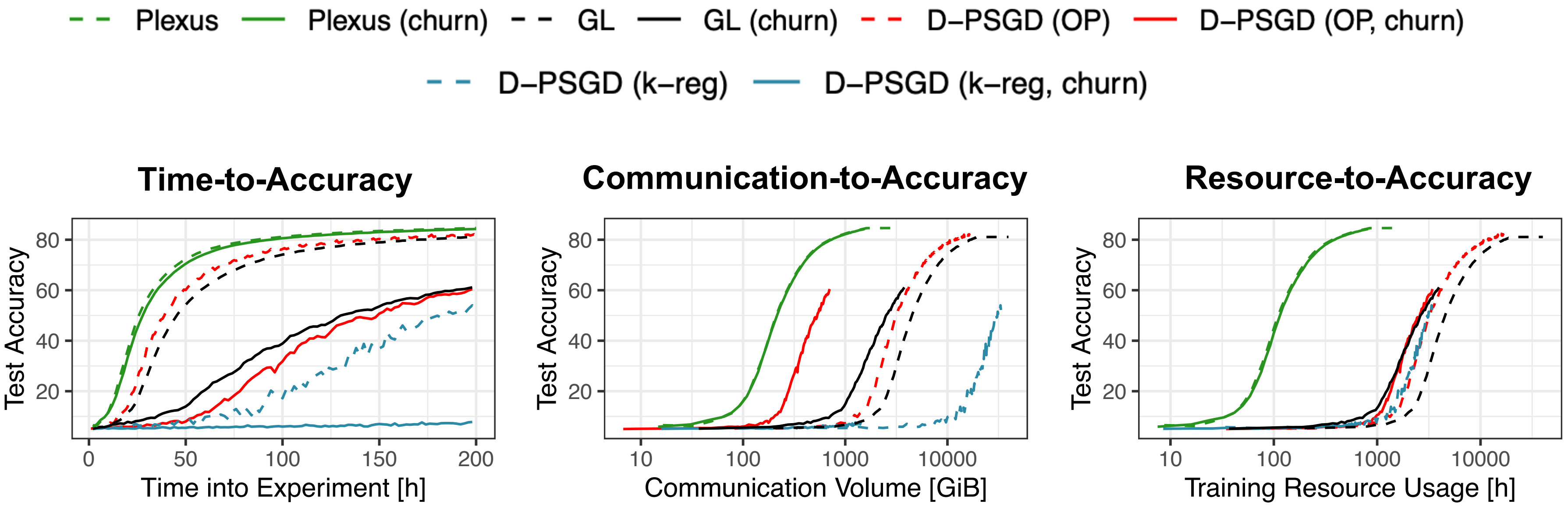
Experiment Setup (2/2)

- Four datasets:

DATASET	TASK	NODES	LEARNING PARAMETERS	MODEL	MODEL SIZE
CIFAR10 [31]	Image classification	1000	$\eta = 0.002$, momentum $= 0.9$	CNN (LeNet [20])	346 KB
CelebA [10]	Image classification	500	$\eta = 0.001$	CNN	124 KB
FEMNIST [10]	Image classification	355	$\eta = 0.004$	CNN	6.7 MB
MovieLens [18]	Recommendation	610	$\eta = 0.2$, embedding dim $= 20$	Matrix Factorization	827 KB

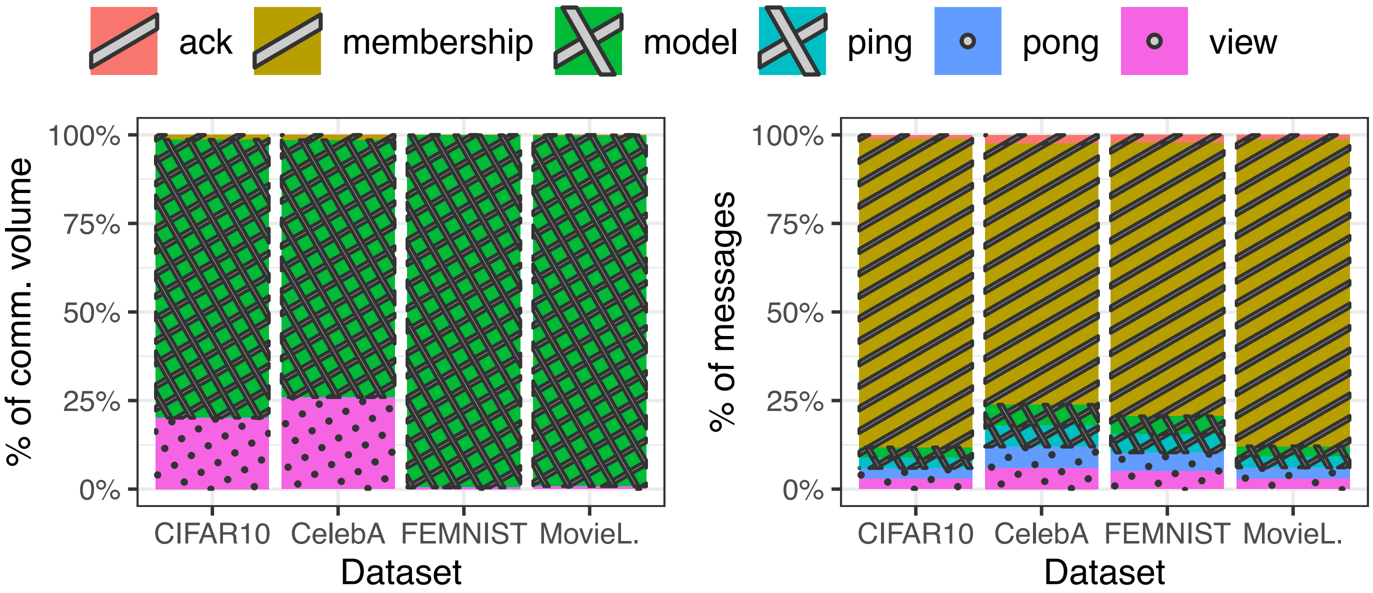
- Three baselines
  - D-PSGD (sparsely connected topology)
  - D-PSGD (k-regular topology)
  - Gossip Learning

Plexus Compared to DL Baselines (FEMNIST)



Plexus shows significant performance improvements over baselines!

Plexus Overhead



Conclusions

- Plexus is a practical and efficient DL system.
- Significant savings in time-to-accuracy (1.2-8.3x), communication-to-accuracy (2.4-15.3x) and resource-to-accuracy (6.4-370x).
- Future work: dealing with Byzantine nodes.

Thank you!

martijn.devos@epfl.ch