



pfl-research: simulation framework for accelerating research in Private Federated Learning

Filip Granqvist, C. Song, Á. Cahill, R. van Dalen, M. Pelikan,
Y. Chan, X. Feng, N. Krishnaswami, V. Jína, M. Chitnis

NeurIPS | Apple | 2024-12-12



github.com/apple/pfl-research

Problems with learning from consumer devices data

Central data curation concerns

- Privacy
- Regulations
- Bandwidth

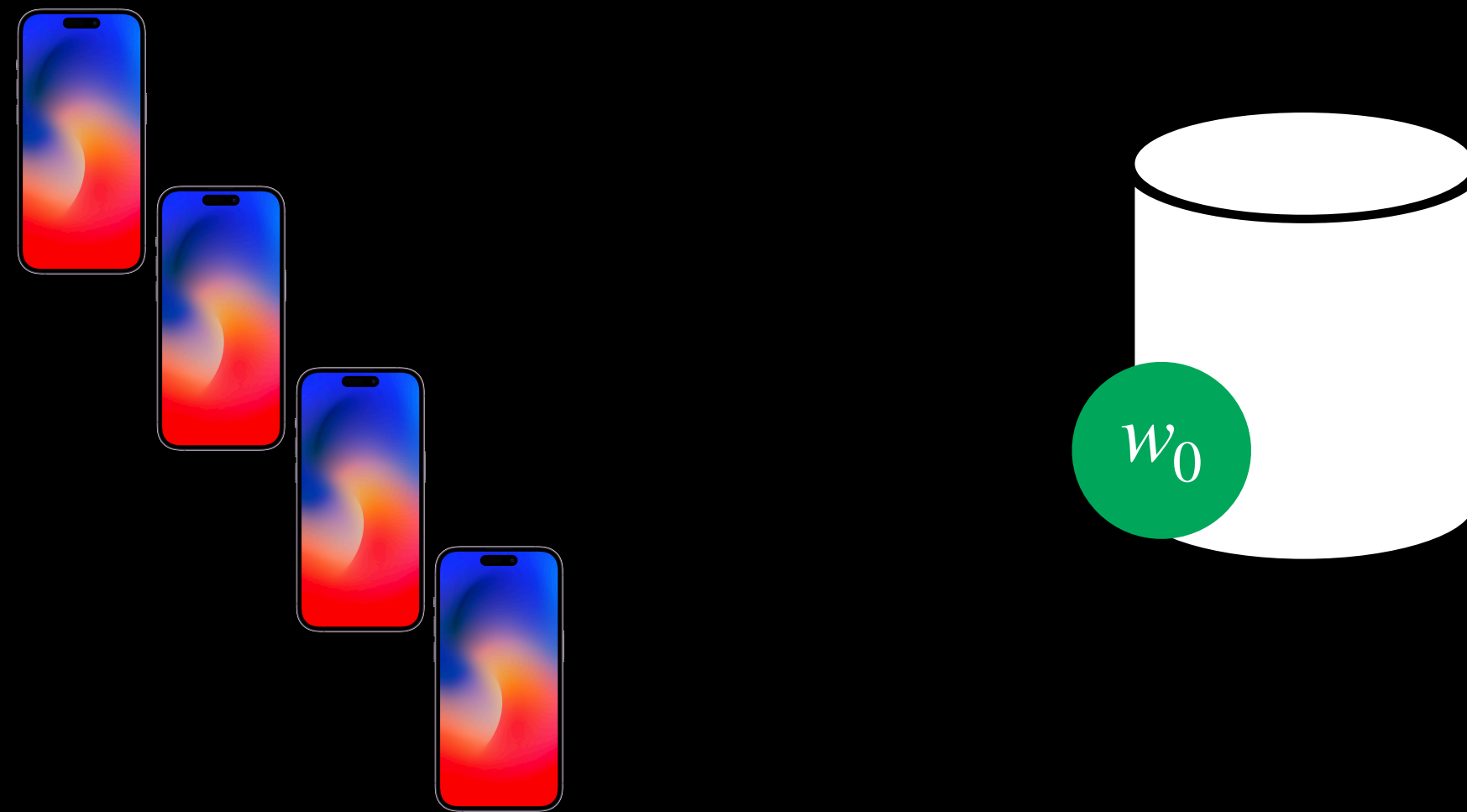
We want to learn from real user generated data on the edge

- Magnitudes more data available because of above
- Same distribution as during inference

