MindFlayer: Efficient Asynchronous Parallel SGD in the Presence of Heterogeneous and Random Worker Compute Times

Arto (Artavazd) Maranjyan

Apple MLR Weekly Seminar
21 November 2024

Artavazd Maranjyan, Omar Shaikh Omar, Peter Richtárik
MindFlayer: Efficient Asynchronous Parallel SGD in the Presence of Heterogeneous and Random Worker Compute Times
arXiv:2410.04285, 2024

My Background

2023 – 2025 (expected)
Ph.D. in Computer Science (Federated Learning)
King Abdullah University of Science and Technology (KAUST)
Advisor: Peter Richtárik

2021 – 2023
M.Sc. in Applied Statistics and Data Science
Yerevan State University

2017 – 2021
B.Sc. in Informatics and Applied Mathematics
Yerevan State University

Collaborators

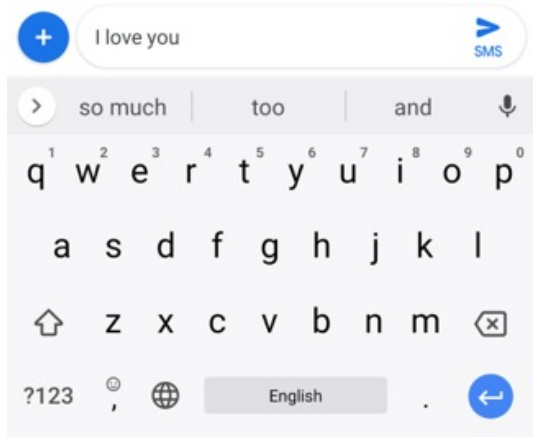
Outline of the Talk

1. What is Federated Learning?
2. Rennala: Optimal Parallel Optimization method
3. MindFlayer


What is Federated Learning?

The First Federated Learning App: Next-Word Prediction

Federated Learning is collaborative machine learning from private data stored across a (large) number of clients/devices (e.g., hospitals, phones)




Parallel optimization methods

Optimization Formulation

$$\min_{x \in \mathbb{R}^d} \{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} [f(x; \xi)] \}$$

model parameters / features

Loss of a data sample ξ

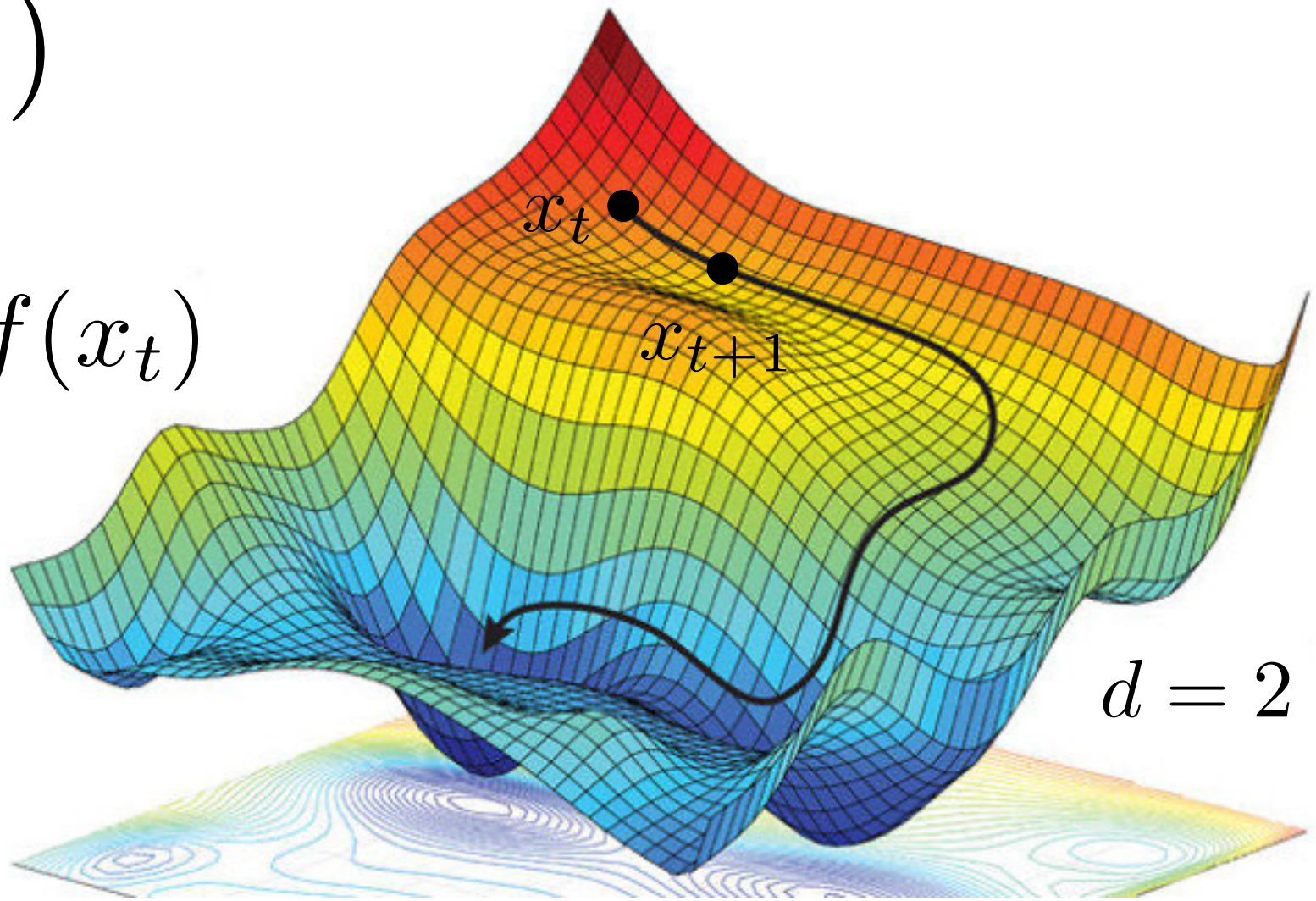
The distribution of the training dataset

