Tackling System Heterogeneity in Federated Learning

FedShuffle: Optimizing Completion Time by Shuffling Clients

Mazhar Ali

Research Seminar Presentation

June 1, 2026

Department of Computer Science & Engineering
Soongsil University

D.I.LAB 숭실대학교 분산지능연구실 Distributed Intelligence Lab

숭실대학교 Soongsil University

Outline

1. Introduction & Motivation
2. FedShuffle Mechanism
3. Experimental Results
4. Outlook

Introduction & Motivation

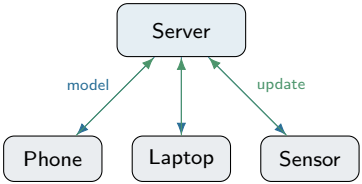
Federated Learning Overview

Train one shared model across many devices — without moving the data.

Each round of cross-device FL:

1. Server sends the model to a set of devices *(download)*
2. Each device trains on its *own local* data *(compute)*
3. Devices send back their updates *(upload)*
4. Server averages them *(FedAvg)*

Clients are phones, laptops, IoT sensors — **thousands of them**, each with very different hardware parameters [1] .



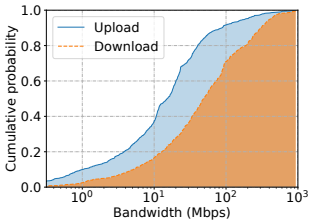
Data **never** leaves the device.

System Heterogeneity Challenges

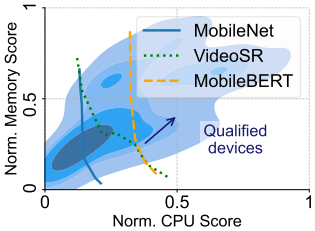
Large-scale FL (e.g., Gboard with ∼1B users), network usage, and compute capability are the major issues [2].

Comm. & Compute Bottleneck

- Client bandwidth varies widely(e.g., ∼5% of devices have $<$4Mbps vs. median ∼81Mbps).
- Edge devices vary widely in hardware capabilities (memory, CPU, availability), as well as software versions, leading to inconsistent response times [3].
- Downstream (server-to-client) and upstream (client-to-server) costs both matter; compute also dominates due to diverse ML tasks and model size.



(a) Bandwidth Dist. [4]



(b) Hardware Heterogeneity [5]

The problem: stragglers in synchronous FL

In synchronous FL, the server must wait for every selected device each round.
A round's cost for a device:

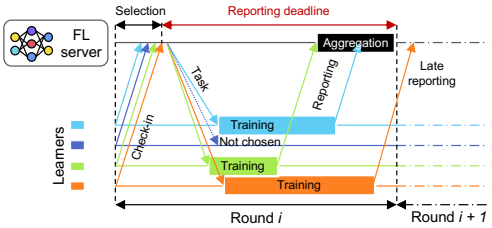
$$T^{\text{dl}} + T^{\text{comp}} + T^{\text{ul}}$$

Round time = the slowest device.

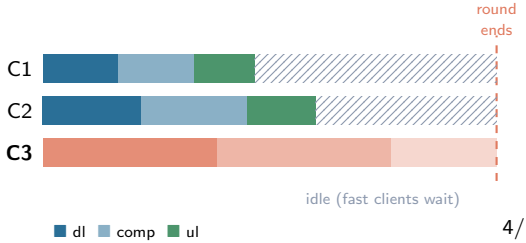
Devices differ along three axes:

- Download speed
- Compute capability
- Upload speed

$\Rightarrow$ Long rounds, idle fast devices, slow convergence.



(c) A round of training in the FL setting



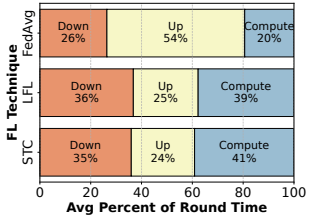
Prior work — and what's missing

- **Compression** (sparsification, quantization, low-rank): shrink payloads [6].

  Mostly tuned for the *downlink*; uplink gradients need different handling.

- **Client selection / overcommitment**: sample extra, drop the slowest [7, 8].

  Biases the model — discards *data-rich but slow* devices.



(d) FL round time breakdown

- **FedFetch**: *prefetch* the model update to reduce download latency [9].

  Powerful, but *downlink-focused*.

The gap

No Prior work handles **download, upload and compute** heterogeneity together.