Federated Learning

Train ML models without uploading the datasets to a central server

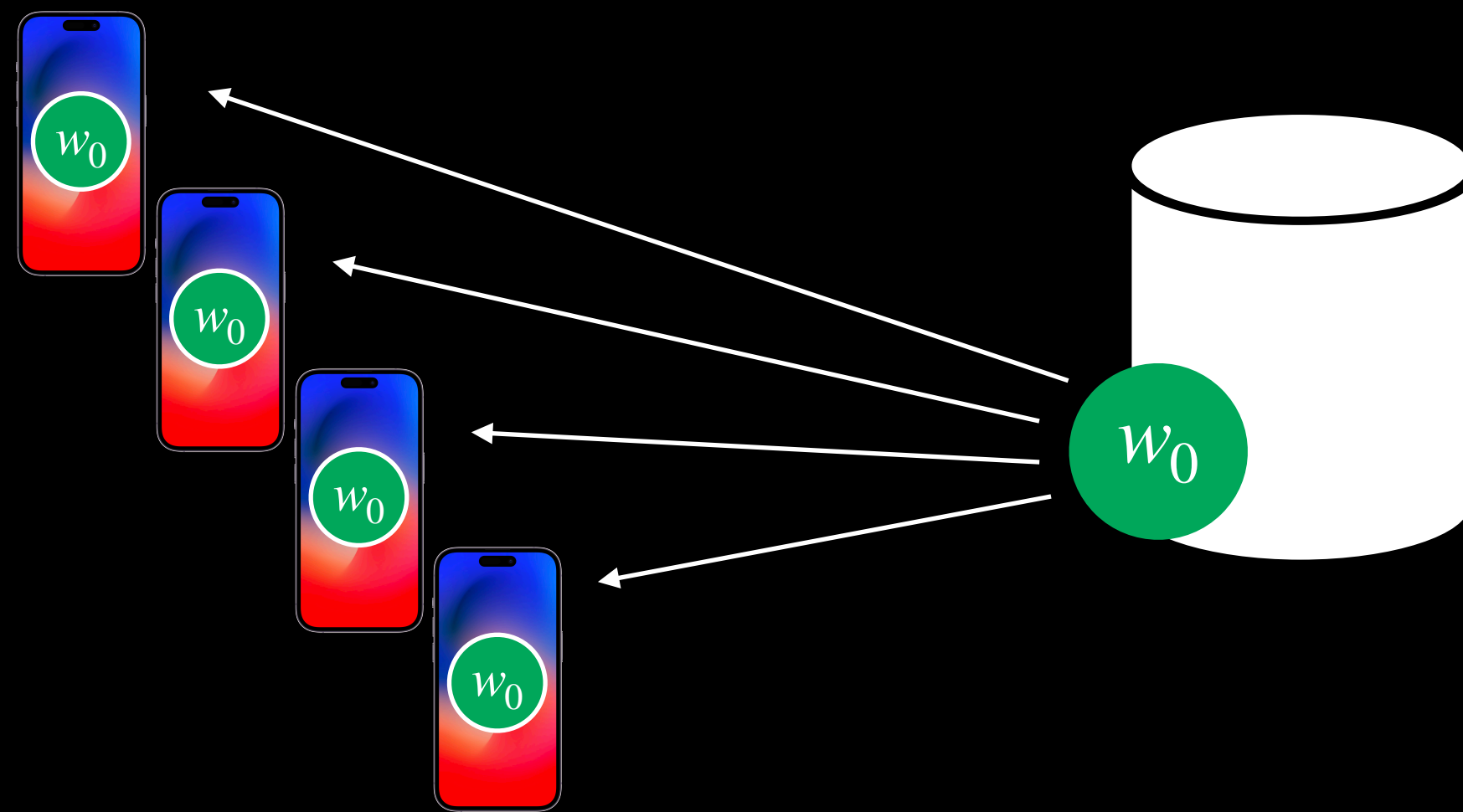
Federated Learning



Initial global model is
held server-side

1. Initialize w_0
2. **for** $t = 0, \dots, T - 1$ **do**
3. $S \leftarrow$ (Random subset of n clients)
4. **for** each client $k \in S$ with dataset D_k in parallel **do**
5. **for** each local iteration i **do**
6. $w_{t+1}^k \leftarrow w_t^k - \eta \mathcal{L}(w_{t+1}^k; D_k)$
7. $\Delta_k = w_{t+1}^k - w_t$
8. $w_{t+1} \leftarrow w_t + \frac{1}{|S|} \sum_{k=1}^{|S|} \Delta_k$