Stochastic Gradient Descent

$$\min_{x \in \mathbb{R}^d} f(x)$$

$$x_{t+1} = x_t - \gamma \widehat{\nabla} f(x_t)$$

Stepsize / Learning rate



Assumptions

1. f is differentiable and L–smooth, i.e.,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^d$$

2. There exist $f^* \in \mathbb{R}$ such that $f(x) \geq f^*$ for all $x \in \mathbb{R}^d$

3. For all $x \in \mathbb{R}^d$, stochastic gradients $\widehat{\nabla} f(x;\xi)$ are unbiased and σ^2-variance-bounded, i.e., $\mathbb{E}_\xi[\widehat{\nabla} f(x;\xi)] = \nabla f(x)$ and $\mathbb{E}_\xi[\|\widehat{\nabla} f(x;\xi) - \nabla f(x)\|^2] \leq \sigma^2$, where $\sigma^2 \geq 0$

Parallel Computing Architecture

Worker 1

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_2$ seconds

τ_2

Worker 2

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_3$ seconds

τ_3

Worker 3

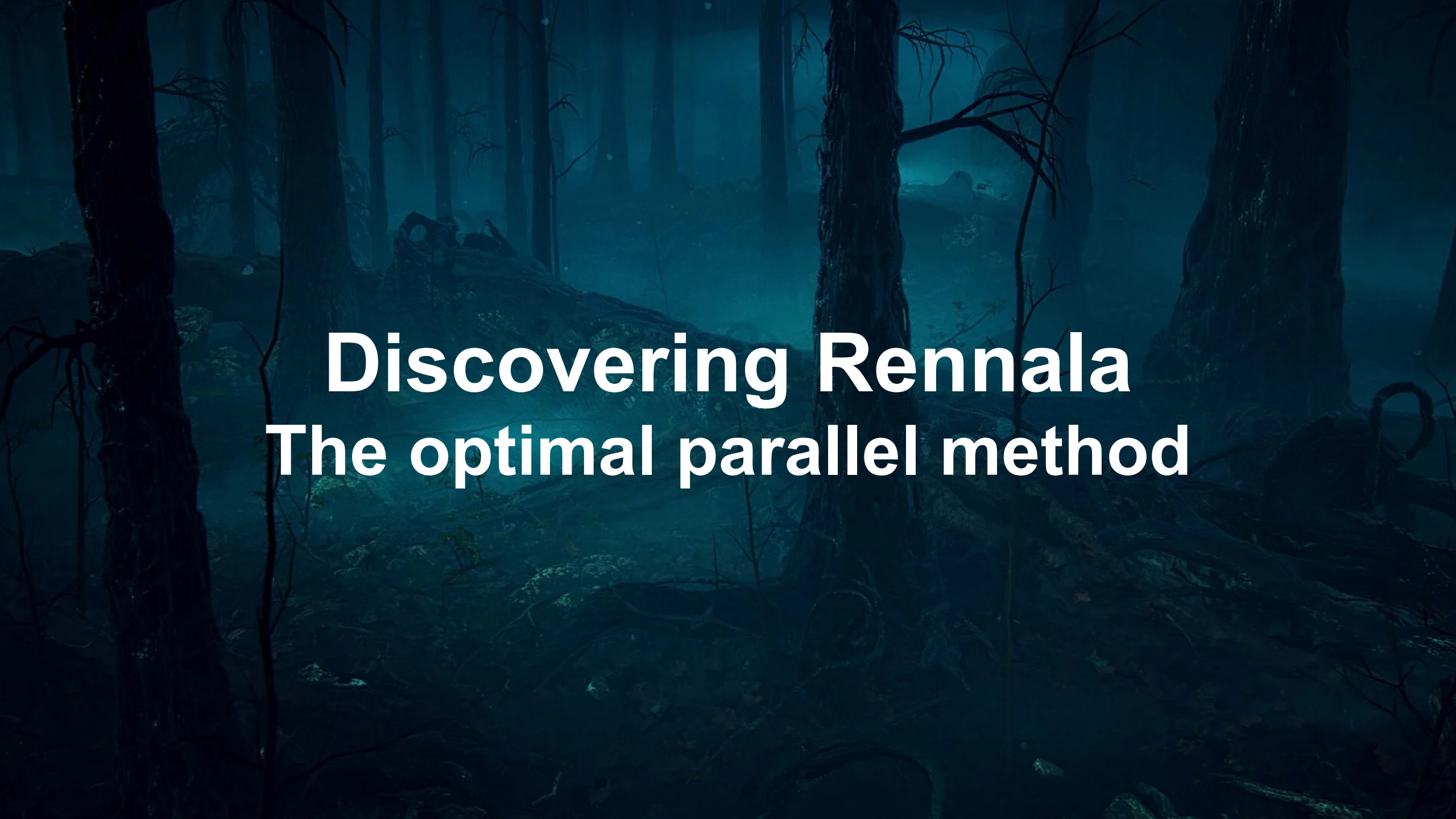
$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_1$ seconds