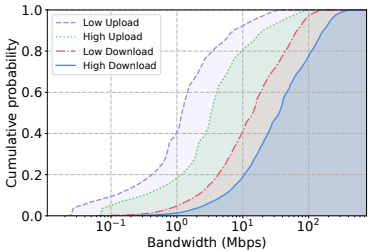
Motivation: prefetching alone is not enough

We tested FedFetch as the bottleneck shifts across dimensions.

Dominant bottleneck	FedFetch speed-up
Download-bound	up to 1.54×
Upload / compute-bound	only ∼1.03×

(Relative total completion time vs. FedAvg)

- Prefetch shines when *download* is the bottleneck.
- Its benefit nearly vanishes when *upload / compute* dominate.



(e) Network Regime	TCT
Low UL + Low DL	**1.03X**
Low UL + High DL	1.13X
High UL + Low DL	1.32X
High UL + High DL	**1.54X**

Takeaway

To tame upload/compute stragglers we also need **scheduling** — not just prefetching.

FedShuffle Mechanism

FedShuffle: two complementary phases

(A) Shuffle — system-aware scheduling

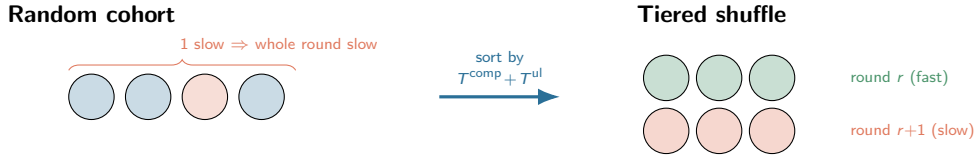
Regroup pre-sampled clients into tiers of similar speed, so no round mixes fast and slow devices.

$\Rightarrow$ kills *compute* / *upload* straggling.

(B) Prefetch — adaptive downloads

Download model updates early, in the background (à la FedFetch), to hide downlink latency.

$\Rightarrow$ kills *download* straggling.



One phase removes upload/compute waiting; the other removes download waiting. **Together: robust across all regimes.**

Experimental Results

Experimental setup

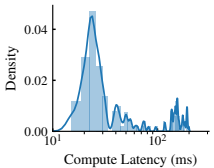
Framework: FedScale (real device traces)
Datasets (non-IID):

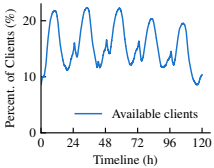
- FEMNIST — handwriting
- CIFAR-10 — images
- Google Speech — keywords

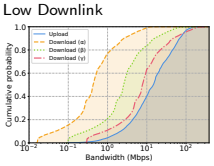
Models: ShuffleNet, ResNet
Config: 1000 rounds, $K = 30 - 100$/round, window $W = 4$, batch size 32, learning rate 0.01.
Compared:

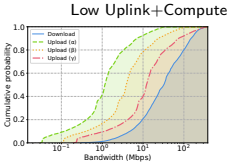
- FedAvg — baseline
- FedFetch — prefetch only
- Shuffle — scheduling only
- **FedShuffle** — both

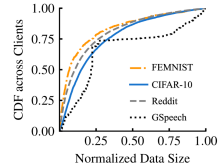
Metric: Time-to-accuracy (virtual clock).

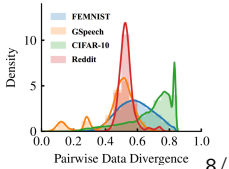




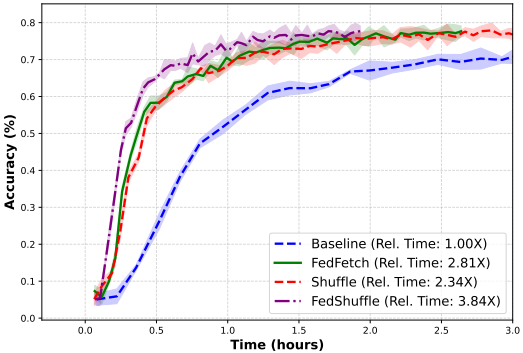


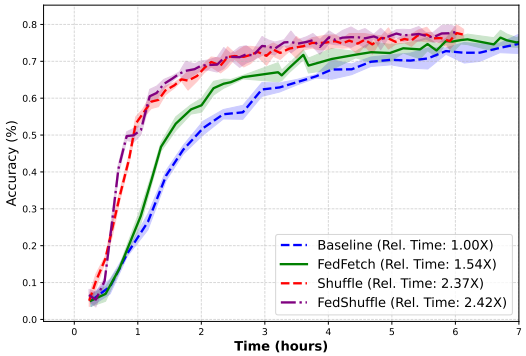






Results: FedShuffle wins in both regimes





(i) Download-bound: up to $3.84\times$ vs FedAvg. Prefetch carries most of the gain.