Federated Learning



The global model is distributed to devices

1. Initialize w_0
2. **for** $t = 0, \dots, T - 1$ **do**
3. $S \leftarrow$ (Random subset of n clients)
4. **for** each client $k \in S$ with dataset D_k in parallel **do**
5. **for** each local iteration i **do**
6. $w_{t+1}^k \leftarrow w_t^k - \eta \mathcal{L}(w_{t+1}^k; D_k)$
7. $\Delta_k = w_{t+1}^k - w_t$
8. $w_{t+1} \leftarrow w_t + \frac{1}{|S|} \sum_{k=1}^{|S|} \Delta_k$

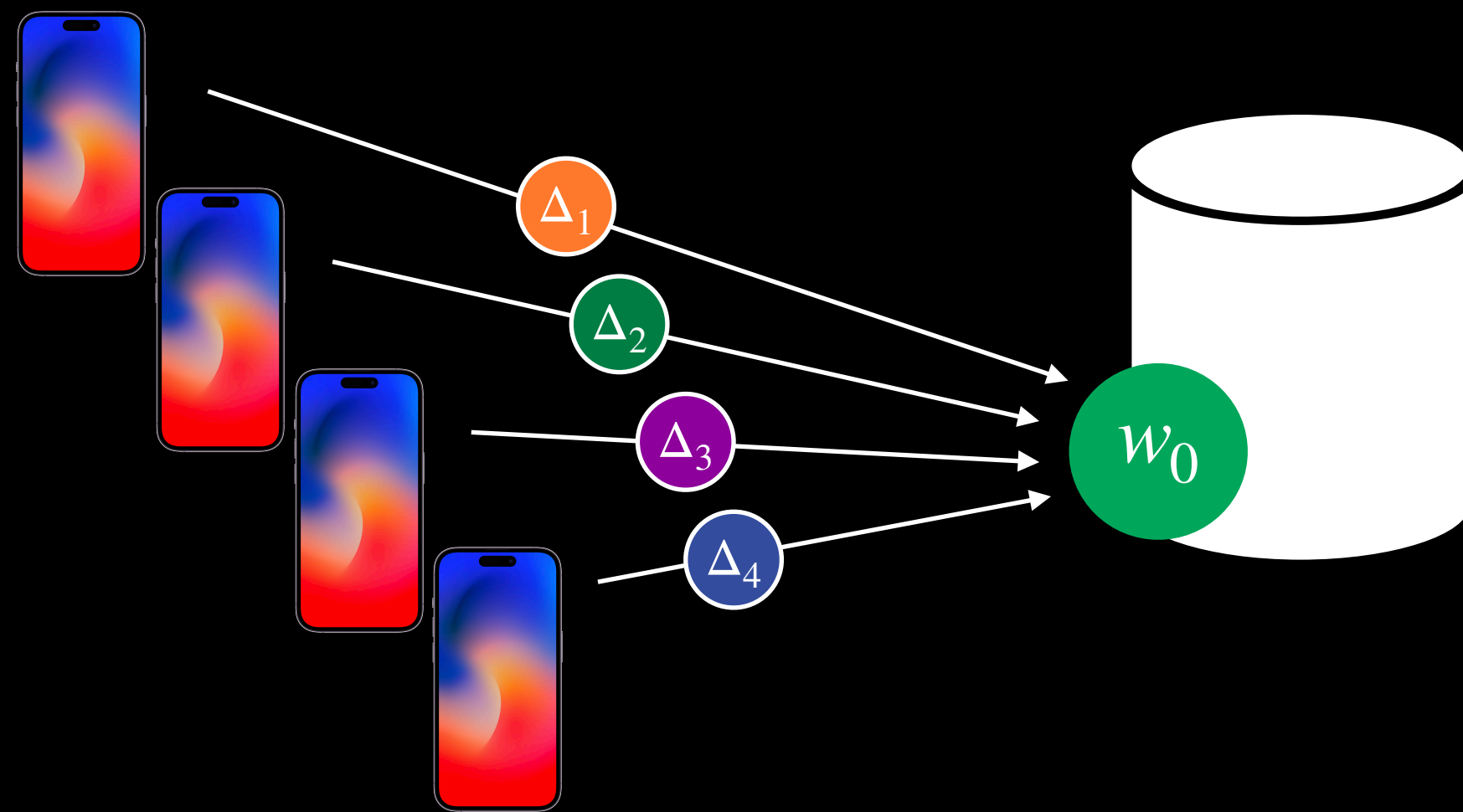
Federated Learning



Devices compute local model updates from user data

1. Initialize w_0
2. **for** $t = 0, \dots, T - 1$ **do**
3. $S \leftarrow$ (Random subset of n clients)
4. **for** each client $k \in S$ with dataset D_k in parallel **do**
5. **for** each local iteration i **do**
6. $w_{t+1}^k \leftarrow w_t^k - \eta \mathcal{L}(w_{t+1}^k; D_k)$
7. $\Delta_k = w_{t+1}^k - w_t$
8. $w_{t+1} \leftarrow w_t + \frac{1}{|S|} \sum_{k=1}^{|S|} \Delta_k$