τ_1

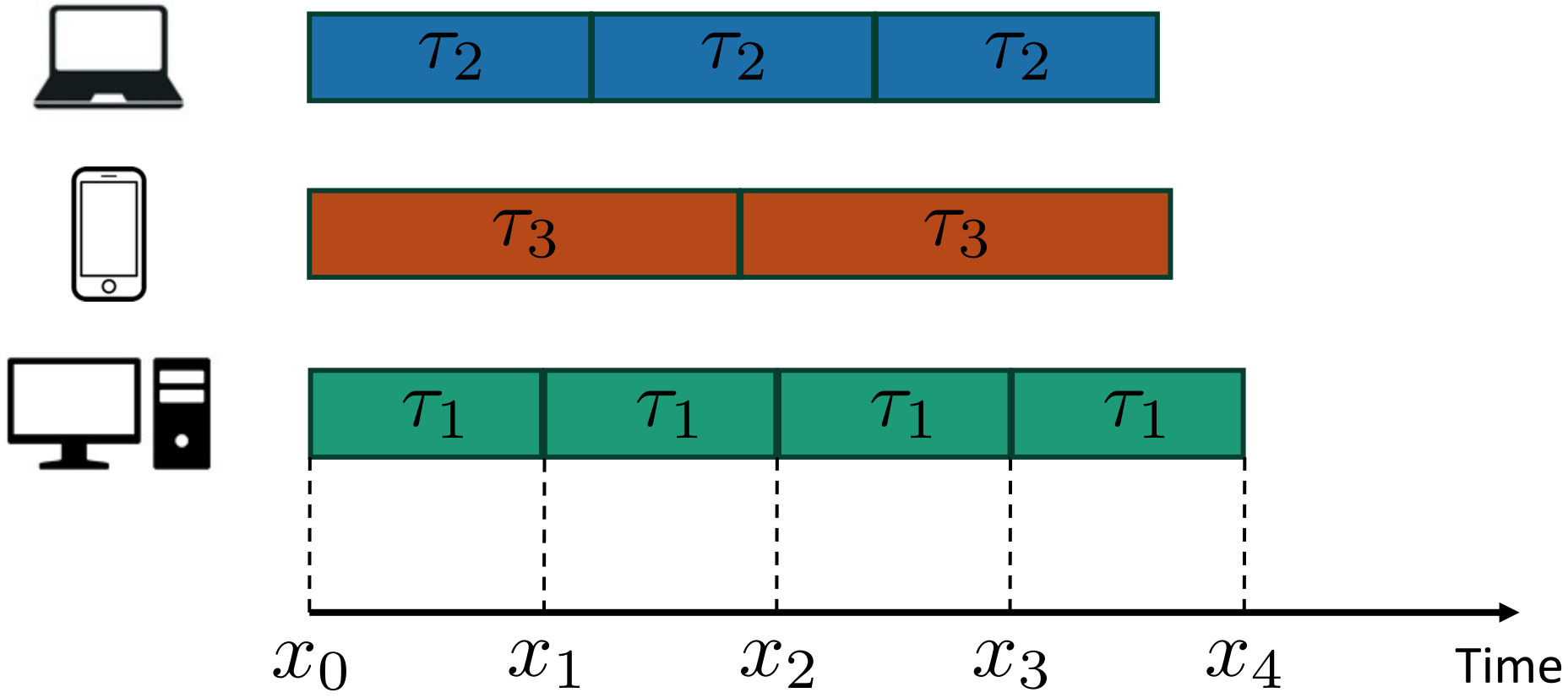
Server

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$$


Discovering Rennala
The optimal parallel method

Hero SGD

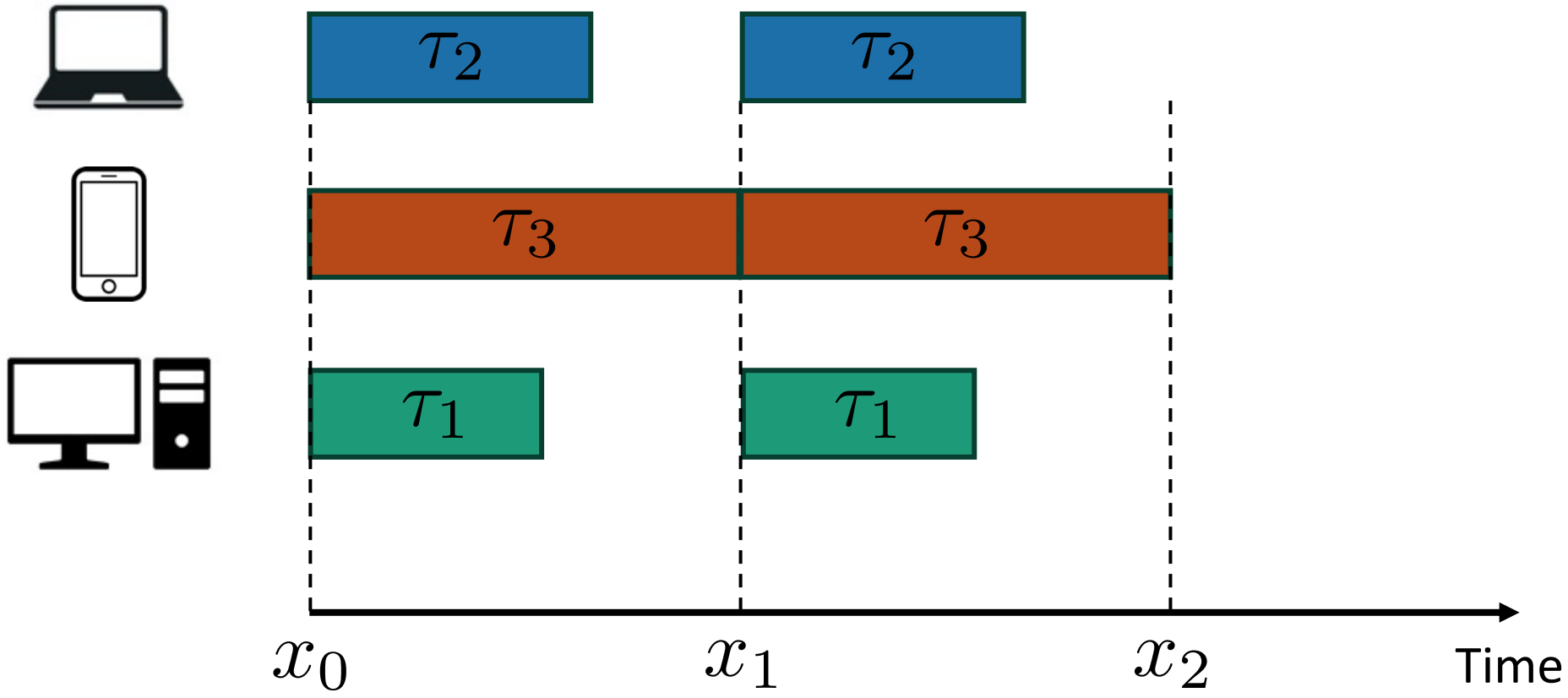
The fastest worker does it all



$$x^{k+1} = x^k - \gamma \widehat{\nabla} f(x^k; \xi^k)$$

“Fair” Minibatch SGD

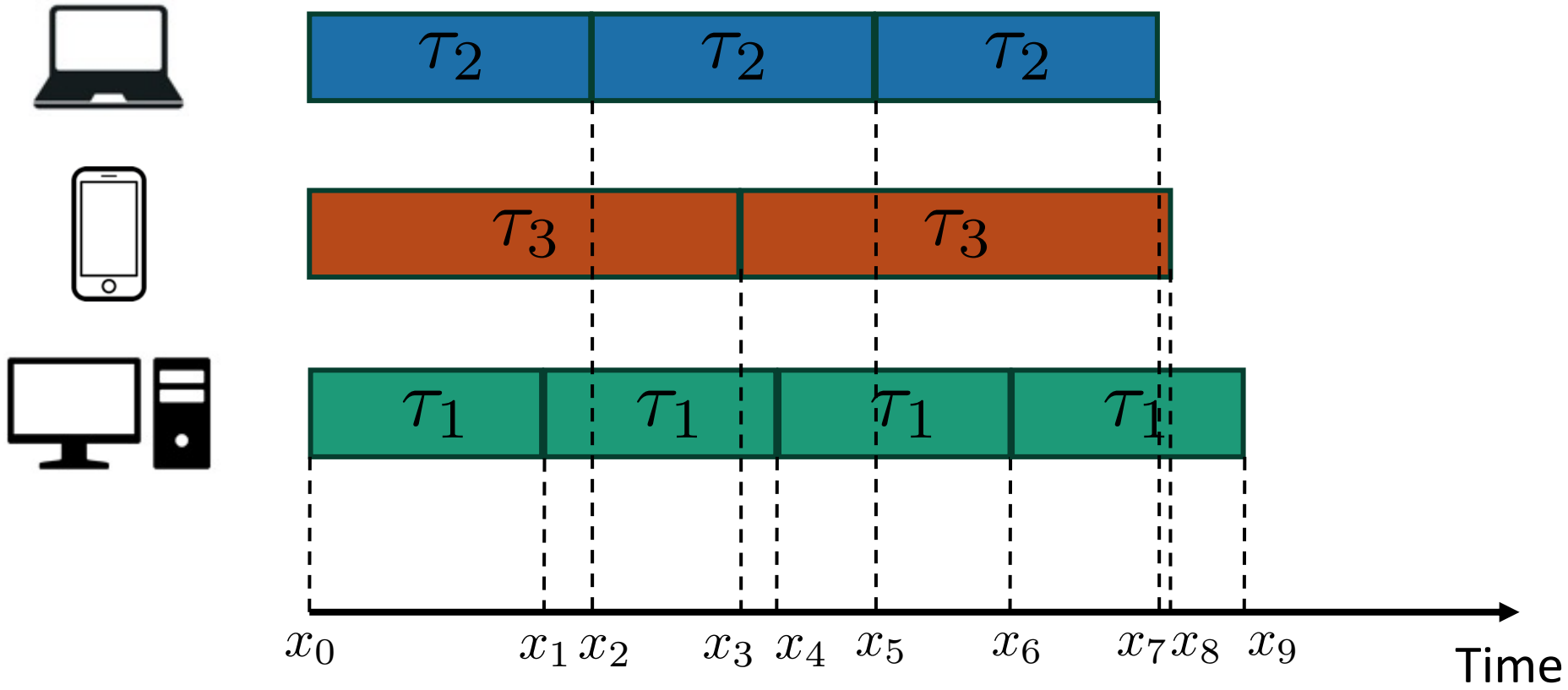
Each worker does one job only



$$x^{k+1} = x^k - \gamma \frac{1}{n} \sum_{i=1}^{n} \widehat{\nabla} f(x^k; \xi_i^k)$$

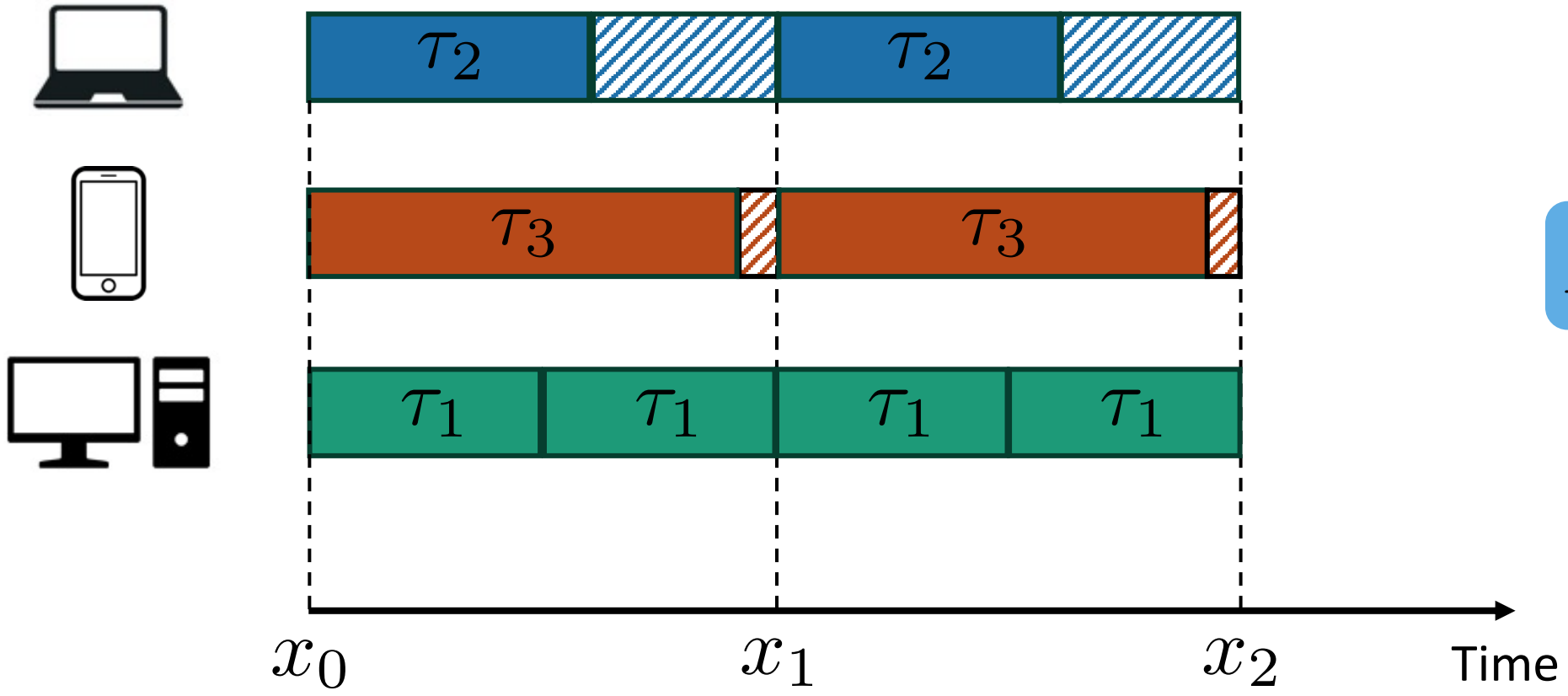
Asynchronous SGD

All workers are slaves and useful



$$\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \nabla f\left(\mathbf{x}_{\mathrm{prev}(k,i_k)}; \xi^{i_k}_{\mathrm{prev}(k,i_k)}\right)$$

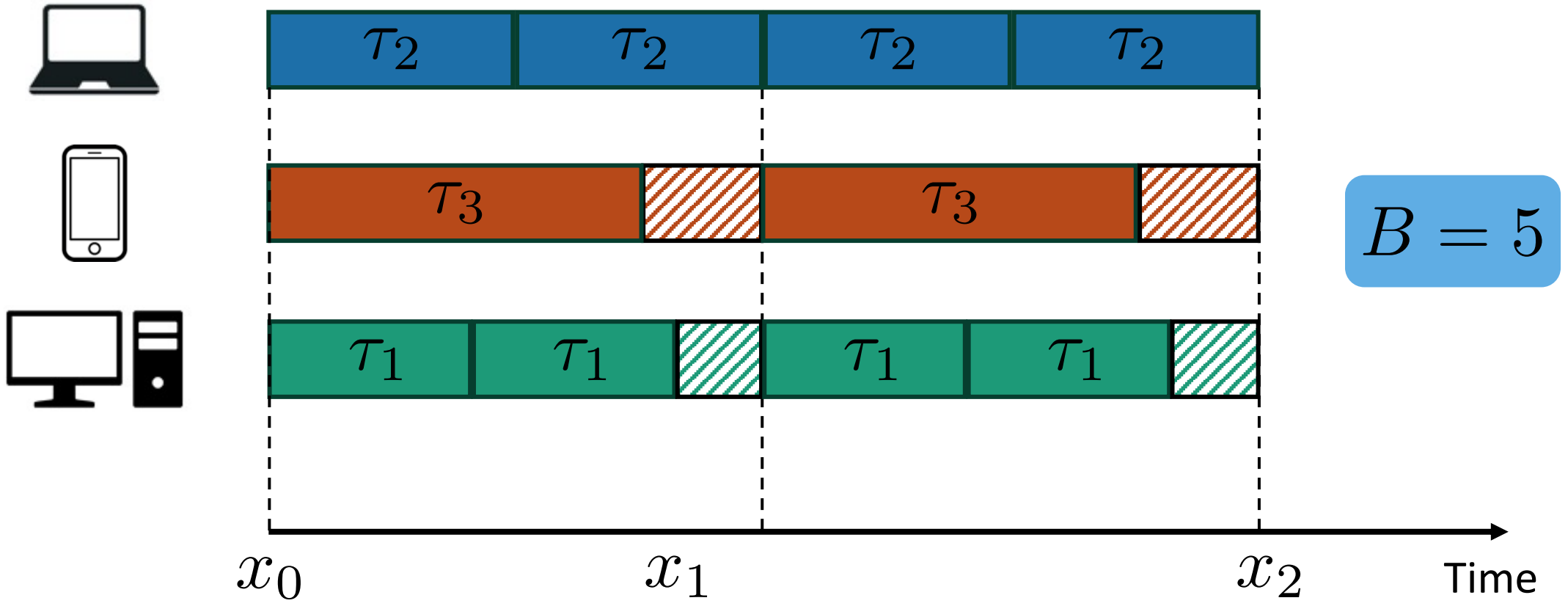
Rennala SGD
Optimal Method



$B = 4$

$$x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{j=1}^{B} \widehat{\nabla} f\left(x^k; \xi_i^j\right)$$

Rennala SGD
Optimal Method



$$x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{j=1}^{B} \widehat{\nabla} f\left(x^k; \xi_i^j\right)$$

Rennala SGD – optimal strategy

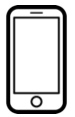
Worker 1

Compute $\widehat{\nabla} f(x;\xi)$ in τ_2 seconds

Compute $\widehat{\nabla} f(x;\xi)$ in τ_2 seconds

Compute $\widehat{\nabla} f(x;\xi)$ in τ_2 seconds

Worker 2



Compute $\widehat{\nabla} f(x;\xi)$ in τ_3 seconds

Worker 3

Compute $\widehat{\nabla} f(x;\xi)$ in τ_1 seconds

Compute $\widehat{\nabla} f(x;\xi)$ in τ_1 seconds

Compute $\widehat{\nabla} f(x;\xi)$ in τ_1 seconds

Compute $\widehat{\nabla} f(x;\xi)$ in τ_1 seconds

Compute $\widehat{\nabla} f(x;\xi)$ in τ_1 seconds

Server

Wait for B stochastic gradients

$$x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{j=1}^{B} \widehat{\nabla} f\left(x^k; \xi_i^j\right)$$

$$B = \max\left\{\left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil, 1\right\}$$

Tyurin, Alexander, and Peter Richtárik. "Optimal time complexities of parallel stochastic optimization methods under a fixed computation model." *Advances in Neural Information Processing Systems* 36 (2024).


MindFlayer

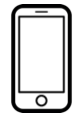
Random Times

Worker 1

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_1 + \eta_1$ seconds

Worker 2



$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_2 + \eta_2$ seconds

$$\eta_i \sim \mathcal{J}_i$$

Worker 3

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_3 + \eta_3$ seconds

Server

- Failing hardware
- Busy with other jobs
- Partial participation
- Inconsistencies in network communications

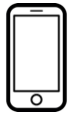
Failure of Rennala

Worker 1

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_1 + \eta_1$ seconds

Worker 2



$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_2 + \eta_2$ seconds

Worker 3

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_3 + \eta_3$ seconds

$$\eta_i = \eta = \begin{cases} 0, & \text{with probability } 1-q, \\ \infty, & \text{with probability } q. \end{cases}$$

Server

Wait for B stochastic gradients

MindFlayer in Upside Down

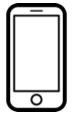
Worker 1

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_1 + \eta_1$ seconds

$t_1 = \tau_1$

Worker 2



$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_2 + \eta_2$ seconds

$t_2 = \tau_2$

Worker 3

$\widehat{\nabla} f(x;\xi)$

Compute time $= \tau_3 + \eta_3$ seconds

$t_3 = \tau_3$

$$\eta_i = \eta = \begin{cases} 0, & \text{with probability } 1-q, \\ \infty, & \text{with probability } q. \end{cases}$$

MindFlayer (Server)

Logistic Regression in Upside Down

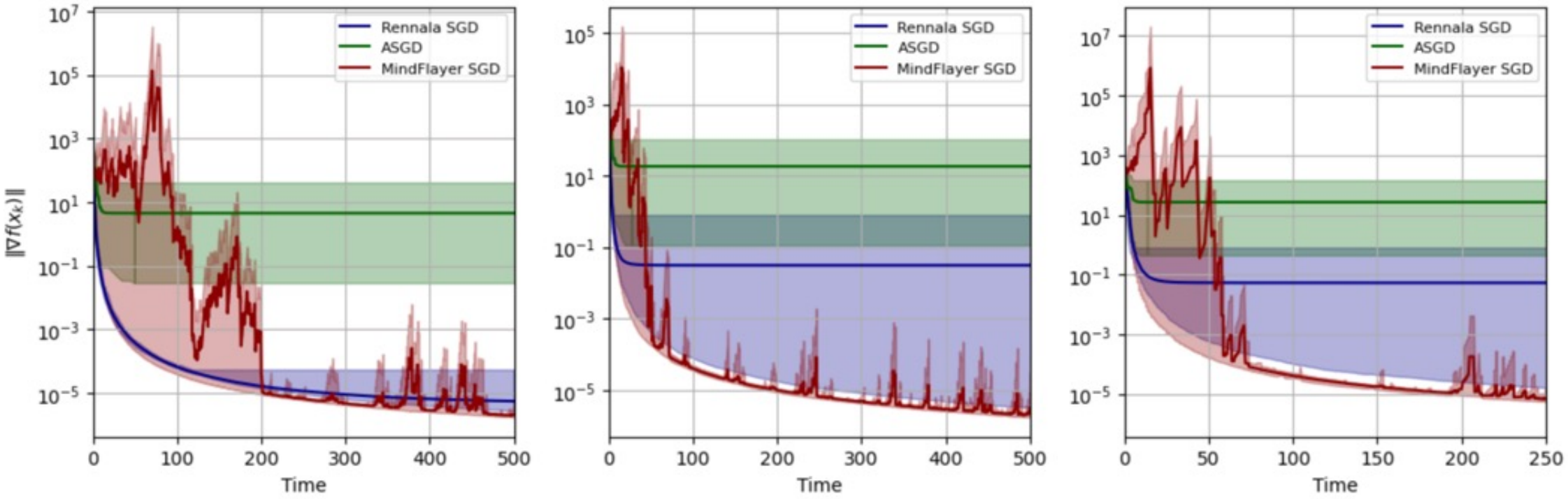


Figure 4: We ran an experiment as described in Section 6 where we employ the same $\mathcal{J}_i =$ InfBernoulli(q) distribution for all clients $i \in [n]$, with different q values. From left to right we have $q = 0.6, 0.7, 0.8$. Additionally, we set $\tau_i = \sqrt{i+1}$. As we observe, with an increase of the probability of failure q unlike Rennala SGD and ASGD, MindFlayer SGD demonstrates the ability to continue optimizing and not be stuck

MindFlayer SGD

Worker 1

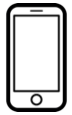
Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_1 seconds

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_1 seconds

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_1 seconds

B_1 trials

Worker 2



Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_2 seconds

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_2 seconds

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_2 seconds

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_2 seconds

B_2 trials

Worker 3

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_3 seconds

Try computing $\widehat{\nabla} f(x^k;\xi)$ within t_3 seconds

B_3 trials

MindFlayer (Server)

$$x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{i=1}^{n} \sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right) \nabla f\left(x^k; \xi_i^j\right)$$

$$B = \sum_{i=1}^{n} p_i B_i \quad \text{and} \quad p_i = F_i(t_i) = P(\eta_i \leq t_i)$$

MindFlayer: Time complexity

$$p_j = F_j(t_j) = P(\eta_j \leq t_j)$$

$$\Delta = f(x^0) - f^{\inf}$$

$$\min_{m \in [n]} \left\{ \left(\frac{1}{m} \sum_{j=1}^{m} \frac{p_j}{\tau_j + t_j} \right)^{-1} \left(\frac{S}{m} + \frac{1}{m} \sum_{j=1}^{m} p_j \right) \frac{\Delta L}{\varepsilon} \right\}$$

$$S = \max\left\{ \frac{\sigma^2}{\varepsilon}, 1 \right\}$$

Experiments on Logistic Regression

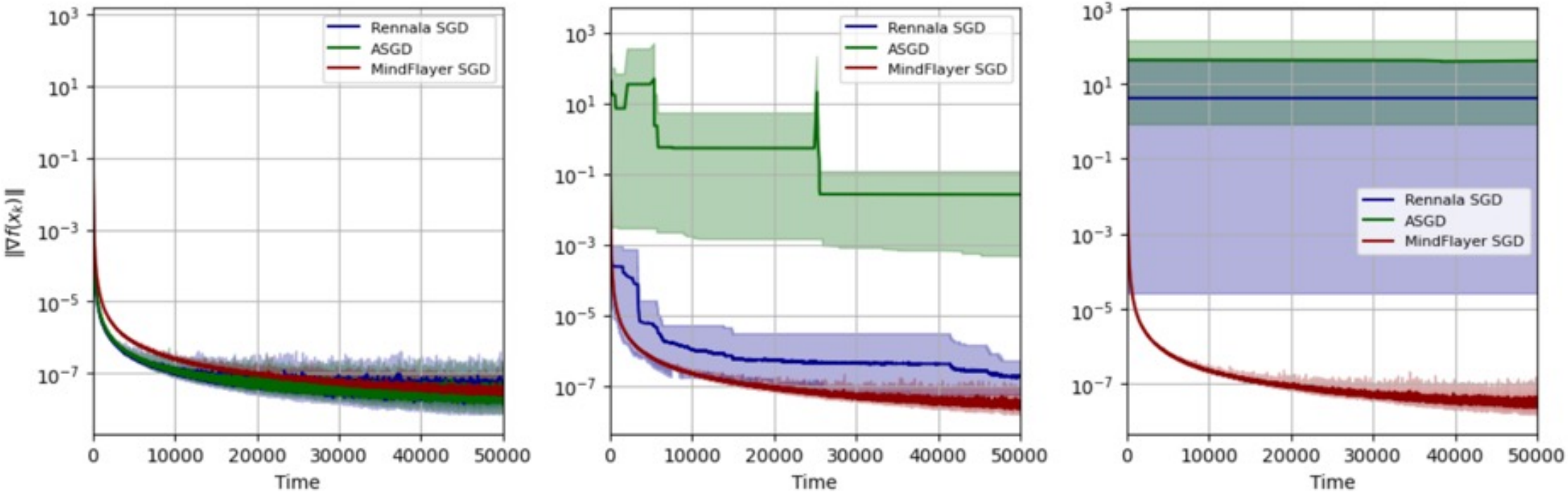
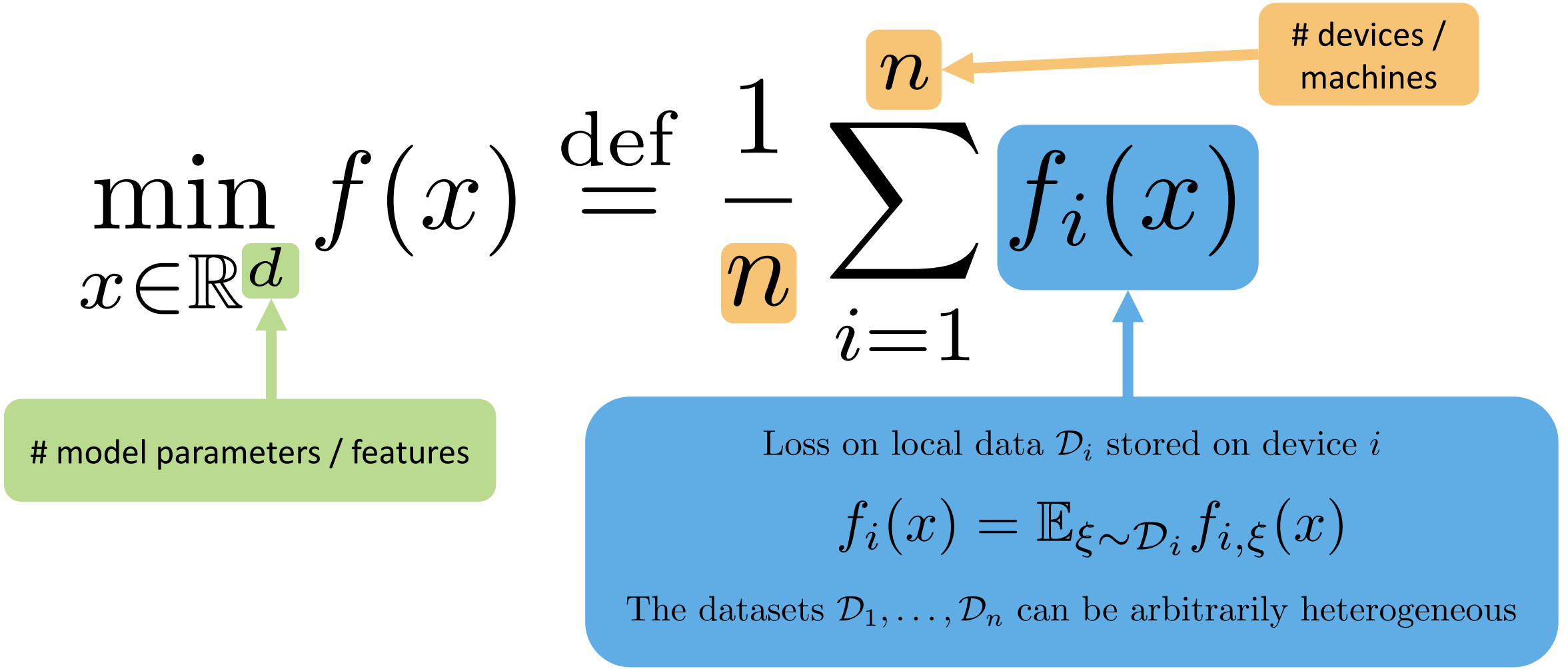


Figure 1: We ran an empirical experiment[2] where we employ the same $\mathcal{J}_i = \text{Lognormal}(0, s)$ distribution for all clients $i \in [n]$, with varying standard deviations s. Specifically, we set $s = 1$ for the left, $s = 10$ for the middle, and $s = 100$ for the right. Additionally, we set $\tau_i = \sqrt{i+1}$. As we observe, with an increase in the variance of the distribution, MindFlayer SGD demonstrates the ability to significantly outperform Rennala SGD and ASGD.


Vecna

Heterogeneous Regime



Vecna: Time complexity

$$p_j = F_j(t_j) = P(\eta_j \leq t_j)$$

$$\Delta = f(x^0) - f^{\inf}$$

$$\frac{\Delta L}{\varepsilon}\left(\tau_n + t_n + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{\tau_i + t_i}{p_i}\right]\frac{\sigma^2}{n\varepsilon} + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{1-p_i}{p_i}(\tau_i + t_i)\right]\frac{\Delta L}{n\varepsilon}\right)$$


The End