(ii) Upload/compute-bound: scheduling carries the gain; up to $1.57\times$ over FedFetch.

FedShuffle showed robust to *all* dimensions of heterogeneity.

Robustness across compression, and tasks

We evaluated all the four methods on three different **compression** schemes and datasets.

- FedShuffle fastest in all 9 settings (3 schemes $\times$ 3 datasets).
- Shuffle suppresses straggler-induced idle time.
- Across compression techniques, absolute gains vary, yet the patterns remain the same.

Compression	Method	FEMNIST (75%)	Speech (61%)	CIFAR-10 (65%)
Masking	Baseline	5.60	25.26	13.67
	FedFetch	4.83	12.27	11.52
	Shuffle	2.90	9.06	6.82
	FedShuffle	**2.50**	**8.12**	**5.88**
Quantization	Baseline	2.18	6.38	8.33
	FedFetch	1.81	4.25	7.38
	Shuffle	1.47	3.20	4.47
	FedShuffle	**0.98**	**2.77**	**4.08**
Low-rank	Baseline	18.75	48.69	34.97
	FedFetch	17.55	38.75	26.18
	Shuffle	10.37	30.36	20.21
	FedShuffle	**9.70**	**26.68**	**15.52**

(iii) Total completion time (TCT) to reach target accuracy. Lower is better.

Ablation studies on Shuffle, OC, TV, Prefetch, and Noise

- **Shuffle Window:** Larger $W \rightarrow$ tighter alignment; $W{=}4-6$ balances gain vs scheduling overhead.
- **Overcommitment:** dropping the slowest pushes gains up to $\sim 13\times$ — at the cost of extra prefetch bandwidth.
- **Transmission Bandwidth Volume:** prefetching does repartitions the required downlink bytes between background prefetch and residual blocking downloads.
- **Prefetch Horizon:** Larger prefetch $P > 9$ may induced model staleness problem and required to avoid for faster convergence.
- **Noisy profiles:** stable up to moderate estimation error.

Outlook

Open directions — where I'd like to collaborate

- **Robust / online profile estimation** under drift and partial observability (today the server assumes good estimates).
- **Joint scheduling + data-aware selection**: combine system-aware tiering with utility/importance to also optimize both *data* and *hardware* efficiency.
- **Multi-Job Scheduling**: intergate different tasks to optimally schedule clients with respect to state heterogeneity and hardware diversity.
- **New settings**: asynchronous FL, model pruning schemes, interactions with privacy (DP / secure aggregation).

Joint Work

These are concrete, fundamental problems in cross-device FL — happy to discuss.

FedShuffle

schedule similar clients together,

and fetch early.

Up to $4.15\times$ faster than FedAvg • up to $1.58\times$ over FedFetch

robust across download / upload / compute and compression.

Thank you — let's collaborate

FedShuffle integrates bottleneck-aware tiering with adaptive prefetching to tackle *system* heterogeneity in cross-device FL.

I'm happy to collaborate on the *state* and *hardware* heterogeneity domain of edge devices in FL, as preliminarily discussed in the presentation.

Mazhar Ali
Distributed Intelligence Lab
Soongsil University
mazhar@soongsil.ac.kr

Questions??

References i

Bonawitz, K., et al. "Towards Federated Learning at Scale: System Design." *MLSys*, vol. 1, pp. 374–388, 2019.

Hard, A., et al. "Federated Learning for Mobile Keyboard Prediction." *arXiv:1811.03604*, 2018.

Li, Z., et al. "VENN: Resource Management for Collaborative Learning Jobs." *SoCC*, 2023.

Measurement Lab. "The M-Lab NDT Data Set." 2024.

Lai, F., et al. "FedScale: Benchmarking Model and System Performance of Federated Learning at Scale." *ICML*, 2022.

Wangni, J., et al. "Gradient Sparsification for Communication-Efficient Distributed Optimization." *NeurIPS*, 2018.

Fu, L., et al. "Client Selection in Federated Learning: Principles, Challenges, and Opportunities." *IEEE IoT Journal*, 2023.