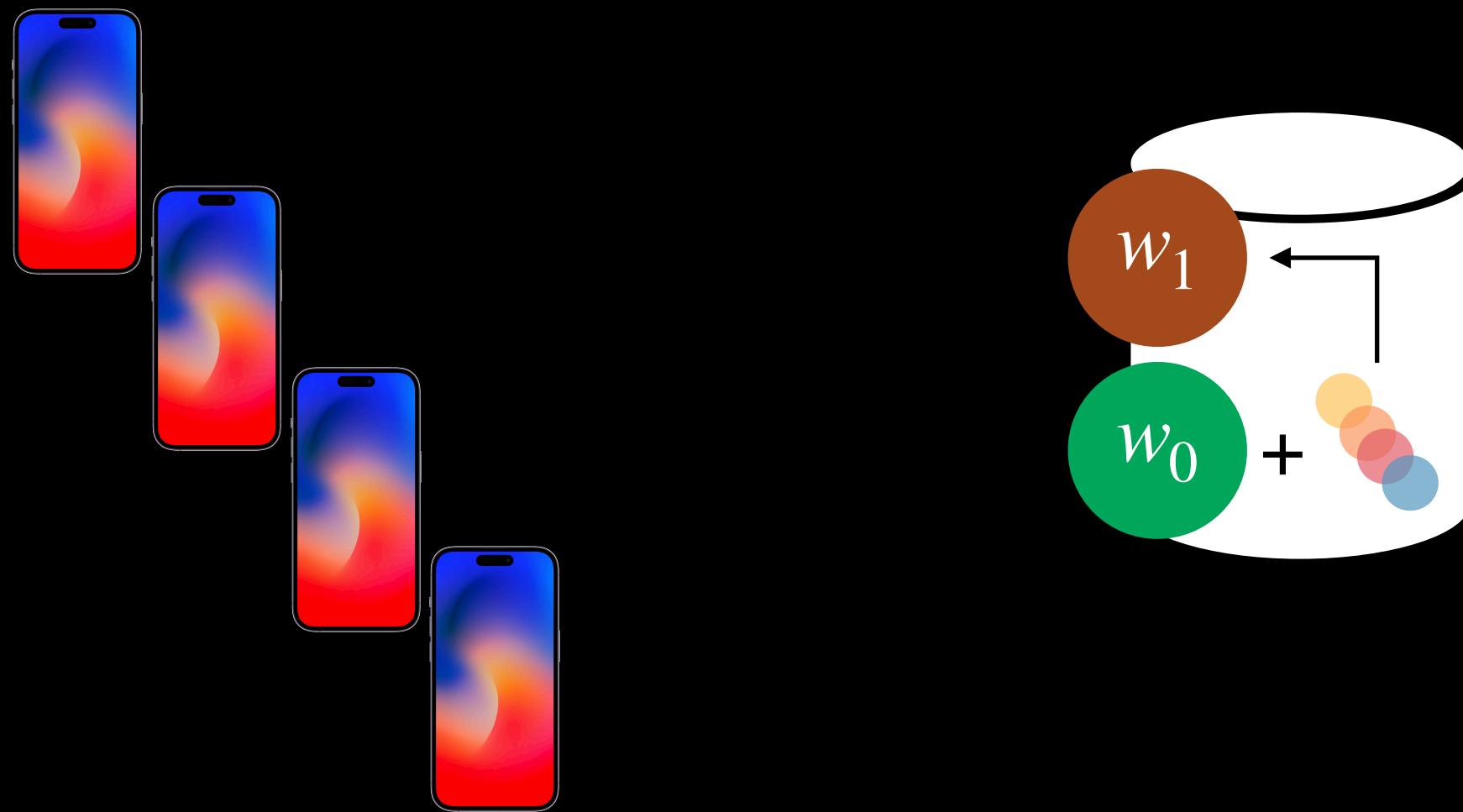
Federated Learning



Devices report model updates

1. Initialize w_0
2. **for** $t = 0, \dots, T - 1$ **do**
3. $S \leftarrow$ (Random subset of n clients)
4. **for** each client $k \in S$ with dataset D_k in parallel **do**
5. **for** each local iteration i **do**
6. $w_{t+1}^k \leftarrow w_t^k - \eta \mathcal{L}(w_{t+1}^k; D_k)$
7. $\Delta_k = w_{t+1}^k - w_t$
8. $w_{t+1} \leftarrow w_t + \frac{1}{|S|} \sum_{k=1}^{|S|} \Delta_k$

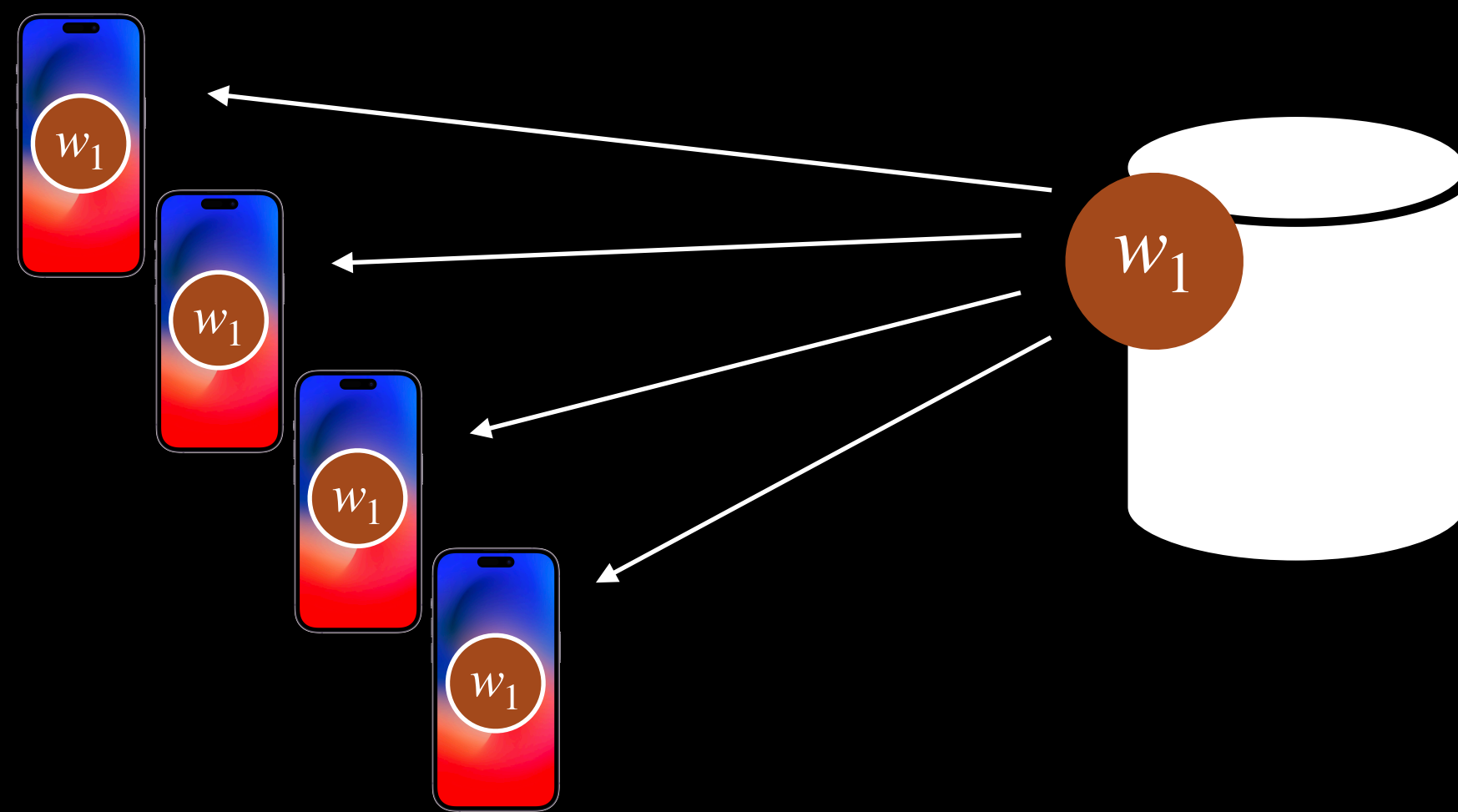
Federated Learning



Central model is updated with aggregated updates

1. Initialize w_0
2. **for** $t = 0, \dots, T - 1$ **do**
3. $S \leftarrow$ (Random subset of n clients)
4. **for** each client $k \in S$ with dataset D_k in parallel **do**
5. **for** each local iteration i **do**
6. $w_{t+1}^k \leftarrow w_t^k - \eta \mathcal{L}(w_{t+1}^k; D_k)$
7. $\Delta_k = w_{t+1}^k - w_t$
8. $w_{t+1} \leftarrow w_t + \frac{1}{|S|} \sum_{k=1}^{|S|} \Delta_k$

Federated Learning



New model is distributed to devices

1. Initialize w_0
2. **for** $t = 0, \dots, T - 1$ **do**
3. $S \leftarrow$ (Random subset of n clients)
4. **for** each client $k \in S$ with dataset D_k in parallel **do**
5. **for** each local iteration i **do**
6. $w_{t+1}^k \leftarrow w_t^k - \eta \mathcal{L}(w_{t+1}^k; D_k)$
7. $\Delta_k = w_{t+1}^k - w_t$
8. $w_{t+1} \leftarrow w_t + \frac{1}{|S|} \sum_{k=1}^{|S|} \Delta_k$

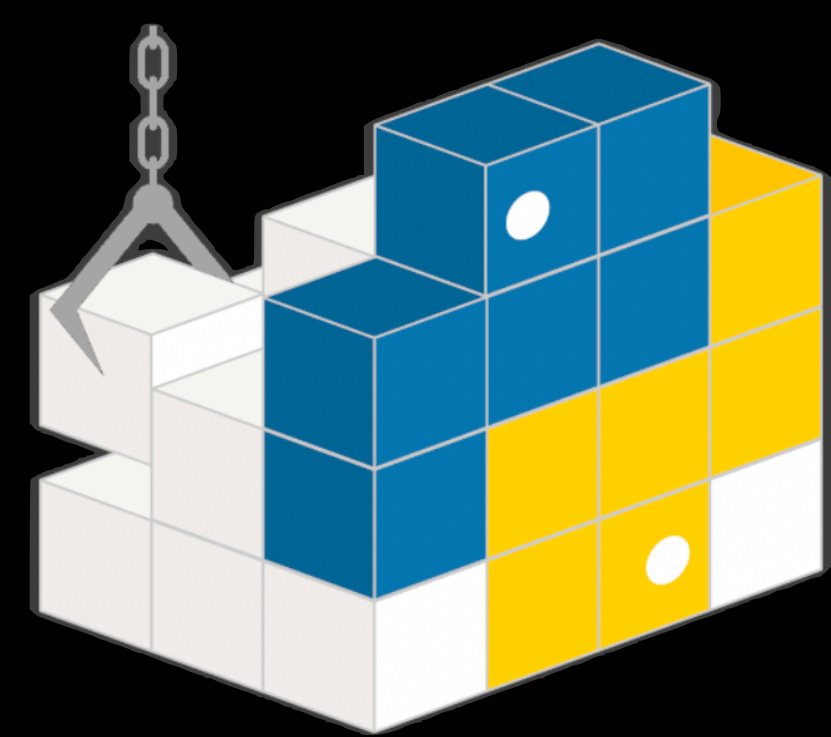
pfl-research

Simulation framework for accelerating research in PFL



github.com/apple/pfl-research

Contributions to open-source



```
pip install pfl
```



Why pfl-research

- Better flexibility, expressiveness and modularity than alternatives.
- **Fastest** scalable simulations for large experiments.
- Supports PyTorch, TF and MLX, we unify benchmarks across frameworks.
- Supports other models in addition to neural networks, e.g. GBDTs, GMMs.
- Support for SoTA open-source privacy accountants & mechanisms
- Simple to understand

pfl-research - example code

```
1 torch_model = torch.nn.Sequential([torch.nn.Linear(4,1)])
2 model = PyTorchModel(torch_model, ...)
3
4 sampler = lambda: user_ids[np.random.randint(0, len(user_ids))]
5 userid_to_data: Dict[int, Tuple[torch.Tensor, torch.Tensor]]
6 train_feddata = FederatedDataset.from_slices(userid_to_data, sampler)
7
8 dp_accountant = PRVPrivacyAccountant(epsilon=2, delta=1e-6, ...)
9 dp_mechanism = GaussianMechanism.from_privacy_accountant(accountant=dp_accountant, clipping_bound=1.0))
10
11 backend = SimulatedBackend(training_data=train_feddata, postprocessors=[dp_mechanism], ...)
12
13 algorithm = FederatedAveraging()
14 algorithm.run(
15     backend=backend,
16     model=model,
17     algorithm_params=NNAgorithmParams(...),
18     model_train_params=NNTrainHyperParams(...),
19     model_eval_params=NNEvalHyperParams(...))
20
```

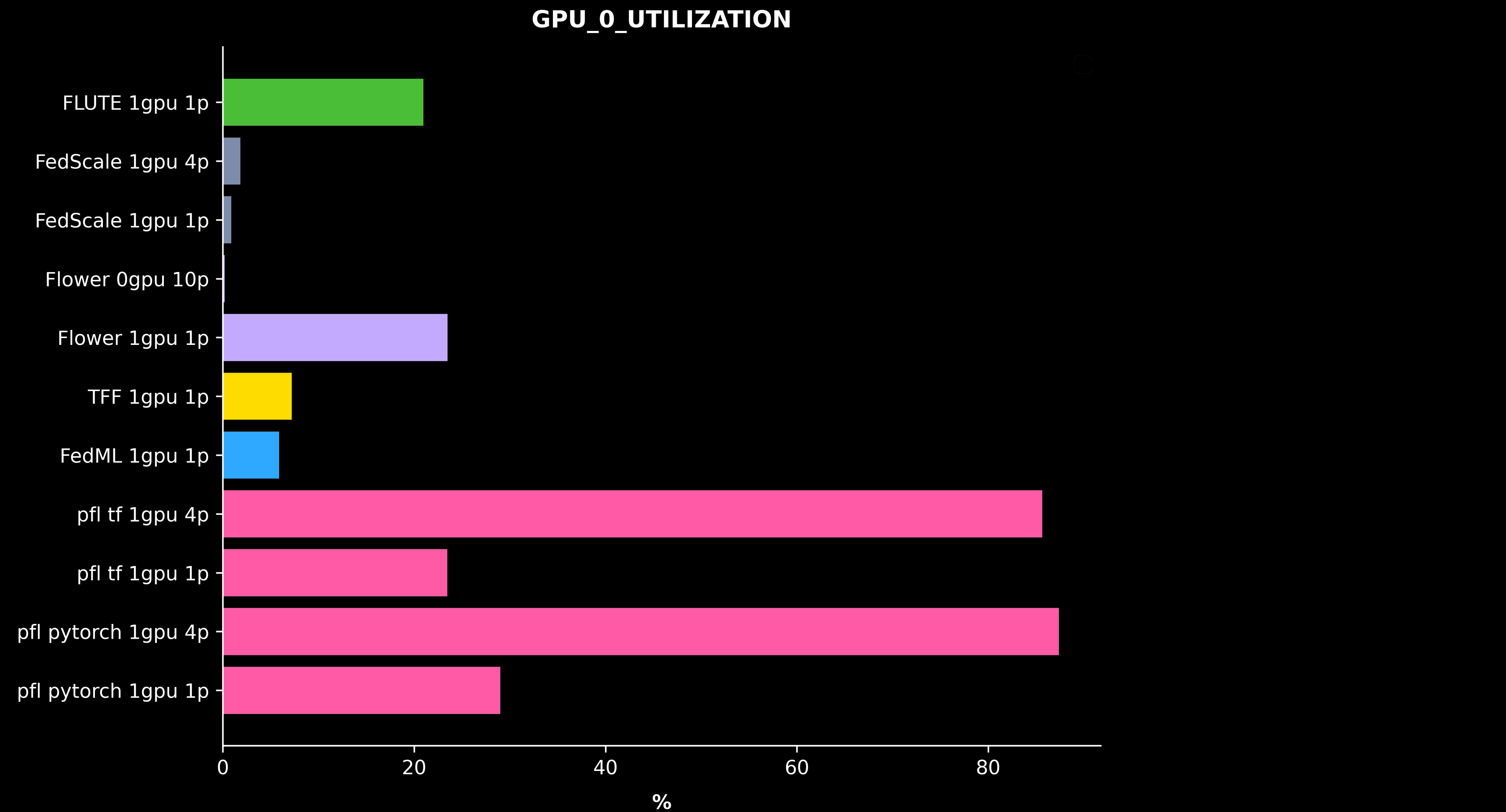
Competitive Analysis

Benchmarking against open-source alternatives, 1GPU

Dataset	FL Framework	#processes	NN Framework	Time to converge (min)	pfl is faster factor
CIFAR10	pfl-research	1	PyTorch	10.13	
		5	PyTorch	4.2	1x
		1	TF2	15.3	
		5	TF2	7.9	
	FedML	1	PyTorch	91	22x
	TFF (Google)	10	TF2	82	20x
	Flower (Oxford/Cambridge)	1	PyTorch	87	16x
		10	PyTorch	29	7x
	FedScale	1	PyTorch	425	101x
		4	PyTorch	302	72x
	FLUTE (Microsoft)	1	TF2	68	16x

Numbers are from the date of release of pfl 0.1.0

Competitive analysis

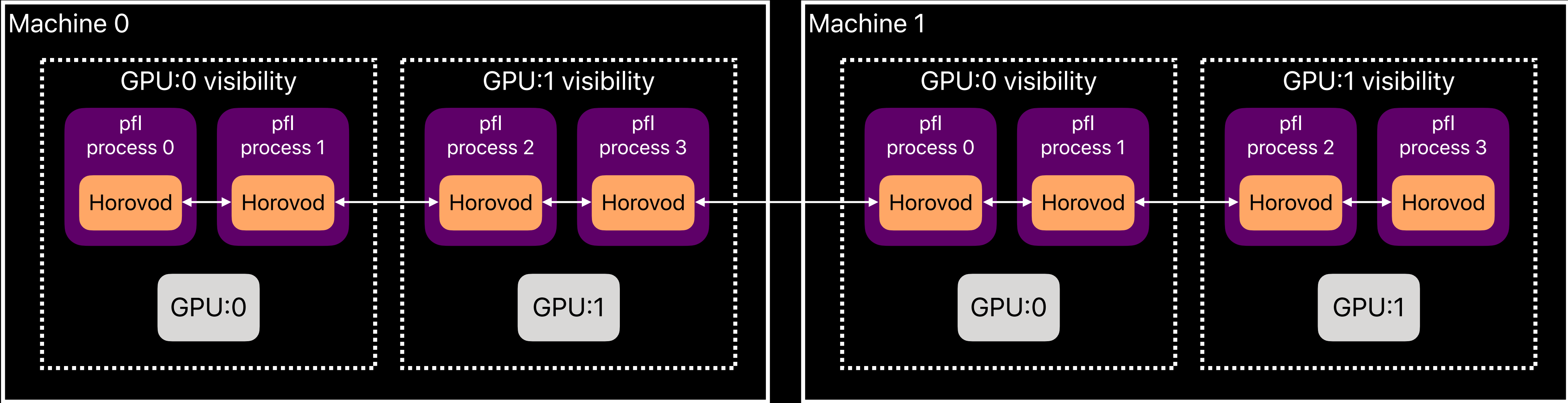


pfl-research - distributed simulations

Multi-process, multi-GPU, multi-machine

Communication overhead

- All-reduce gradients + metrics



Competitive Analysis

Benchmarking against open-source alternatives

Dataset	FL Framework	GPUs	#processes per GPU	NN Framework	Time to converge (h)	pfl is faster factor
FLAIR	pfl-research 0.1.0	4	2	PyTorch	2.16	
	pfl-research 0.2.0	4	2	PyTorch	1.77	1x
	TFF (Google)	4	5	TF2	18	10x
	Flower	4	5	PyTorch	28.8	16x

Official benchmarks



* May still be work in progress at the time of this presentation

Benchmarking pfl-research with MLX on Apple silicon

Dataset	Hardware	#processes	NN Framework	Time to converge (min)	A100 is faster factor
CIFAR10	1x Nvidia A100 GPU	1	PyTorch	10.13	
		5	PyTorch	4.2	
		1	TF2	15.3	
		5	TF2	7.9	
	M1 Pro	1	MLX	21.7	5.2x
		4		15.2	3.6x
	M3 Max	1		10.2	2.4x
		5		5.3	1.3x

Benchmarking pfl-research with MLX on Apple silicon

Dataset	Hardware	#processes	NN Framework	Time to converge	A100 is faster factor
StackOverflow	4x Nvidia A100 GPU	5	PyTorch	52m	
	M1 Pro	5	MLX	8h 12m	9.4x
	M3 Max	5		3h 6m	3.6x

Getting started

Tutorials



github.com/apple/pfl-research/tree/main/tutorials

Documentation



apple.github.io/pfl-research

Benchmarks



github.com/apple/pfl-research/tree/main/benchmarks