References ii

Hard, A., et al. "Learning from Straggler Clients in Federated Learning." *arXiv:2403.09086*, 2024.

Q. Yan, et al. "FedFetch: Faster Federated Learning with Adaptive Downstream Prefetching." *IEEE INFOCOM*, 2025.

Kairouz, P., and McMahan, H. B. "Advances and Open Problems in Federated Learning." *Foundations and Trends in ML*, vol. 14, pp. 1–210, 2021.

Li, T., et al. "Federated Optimization in Heterogeneous Networks." *MLSys*, vol. 2, pp. 429–450, 2020.

Ye, M., et al. "Heterogeneous Federated Learning: State-of-the-Art and Research Challenges." *ACM Computing Surveys*, 2023.

Backup — Phase A: Shuffle (tiered scheduling)

Over a sliding window of W rounds:

1. Pre-sample $K \cdot W$ clients.
2. Score each by its *post-download* cost:

$$s_i \;=\; T_i^{\text{comp}} + T_i^{\text{ul}}, \qquad T_i^{\text{comp}} \approx \gamma B E\, n_i, \quad T_i^{\text{ul}} \approx \frac{S_i^{\text{ul}}}{u_i}.$$

3. Sort, then split into W tiers of K — each tier trains together in one round.

Effect

Within every round, completion times are **aligned** $\Rightarrow$ the slow "tail" shrinks and fast clients stop waiting.

Backup — Phase B: Adaptive prefetch

- Each scheduled client starts downloading up to P rounds early, in the background.
- A per-round budget T_b caps background traffic and **bounds staleness** ($r^\star - p_i^\star \le P$):

$$T_{b_t} = \alpha\, D_t + (1-\alpha)\, T_{b_{t-1}} \qquad \text{(EMA of recent round durations)}$$

- Pick the *latest* start round that still finishes in time $\Rightarrow$ minimal staleness, no blocking fetch.

Key property

Prefetching does **not** increase total bytes — it just *shifts* downloads to earlier, idle rounds.

Putting it together

FedShuffle (per scheduling window w)

1. Pre-sample pool of $K \cdot W$ clients.
2. Score $s_i = T_i^{\text{comp}} + T_i^{\text{ul}}$; sort the pool.
3. Partition into W tiers of size K (one per round).
4. For each round: choose each client's prefetch start $p_i^\star$ within horizon P, using the EMA budget T_b.
5. Clients prefetch in background $\rightarrow$ fetch any residual $\rightarrow$ train locally $\rightarrow$ upload.
6. Aggregate with **FedAvg**; update T_b.

Server-orchestrated, synchronous, FedAvg-compatible — a drop-in scheduler.

Backup — Prefetch timing (fetch-time estimator)

For a candidate start round p, simulate background downloads round-by-round under budget T_b:

$$\text{(slack) } \tfrac{S_p^{(j)}}{d_i} \le T_b: \; S_p^{(j+1)} = \max\Big(0,\; \Delta(\ell_j, j-1) - \big(T_b - \tfrac{S_p^{(j)}}{d_i}\big)\, d_i\Big)$$

$$\text{(no slack) otherwise: } S_p^{(j+1)} = \max\big(0,\; S_p^{(j)} - T_b\, d_i\big)$$

Blocking fetch cost at the training round $r^\star$:

$$\widehat{F}_b(i,p) \;=\; \frac{S_p^{(r^\star)} + \Delta(\ell_{r^\star-1},\, r^\star-1)}{d_i}.$$

Scheduler picks the *latest* viable start: $p_i^\star = \arg\max\{p : \widehat{F}_b(i,p) \le \tau\}$, with τ the cohort's q-th percentile $\Rightarrow$ staleness $\le P$.

Backup — Overcommitment (FEMNIST, target 75%)

Impact of overcommitment (OC) factor in terms of TCT, total transmission volume (TV) in 100^2 GB, download (fetch) volume (DV), and prefetch volume (PV) (FEMNIST with Trg 75%).

	Shuffle			**FedFetch**				**FedShuffle**			
OC	**TCT**	**TV**	**DV**	**TCT**	**TV**	**DV**	**PV**	**TCT**	**TV**	**DV**	**PV**
1.0	2.75	2.23	1.80	4.69	2.24	1.30	0.49	**2.52**	**2.22**	1.26	0.51
1.1	1.23	**2.37**	1.91	1.97	3.19	2.13	0.63	**1.12**	3.19	2.11	0.66
1.2	0.41	**2.44**	1.97	0.57	3.25	2.02	0.90	**0.34**	3.36	1.99	0.91
1.3	0.28	**2.38**	1.94	0.32	3.58	1.76	1.39	**0.18**	3.62	1.74	1.42
1.4	0.16	**2.39**	1.96	0.21	3.71	1.64	1.61	**0.09**	3.72	1.66	1.60

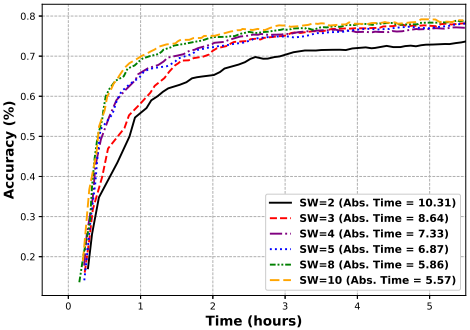
TCT in hours; TV/PV are transmission / prefetch volume. More OC $\rightarrow$ less latency, more prefetch traffic (up to $\sim 3.1\times$).

Backup — Shuffle-window & noise ablations

Shuffle window W (FEMNIST):

- W=2: 10.31 h
- W=4: 7.33 h (our default)
- W=10: 5.57 h

Larger $W \rightarrow$ tighter alignment; W=4 balances gain vs scheduling overhead.

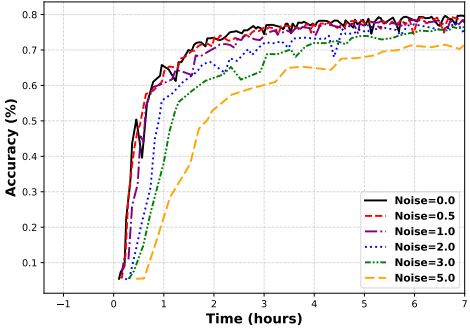


Estimation noise σ:

Multiplicative noise $\hat{x} = x \cdot \max(1+\varepsilon, 0.05)$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

- $\sigma \leq 1.0$: curves track the noiseless baseline.
- $\sigma = 3$–5: noticeable degradation.

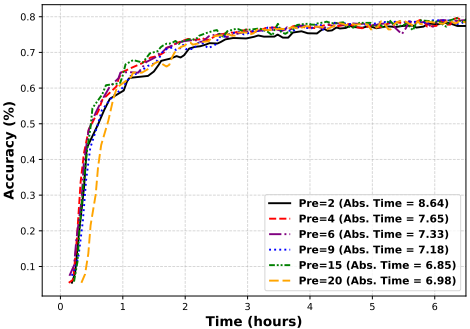
$\Rightarrow$ robust to realistic profiling error.



Backup — Prefetch-horizon & Bandwidth Usage ablations

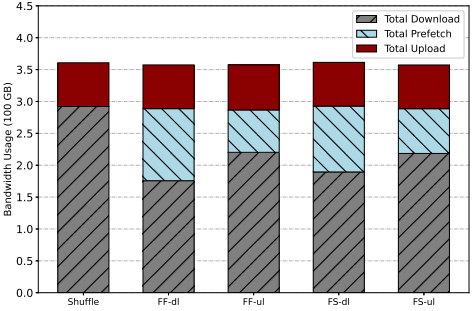
Prefetch window W (FEMNIST):

Larger $W \rightarrow$ tighter alignment; W=4 balances gain vs scheduling overhead.



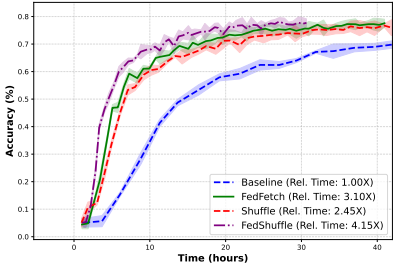
Bandwidth Volume:

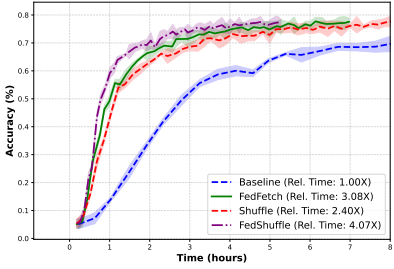
Larger $W \rightarrow$ tighter alignment; W=4 balances gain vs scheduling overhead.



Backup — Remaining DL & UP Scenarios

Download:





Upload+Compute: