Polaris: Accelerating Asynchronous Federated Learning with Client Selection

Yufei Kang, *Member, IEEE,* Baochun Li, *Fellow, IEEE*

Abstract—Federated learning has garnered significant research attention as a privacy-preserving learning paradigm. Asynchronous federated learning has been proposed to improve scalability by accommodating slower clients, commonly referred to as stragglers. However, asynchronous federated learning suffers from slow convergence with respect to wall-clock time, due to the existence of data heterogeneity and staleness. Existing strategies struggled to tackle both difficulties for a wide range of deep learning models. To address the problem, we propose *Polaris*, a theoretically sound design and a new take to client selection for asynchronous federated learning. With *Polaris*, we first theoretically investigated the design space of client sampling strategies from a geometric optimization perspective, taking both data heterogeneity and staleness into account. Our design is not only theoretically proven, but also thoroughly tested in our reproducible experimental open-source testbed. Our experimental results demonstrates overwhelming evidence that *Polaris* outperformed existing state-of-the-art client selection strategies by a substantial margin over a wide variety of tasks and datasets, as we train image classification models using `CIFAR-10`, `CIFAR-100`, `CINIC-10`, `Federated EMNIST`, and a language modeling model using the `Tiny Shakespeare` dataset. Further, our extensive array of ablation studies have also shown that *Polaris* is both scalable and robust as the size of datasets scale up and data heterogeneity vary.

Index Terms—Federated Learning, Distributed Machine Learning, Cloud Computing.

1 Introduction

As one of the emerging distributed machine learning paradigms, federated learning (FL) [1] has attracted widespread research attention, thanks to its merits in preserving data privacy. In federated learning, a central server coordinates training among edge devices, referred to as clients, to build a shared model. The clients send their model updates only and keep their raw data locally, allowing for collaboration without revealing private data. Federated learning has been used in a variety of real-world applications, *e.g.* Apple's Siri for automatic speech recognition.

Yet, most existing mechanisms on federated learning failed to scale efficiently beyond a few hundred clients with the presence of extremely slow clients, called *stragglers*. This is because they assumed fully synchronized aggregation of client updates and the server always waits for stragglers before aggregation. To alleviate the negative impact of stragglers on model convergence, asynchronous federated learning was proposed [2]. In asynchronous design, the server can perform aggregation as long as a portion of the clients have reported [3], which helps reduce the time it takes to complete each round of aggregation.

While it improves training performance with respect to the elapsed wall-clock time, such performance improvement brought by asynchronous federated learning can be limiting due to two reasons: *statistical heterogeneity* and *staleness*. With respect to statistical heterogeneity, training data is not independent and identically distributed (i.i.d.), local updates from different clients may diverge and global model convergence may be slower. With respect to staleness, when slow clients eventually finish their local training, their reported model updates may be based on a global model in a much earlier round that is already out-of-date. These staled model updates adversely affect model convergence in federated learning.

In the literature, existing studies focused on two strategies for improving the convergence performance. *First*, adaptive server aggregation algorithms were designed such that model updates from clients with worse data quality or with severe staleness are associated with smaller aggregation weights. However, these algorithms do not materially affect the wall-clock time it takes to train and aggregate in each communication round.

Second, adaptive sampling strategies were proposed to sample clients with good data and prompt responses with higher probabilities. With these strategies, data quality is typically evaluated based on local gradients or losses, while the response time is measured by the wall-clock training time or staleness. However, clients with good data may have long training times, which presents a challenge for balancing data quality and response time. Solutions to such a problem has only recently been attempted with heuristic designs [4], with limited generality and theoretical analysis. We need a more systematic and theoretically sound design to better address the need for balancing data quality and client response times. This motivates our work on a new design for client sampling.

In this paper, with the presence of system and statistical heterogeneity, we aim to explore the optimal way of sampling clients in asynchronous federated learning that maximizes performance. More specifically, our objective is to minimize the wall-clock training time while guaranteeing

- *Yufei Kang and Baochun Li are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada, M5S 3G4.*
  E-mail: yufei.kang@mail.utoronto.ca, bli@ece.toronto.edu.

model convergence. To achieve this in a theoretically sound approach, we formulate our objective as an optimization problem, with convergence as constraints and client sampling probabilities as variables.

Theoretically, to ensure model convergence, we derive convergence bounds for asynchronous federated learning that characterize the relationship between client sampling probabilities and the number of rounds for the first time. Based on this, we explore how the wall-clock training time and clients sampling probabilities affect the total training time, given both statistical heterogeneity and staleness. Formally, we develop an estimation of the total training time as an explicit optimization objective. Furthermore, to make it easily solvable by common optimization tools, we make a relaxation and approximate the non-convex problem to a geometric optimization problem with the client sampling probability as the only variable.

In addition, we notice that although extensive research has been carried out on estimating statistical heterogeneity based on either gradients or losses in local optimization, it is unclear how different methods impact the estimation results. To further exacerbate the problem, no agreement has been reached on an effective estimation method. To fill this gap, we conduct experiments with a variety of estimation methods based on gradients, losses and model updates, with different averages across epochs and batches. Supported by our results, we conclude that statistical estimators have little impact on estimating the local data quality and thus client sampling. To avoid extra computation costs for local clients, we estimate data quality by model updates in this paper.

Building on the insights gained from our theoretical and empirical work, we introduce a new client sampling algorithm, called *Polaris*, for asynchronous federated learning. Our algorithm operates on the server side and only requires model updates from clients, ensuring the privacy of all participants and encouraging greater participation in the federated learning process. We have implemented *Polaris* using our own open-source framework, *Plato*, designed specifically for fair across-the-board performance comparisons in federated learning research.

Highlights of our original contributions are as follows. First, we establish the analytical relationship between the wall-clock training time and the client sampling probability, considering statistical heterogeneity and staleness in asynchronous federated learning, and ensuring convergence. Second, we model the sampling problem as a geometric optimization problem. Empirically, we demonstrate that different methods for estimating local data quality produce similar results in client sampling. Based on these insights, we propose *Polaris*, our new sampling algorithm. We evaluate the performance of *Polaris* on multiple datasets, including `CIFAR-10`, `CIFAR-100`, `CINIC-10`, `Federated EMNIST`, and `Tiny Shakepeare`, and show that it outperforms existing methods. Our experiments demonstrate the only hyperparameter in *Polaris* is not sensitive and can be generally applied to various scenarios. Our extensive ablation studies confirm the effectiveness of *Polaris* in various scenarios with different client sizes, local data partitions and heterogeneity levels.

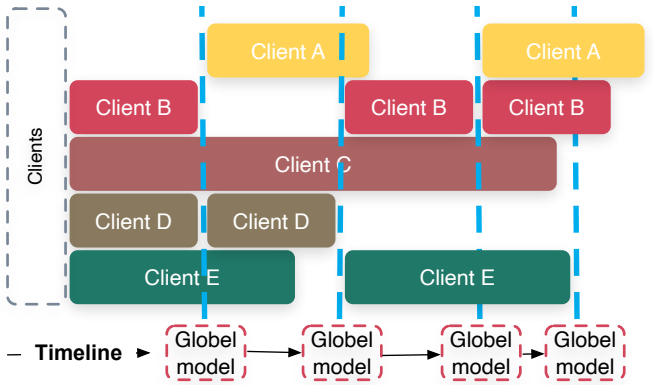


Fig. 1. Asynchronous Federated Learning Workflow

2 Preliminaries

Asynchronous federated learning is a federated learning paradigm that allows multiple clients to train a shared machine learning model, without the need for all clients to be connected to a central server at the same time. Compared to traditional synchronous federated learning's weakness in handling stragglers, the asynchronous paradigm allows the server to perform aggregation as long as reports from a subset of the clients arrive, and therefore accelerates convergence.

In asynchronous federated learning, each client trains a local model on its own data, and then periodically sends reports to a central server. When the amount of arrived reports reaches a minimum bound, the server aggregates the reports and updates the shared global model. This process continues iteratively until the global model has converged to a satisfactory level of accuracy.

Optimization Objective. Consider an asynchronous federated learning system involving a set of $\mathcal{N} = 1, \ldots, N$ clients, coordinated by a central server. Each client i has n_i local training data samples $(x_{i,1}, \ldots, x_{i,n_i})$, and the total number of training data samples across N devices is $n_{total} = \sum_{i=1}^{N} n_i$.

Define $f(\cdot, \cdot)$ as the loss function where $f(\omega; x_{i,j})$ indicates how the machine learning model parameter ω performs on the input $x_{i,j}$, which is the j-th data sample of client i. Thus, the local loss function $F_i(\omega)$ of client i can be defined as

$$F_i(\omega) := \frac{1}{n_i} \sum_{j=1}^{n_i} f(\omega; x_{i,j}). \tag{1}$$

We denote p_i as the aggregation weight of the i-th client such that $\sum_{i=1}^{N} p_i = 1$. Then, by denoting $F(\omega)$ as the global loss function, the goal of asynchronous federated learning is to find the solution ω of the following optimization problem:

$$\min_{\omega} F(\omega) := \sum_{i=1}^{N} p_i F_i(\omega). \tag{2}$$

Let us now denote t to be the index of an asynchronous federated learning round. We describe one round (*e.g.*, the t-th) of the random client selection algorithm as follows. The server uniformly samples a subset of K clients at random (*i.e.*, $K = \|\mathcal{K}\|$ and $\mathcal{K} \subseteq \mathcal{N}$) and broadcasts the latest model

ω^t to the sampled clients. Each sampled client i sets the local model as $\omega_i^{t,0} = \omega^t$, and runs E steps of local optimization on (1) to compute local update model $\omega_i^{t,E}$. After finishing local training, the sampled client sends the updated model back to the server.

When the server receives Ω reports from sampled K clients ($\Omega \leq K$), it performs aggregation with weight p_i and compute a new global model by

$$\omega^{t+1} = \sum_{i}^{\Omega} p_i \omega_i. \tag{3}$$

Notably, ω_i does not have to be local update based on the latest round $\omega_i^{t,E}$. It can be $\omega_i^{t-\tau_i,E}$ that was optimized on a global model in a much earlier round, as staleness $\tau_i < t$ is allowed. This process repeats for a number rounds until the global loss converges.

With such an asynchronous mode of operation as the essential idea, FedBuff [3] is proposed. In FedBuff, the server randomly sampled clients at each round as FedAvg but introduces a buffer to store local updates and only aggregates when the buffer size reaches a certain aggregation goal. Due to its simplicity, FedBuff and its variants with different weighting methods are commonly used.

Slow convergence. While FedBuff alleviates the negative impact of stragglers, it lacks explicit control of slow clients and therefore the staleness across the sampled clients can go unbounded without any punishment. In cases of extreme system heterogeneity, it may suffer from slow convergence with respect to the elapsed wall-clock time. This is because the total training time depends on both the number of training rounds for reaching the target accuracy and the wall-clock time elapsed in each round. Though uniform sampling in FedBuff guarantees that the aggregated model update in each round is unbiased compared to full client participation, the aggregated model may have a high variance due to data heterogeneity, thus requiring more training rounds to converge. Furthermore, with a substantial degree of staleness, local updates adversely affect global model convergence.

Moreover, considering the clients' heterogeneous communication delay and computation speed, uniform sampling also suffers from the straggling effect, as the probability of sampling a straggler within the sampled subset in each round can be relatively high, thus yielding a long per-round time. Therefore, it is preferrable to adopt adaptive client sampling.

3 Problem formulation and Convergence Analysis

Open Challenges. To increase convergence performance with regard to the wall-clock time, fast clients with high quality local data are expected to be prioritized when sampling clients. However, in reality, such ideal clients may be difficult to find. It is more typical that clients are fast with poor data quality or slow with high quality data. Even worse, a small subset of clients are slow and contribute a little to the global model, and they should be avoided.

If we prioritize fast clients, the average round latency will be shortened but the number of training rounds required to reach convergence will increase. To avoid inflating the number of rounds when handling stragglers, client selection should also account for local data quality.

In short, to achieve a shorter time to convergence, which is the product of the average round latency and the number of rounds taken to reach the target accuracy, a good strategy must strike a balance between data heterogeneity and local staleness.

Problem Formulation. Our goal is to minimize the expected total learning time $\mathbb{E}[T_{total}(\mathbf{q}, t)]$, while ensuring that the expected global loss $\mathbb{E}[F(\omega^t(\mathbf{q}))]$ converges to the minimum value F^* with an ϵ-precision, with $\omega^t(\mathbf{q})$ being the aggregated global model after t times of server aggregation with client sampling probability $\mathbf{q}$. This can be described as the following optimization problem:

$$\begin{aligned} \textbf{P1:} \quad \min \mathbf{q}, t \quad & \mathbb{E}[T_{tot}(\mathbf{q}, t)] \\ \text{s.t.} \quad & \mathbb{E}[F(\omega^t(\mathbf{q}))] - F^* \leq \epsilon, \\ & \sum_{i=1}^{N} q_i = 1, \\ & q_i > 0, \forall i \in \mathcal{N}, t \in \mathbb{Z}^+. \end{aligned} \tag{4}$$

The expectation in $\mathbb{E}[T_{tot}(\mathbf{q}, t)]$ and $\mathbb{E}[F(\omega^t(\mathbf{q}))]$ is due to the randomness in client sampling $\mathbf{q}$ and local optimization (*e.g.* SGD).

To explore the optimal client sampling strategy that addresses system and statistical heterogeneity so as to minimize the wall-clock time for model convergence in asynchronous federated learning, we describe the system model as follows.

Sampling: In asynchronous federated learning, to reduce the communication overhead, the server samples a subset of K clients, where $K = \|\mathcal{K}(\mathbf{q})\|$ and $\mathcal{K}(\mathbf{q}) \subseteq \mathcal{N}$, according to a probability distribution $\mathbf{q} = \{q_i, \forall i \in \mathcal{N}\}$ without replacement, where $0 \leq q_i \leq 1$ and $\sum_{i=1}^{N} q_i = 1$.

Statistical Heterogeneity: We consider the asynchronous federated learning setting where the training data are not independently and identically distributed among client devices.

System Heterogeneity: For system heterogeneity, we follow the same setup of [5] and [6]. Denote t_i as the personal round time of client i, which includes both local training time and global communication time. For simplicity, we assume that t_i remains the same across different rounds for each client i, while for different clients i and j, t_i and t_j can be different.

Aggregation with staleness: At the t-th aggregation round, denote η^t as the global learning rate, and $\tau_i(t) \geq 1$ as the staleness of an update contributed by client i. Specifically, when $i \in \mathcal{K}(\mathbf{q})$, the update $\Delta_i(\omega^{t-\tau_i(t)})$ reported by client i was computed by starting point from $\omega^{t-\tau_i(t)}$. When $\tau_i(t) = 1$, there is no staleness in the model update, and more generally $\tau_i(t) > 1$ corresponds to some staleness: i.e. $t - \tau_i(t)$ server updates have taken place between when the client last pulled a model from the server and when the client's update is being incorporated at the server. Therefore, the server aggregation can be described succinctly as

$$\omega^{t+1} = \omega^t + \eta^t \sum_{i=1}^{\Omega} p_i \Delta_i(\omega^{t-\tau_i(t)}). \tag{5}$$

Wall-clock Time: We consider the mainstream asynchronized federated learning model where each sampled client performs multiple E steps of local SGD before sending back their model updates to the server. For asynchronous federated learning, the per-round time T^t is computed as the elapsed time between the last aggregation and the moment Ω-th client arrives. Therefore, the total training time T_{tot} after T rounds is

$$T_{tot}(T) = \sum_{t=1}^{T} T^t. \tag{6}$$

Solving Problem P1, however, is challenging in four aspects. First, it's almost impossible to know how the client sampling probabilities $\mathbf{q}$ and communication round t affect the final model $\omega_t(\mathbf{q})$ during the federated learning process. Besides, it is difficult to obtain an analytical expression result for asynchronous federated learning, and to find out how the client sampling probabilities $\mathbf{q}$ and communication round t affect the wall-clock training time. Additionally, the objective $E[T_{tot}(q, R)]$ is complicated to optimize due to the straggling effect. Therefore, the total learning time minimization problem can be complex and non-convex. Furthermore, there are a variety of methods for estimating the local data heterogeneity, but whether they perform consistently in a wide variety of settings has not been well explored in the literature.

To ensure a tractable convergence analysis, we make the following assumptions throughout.

Assumption 1. (L-smooth): For each client $i \in \mathcal{N}$, F_i is L-smooth, i.e., $\|\nabla f(\mathbf{v}) - \nabla f(\mathbf{w})\| \leq L\|\mathbf{v} - \mathbf{w}\|$ for all $\mathbf{v}$ and $\mathbf{w}$.

Assumption 2. (Strongly-convex): For each $i \in \mathcal{N}$, F_i is μ-strongly convex, i.e., $F_i(\mathbf{v}) \geq F_i(\mathbf{w}) + (\mathbf{v}-\mathbf{w})^T \lambda F_i(w) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{w}\|_2^2$ for all $\mathbf{v}$ and $\mathbf{w}$.

Assumption 3. (Bounded local variance): For each device $i \in \mathcal{N}$, the variance of its stochastic gradient is bounded: $\mathbb{E}\|\nabla F_i(\mathbf{w}_i, \xi_i) - \nabla F_i(\mathbf{w}_i)\|^2 \leq \sigma_i^2$.

Assumption 4. (Bounded local gradient): For each client $i \in \mathcal{N}$, the expected squared norm of stochastic gradients is bounded: $\mathbb{E}[\|\nabla F_i(\mathbf{w}_i, \xi_i)\|^2] \leq G_i^2$.

Assumption 5. (Bounded Staleness when $\Omega = 1$) For all clients $i \in \mathcal{N}$ and for each server step t, the staleness $\tau_i(t)$ between the model version in which client uses to start local training and the model version in which Δ_i is used to update the global model is not larger than $\tau_{max,1}$ when $w = 1$.

Assumption 1–3 are common in the convergence analysis of convex federated learning problems such as l_2-norm regularized linear regression, logistic regression [7], [8]. Nevertheless, the experimental results, to be presented in Sec. 6, show that the proposed method based on geometric optimization also works well for non-convex loss functions.

To account for the local data heterogeneity and explore its influence on convergence, we do not follow the assumption made in [7], [8], which states that G_i is uniformly bounded by a universal G for all clients. Instead, we make the Assumption 4 same as [9] that each client i has its unique local G_i, and G_i is considered as an important factor when we formulate and solve the optimization problem for optimal client sampling.

TABLE 1
Parameters

$\mathcal{K}(\mathbf{q})$	The subset of sampled clients
$\mathbf{q}$	Client sampling probability distribution
$\mathbf{w}$	Global model weights
$\mathbf{w}_i$	Local model weights for client i
M	The total number of sampled client at each round
p_i	Aggregation weight for client i
τ_i	Staleness for client i
T	Total round number
t	Current round
G_i	Local statistical bound for client i
η	Global learning rate on the server
Ω	The number of staled clients
N	Total number of clients

Assumption 5 is commonly made under asynchrony [3] where staleness cannot go unbounded.

Convergence Analysis. We now show how aggregate clients' model updates are to be aggregated under sampling probabilities $\mathbf{q}$, such that the aggregated global model is unbiased compared to that with full client participation, which leads to our convergence result. We list the notations used in this section in Table 1 for ease of reading.

We first define the ideal weighted model aggregated from full client participation with no staleness in round t as

$$\bar{\mathbf{w}}^t = \sum_{i=1}^{N} \mathbf{w}_i^t. \tag{7}$$

We then define the actual weighted model aggregated from M arrived clients, Ω of which has staleness of τ_i, as $\tilde{\mathbf{w}}^t$.

With these, we can derive Lemma 1.

Lemma 1. At t-th communication round of asynchronous federated learning, when client $\mathcal{K}(\mathbf{q})^t$ are sampled with probability $\mathbf{q} = q_1, \ldots, q_N$, and their local updates are aggregated as

$$\tilde{\mathbf{w}}^{t+1} \leftarrow \frac{1}{M}\Big(\sum_{i=1}^{M-\omega} \frac{p_i}{q_i}\mathbf{w}_i^{t+1} + \sum_{i=M-\omega+1}^{M} \frac{p_i}{q_i}\mathbf{w}_i^{t+1-\tau_i}\Big), \tag{8}$$

where aggregation weights p_i^t of client i at t-th round satisfies

$$\sum_{i}^{M} p_i^t = 1, \tag{9}$$

we have

$$\mathbb{E}_{\mathcal{K}(\mathbf{q})^t}[\tilde{\mathbf{w}}^{t+1}] = \bar{\mathbf{w}}^{t+1}. \tag{10}$$

Proof sketch: We first take the expectation over the weighted aggregation formula (8) on both sides. Note that the sampling probability $\sum_i^N q_i = 1$. Together with (9), we can obtain (10) with some mathematical derivations.

Based on Lemma 1, we derive Lemma 2.

Lemma 2. As the expected aggregated global model $\mathbb{E}_{\mathcal{K}(q)^t}[\tilde{\mathbf{w}}^{t+1}]$ is unbiased compared to ideal model $\bar{\mathbf{w}}^{t+1}$ with full client participation, the expected difference of the expected aggregated global model $\mathbb{E}_{\mathcal{K}(q)^t}[\tilde{\mathbf{w}}^{t+1}]$ and ideal model $\bar{\mathbf{w}}^{t+1}$ at the t-th round is bounded as follows:

$$\mathbb{E}_{\mathcal{M}(\mathbf{q})^t}\|\tilde{\mathbf{w}}^{t+1} - \bar{\mathbf{w}}^{t+1}\|^2 \leq \frac{4}{M}\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i}(\eta_t E)^2 + \frac{\omega}{M}\eta_t \sum_{i=1}^{N} p_i^2 q_i \tau_i G_i. \quad (11)$$

Proof sketch:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{K}(\mathbf{q})^t}\|\tilde{\mathbf{w}}^{t+1} - \bar{\mathbf{w}}^{t+1}\|^2 \\
=&\mathbb{E}_{\mathcal{M}(q)^t}\|\frac{1}{M}(\sum_{i=1}^{M-\omega}\mathbf{w}_i^{t+1} + \sum_{i=M-\omega+1}^{M}\mathbf{w}_i^{t+1-\tau_i}) - \bar{\mathbf{w}}^{t+1}\|^2 \\
=&\mathbb{E}_{\mathcal{K}(q)^t}\|\frac{1}{M}(\sum_{i=1}^{M-\omega}\mathbf{w}_i^{t+1} + \sum_{i=M-\omega+1}^{M}\mathbf{w}_i^{t+1}) \\
&+\frac{1}{M}(\sum_{i=M-\omega+1}^{M}(\mathbf{w}_i^{t+1-\tau_i} - \mathbf{w}_i^{t+1})) - \bar{\mathbf{w}}_i^{t+1}\|^2 \\
\leq&\mathbb{E}_{\mathcal{K}(q)^t}\|\mathbf{w}^{t+1} - \bar{\mathbf{w}}^{t+1}\|^2 \\
&+\mathbb{E}_{\mathcal{K}(q)^t}\|\frac{1}{M}\underbrace{\sum_{i=M-w+1}^{M}\mathbf{w}^{t+1-\tau_i} - \mathbf{w}^{t+1})}_{\text{local drift}}\|^2 \\
\leq&\frac{4}{M}\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i^2}(\eta_t E)^2 + \frac{\omega}{M}\eta_t\sum_{i=1}^{N}p_i^2 q_i\tau_i G_i
\end{aligned}$$

Based on Lemma 1 and Lemma 2, we present the main convergence result for arbitrary client sampling in Theorem 1.

Theorem 1. **(Convergence Upper Bound)** Let Assumptions 1 to 5 hold, $\gamma = \max\{\frac{8L}{\mu}, E\}$, and the decaying global learning rate $\eta_t = \frac{2}{\mu(\gamma+t)}$. For given client sampling probabilities $\mathbf{q} = \{q_i, \ldots, q_N\}$, the corresponding aggregation weights p_i, and staleness τ_i, the optimality gap after t rounds satisfies

$$\mathbb{E}[F(\mathbf{w}(\mathbf{q})^T)] - F^* \leq \frac{1}{T}(\alpha\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i} + \beta + \theta), \quad (12)$$

where

$$\beta = \frac{2L}{\mu^2 E}B + \frac{12L^2}{\mu^2 E}\Gamma + \frac{4L^2}{\mu E}\|\mathbf{w}_0 - \mathbf{w}^*\|^2, \quad (13)$$

and $\alpha = \frac{8LE}{\mu^2 K}$ with $B = \sum_{i=1}^N p_i^2\sigma_i^2 + 8\sum_{i=1}^N p_i G_i^2 E^2$ and $\Gamma = F^* - \sum_{i=1}^N p_i F_i^*$ and $\theta = \frac{\omega}{M}\eta_t\sum_{i=1}^N p_i^2 q_i\tau_i G_i$.

Proof Sketch. First, following the similar proof of convergence with arbitrary client sampling in [6], we show that

$$\mathbb{E}[F(\mathbf{w}^T(\mathbf{q}))] - F^* \leq \frac{1}{T}(\alpha\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i} + \beta), \quad (14)$$

where $\mathbb{E}[F(\bar{\mathbf{w}}^T)]$ is the expected global loss after T rounds with clients sampled following probability distribution $\mathbf{q}$, and α, β are the same at in (12).

In Lemma 2, we obtain

$$\mathbb{E}_{\mathcal{M}(\mathbf{q})^t}\|\tilde{\mathbf{w}}^{t+1} - \bar{\mathbf{w}}^{t+1}\|^2 \leq \frac{4}{M}\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i}(\eta_t E)^2 + \frac{\omega}{M}\eta_t\sum_{i=1}^{N}p_i^2 q_i\tau_i G_i. \quad (15)$$

By induction, we can obtain a non-recursive bound on $\mathbb{E}_{\mathcal{K}(q)^t}\|\mathbf{w}^T - \mathbf{w}^*\|^2$, which is converted to a bound on $\mathbb{E}[F(\mathbf{w}^t(q))] - F^*$ using L-smoothness in Assumption 1. Finally, we show that the main difference of the contraction bound compared to full client participation is the sampling variance in (15), which yields the term in (12).

Remarks. Our convergence bound in Theorem 1 partially alleviates the difficulties in solving optimization problem P1. First, it characterizes the relationship between client sampling probabilities $\mathbf{q}$ and the number of rounds T for convergence. Notably, our bound generalizes the convergence where clients are uniformly sampled ($q_i = \frac{1}{N}$) or weighted sampled ($q_i = p_i$). More importantly, It lays a foundation to obtain an analytical expression result for asynchronous federated learning and find out how client sampling probabilities $\mathbf{q}$ and communication round T affect the wall-clock training time. It motivates the following optimal client sampling design for asynchronous federated learning with statistical and system heterogeneity and provides significant theoretical support.

4 Algorithm Design

Substituting the convergence constraint we obtained in Theorem 1, we can write the original optimization problem as:

$$\begin{aligned}
\textbf{P2:}\quad \min \mathbf{q}, t \quad & \mathbb{E}[T_{tot}(\mathbf{q}, t)] \\
\text{s.t.} \quad & \frac{1}{t}(\alpha\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i} + \beta + \theta) \leq \epsilon, \\
& \sum_{i=1}^{N} q_i = 1, \\
& q_i > 0, \forall i \in \mathcal{N}, t \in \mathbb{Z}^+.
\end{aligned} \quad (16)$$

In asynchronous federated learning, a central server waits for a minimum number of client reports before it performs aggregation. However, when a few malicious stragglers exist and they delay local updates on purpose, the system would be threatened. Hence, for system robustness to malicious stragglers, the central server waits for $T_{\max}$ at most at each round. Therefore, we obtain

$$\mathbb{E}[T_{tot}(\mathbf{q}, T)] = T_{\max} T. \quad (17)$$

Further, the original optimization problem can be approximated as:

$$\begin{aligned}
\textbf{P3:}\quad \min \mathbf{q}, t \quad & T_{\max} T \\
\text{s.t.} \quad & \frac{1}{t}(\alpha\sum_{i=1}^{N}\frac{p_i^2 G_i^2}{q_i} + \beta + \theta) \leq \epsilon, \\
& \sum_{i=1}^{N} q_i = 1, \\
& q_i > 0, \forall i \in \mathcal{N}, t \in \mathbb{Z}^+.
\end{aligned} \quad (18)$$

For **P3**, relaxing T as a continuous variable, suppose $(\mathbf{q}^*, t^*)$ is the optimal solution, then we have

$$\frac{1}{t^*}(\alpha \sum_{i=1}^{N} \frac{p_i^2 G_i^2}{q_i^*} + \beta + \theta) = \epsilon. \tag{19}$$

If the above holds, we can always find a $t' < t^*$ that satisfies it with equality, with $(\mathbf{q}^*, t^*)$ leading to a smaller value. Therefore, we can obtain t from the above equation and the objective function of the optimization problem would be

$$T_{\max}(\alpha \sum_{i=1}^{N} \frac{p_i^2 G_i^2}{q_i} + \beta + \theta), \tag{20}$$

where $\mathbf{q}$ is the only optimization variable. Hence, we obtain the approximating optimization problem below

$$\begin{aligned} \min_{\mathbf{q}} \quad & T_{\max}(\alpha \sum_{i=1}^{N} \frac{p_i^2 G_i^2}{q_i} + \beta + \theta) \\ \text{s.t.} \quad & \sum_{i=1}^{N} q_i = 1, \\ & q_i > 0, \forall i \in \mathcal{N}. \end{aligned} \tag{21}$$

where

$$\beta = \frac{2L}{\mu^2 E} B + \frac{12L^2}{\mu^2 E} \Gamma + \frac{4L^2}{\mu E} \|\omega_0 - \omega^*\|^2, \tag{22}$$

and $\alpha = \frac{8LE}{\mu^2 K}$, $B = \sum_{i=1}^{N} p_i^2 \sigma_i^2 + 8 \sum_{i=1}^{N} p_i G_i^2 E^2$, $\Gamma = F^* - \sum_{i=1}^{N} p_i F_i^*$ and $\theta = \frac{1}{K} \Omega \eta_t \sum_{i=1}^{\Omega} p_i^2 q_i \tau_i G_i$.

Clean up the constant terms without optimization variable $\mathbf{q}$, the approximating optimization problem can be expressed as:

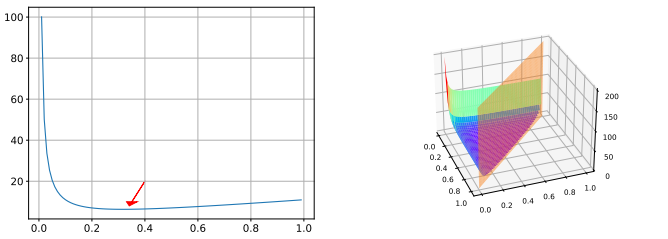
$$\begin{aligned} \textbf{P4:} \quad \min_{\mathbf{q}} \quad & \sum_{i=1}^{N} \frac{p_i^2 G_i^2}{q_i} + A \sum_{i=1}^{N} p_i^2 q_i \tau_i G_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} q_i = 1, \\ & q_i > 0, \forall i \in \mathcal{N}. \end{aligned} \tag{23}$$

where A is a constant number.

Remarks. So far, we have addressed the third challenge for solving the optimization problem, by analytically discussing how client sampling probabilities $\mathbf{q}$ and communication round T affect the wall-clock training time. Further, we have completed the problem formulation for asynchronous federated client sampling.

Note that no additional information is required when a server decides which clients to choose and therefore no additional communication costs are introduced. From **P4**, we can tell that the server assigns a higher sampling probability to clients with a higher model contribution G and lower staleness τ, which satisfies our intuition. Besides, the model aggregation weights p_i play an important role in balancing the tradeoff.

Notably, problem **P4** is a geometric problem and here we conceptually visualize the geometric formulation 3D and 2D in Figure 2. Geometric optimization is widely applicable and can be used in a variety of fields, including computer graphics, computer-aided design, and machine learning. From the plot, we know that the optimal solutions exist.



(a) Geometric Optimization (2D) (b) Geometric Optimization (3D)

Fig. 2. Optimal points exist in geometric formulations.

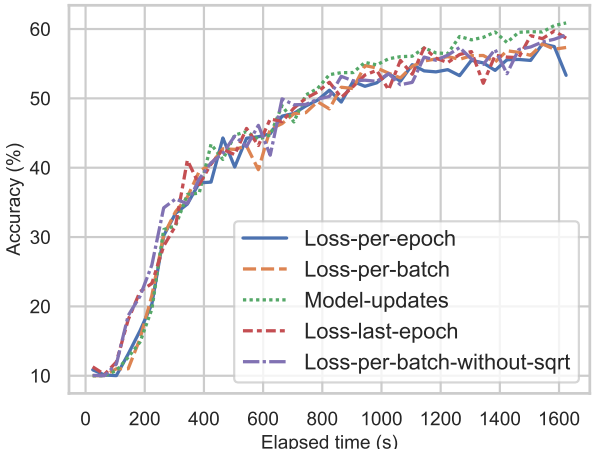


Fig. 3. Different estimators perform similarly to one another with the `CIFAR-10` dataset.

Geometric optimization algorithms are often efficient and effective at finding the optimal solution, especially when combined with gradient-based optimization techniques. For example, we can solve for the client sampling probability $\mathbf{q}^*$ very efficiently via convex optimization tools, e.g. CVX, without incurring too much computation cost.

Influence of Statistical Estimators. In the formulated optimization problem **P4**, the local data heterogeneity G_i plays an important role in deciding which clients to select. However, it cannot be directly measured as an observable parameter. In existing studies, researchers estimate local data heterogeneity with a variety of methods for client sampling. Unfortunately, the impact of different estimation methods has not been discussed and compared with one another in a detailed manner.

To explore the influence of different estimators on client sampling, we ran experiments in Pisces, the state-of-the-art, with statistical estimators. We trained `ResNet-18` models on the `CIFAR-10` dataset in a setting of 200 clients, 20 of whom were selected at each round. For system heterogeneity, we apply the Zipf distribution with a parameter of 1.2. For statistical heterogeneity, we apply a Pareto distribution with a concentration factor of 1. Here, we include 5 estimators:

Loss-per-batch. At each communication round, a client estimates the loss via the weighted moving average with the parameter of 0.9 of the losses across batches over 5 local

epochs. Subsequently, the client determines the statistical heterogeneity by employing the following

$$G_i = n_i \sqrt{\frac{1}{n_i}\sqrt{\text{loss}}}, \tag{24}$$

where n_i is the sample number on client i.

Loss-per-epoch. In each epoch, a client averages the loss values over batches. Then, the client estimates the statistical heterogeneity via the weighted moving average with the parameter of 0.9 over local epochs.

Loss-last-epoch. In the last epoch, a client estimates loss via the averages loss values over all batches. Then, client compute statistical heterogeneity via formula (24).

Loss-per-epoch without square root. The client estimates local loss values as *Loss-per-epoch*. Additionally, it calculates the statistical heterogeneity following

$$G_i = n_i \sqrt{\frac{1}{n_i}\text{loss}}. \tag{25}$$

Model updates. At each communication round, a client estimates the statistical heterogeneity by the norm of the local model updates.

The results are plotted in Fig. 3, which shows that the achieved accuracy may vary slightly with different estimators. For instance, Pisces with model updates based estimation attains the highest test accuracy among all comparisons. However, all runners exhibit similar convergence performances. Therefore, we can conclude that estimation methods do not affect sampling strategies, which means that the optimization formulation is applicable to all estimators for G_i. This addresses the fourth challenge.

As the loss-based estimation methods require extra computation about the local loss values on the client side, we estimate G_i via the local model updates, so that the central server can figure out the optimal sampling probability without requiring additional information.

Fair evaluation for unexplored clients. Since the server does not have the access to clients before sampling, it is essential to make informed predictions for unexplored clients to facilitate fair sampling decisions. We observe that the model updates within the federated learning process are continuously evolving. To maintain fairness, we use the average of the most recent updates from participating clients as the basis for evaluating unexplored clients, and substitute it with the actual value once the client is selected. The prediction is calculated following:

$$G_{\text{unexplored}} = \gamma \cdot \bar{G}_{\text{latest\_updates}}, \tag{26}$$

where the hyperparameter γ is a scale number. For the initial round, clients were chosen uniformly.

***Polaris*.** So far, we address the client sampling model in asynchronous federated learning and be able to solve out the sampling probability directly on the server side. We propose the *Polaris* client sampler.

The *Polaris* sampler can be seamlessly integrated into all asynchronous federated learning systems as a pluggable component without the need for additional intermediate values. This enhances the system's flexibility and enables it to evolve alongside the advancements in asynchronous federated learning. We provide a summary of the *Polaris* samplers with FedBuff aggregation rule below.

The *Polaris* sampler operates as follows: In each communication round t, the server calculates the probability distribution $\mathbf{q}$ on the server side and uses it to sample a subset of K clients. The latest model ω^t is then broadcasted to the selected clients, where the server conducts a uniform sampling in the first round. Each selected client i sets the local model and performs E steps of local optimization to generate the updated model $\omega_i^{t,E}$. Once the local training is complete, the client sends the updated model back to the server. After receiving Ω reports from the selected K clients ($\Omega \leq K$), the server aggregates the models using the weight p_i and computes a new global model. Additionally, the server updates records for both reported clients and unexplored clients and calculates the client probability distribution $\mathbf{q}$ for next round.

Algorithm 1 FedBuff with *Polaris* Sampler

Input: Initial model $\omega_{\mathbf{0}}$, minimum aggregation bound Ω, number of local steps E
Output: Final global model parameter $\omega_{\mathbf{T}}$
for communication round $t = 1, \ldots, T$ **do**
  Server calculates optimal sampling probability $\mathbf{q}$ and selects a subset of K clients following the distribution.
  for sampled client $i \in \mathcal{K}(\mathbf{q})$ **do**
    Client i does local optimization for E step.
    Client i reports model $\omega_i^{t,E}$, staleness τ_i to server.
  end for
  if server receives Ω reports from sampled clients **then**
    Server performs aggregations over Ω reports and updates records for reported clients.
    Server makes predictions for unexplored clients following formula (26).
  end if
end for

5 Implementation

Platform. We conducted all of our experiments on *Plato*, an open-source framework designed for scalable federated learning research. The *Plato* framework uses object-oriented subclassing, leveraging Python 3's ABC library and Data Classes. Additionally, the framework supports defining callback classes and customizing trainers by providing a list of custom callback classes. With its ability to scale to a large number of clients and its extensibility to accommodate a wide variety of datasets, models, and FL algorithms, *Plato* is an ideal tool for federated learning research. The framework abstracts away the underlying ML drivers using convenient APIs, making it agnostic to deep learning frameworks such as *TensorFlow*, *PyTorch*, and *MindSpore*.

Plato facilitates client-server communication via industry-standard WebSockets. The server can either run on the same GPU-enabled physical machine as its clients for emulation research testbeds or be deployed in a cloud datacenter. *Plato* supports heterogeneity in both client local time and local data distribution during sampling. The framework allows for specifying random seeds for random number generators and protecting random number generation from third-party frameworks

using random.getstate() and random.setstate(). Unlike other frameworks that only count communication round, *Plato* supports wall-clock time measurement, and we used a pre-specified constant duration instead of measuring the actual wall-clock time of local training loops on each client to ensure the same set of clients would be selected in each round across different algorithms and runs. With these mechanisms in place, *Plato* enables fully reproducible experiments through its reproducible mode, where the same set of clients and data samples are selected across runs.

Polaris. We devoted considerable effort to implementing *Polaris* with just 220 lines of code. Our goal was to create an efficient and lightweight solution that can easily integrate with various clients and local trainers, making it a versatile and plug-and-play module.

Pisces. Pisces is a cutting-edge client sampling algorithm for asynchronous federated learning that calculates the overall utility for each client and selects the top-k clients for model training. To ensure fairness in model testing, we have implemented Pisces [4] on top of the *Plato* framework with just 350 lines of code. The implementation comprises independent server, client, and trainer modules that are highly readable, easy to reuse, and reproduce.

Oort. Oort is a state-of-the-art client sampling algorithm used in synchronous federated learning. It ranks clients based on both their system time and statistical heterogeneity and selects the top-ranked clients in each communication round. To ensure fair model testing, we have implemented Oort [10] on top of *Plato*.

FedBuff. FedBuff is a popular and widely used asynchronous federated learning algorithm due to its simplicity. At each round, FedBuff randomly selects a subset of clients and performs model aggregation over the selected clients. We have implemented FedBuff on top of Plato and made the entire code available as open source on [3].

6 Performance Evaluation

6.1 Experimental Setup

We run all experiments using *Plato* framework on a server, with 3 NVIDIA RTX A4500 GPUs with 20GB of CUDA memory. In each run, 200 clients participate and up to 10% of them are selected by the server sampler at each round. The minimum aggregation bound for asynchronous federated learning is 10 so that when 50% of the selected clients arrive, the server performs aggregation and updates the global model.

For system heterogeneity, we configure clients' processing latency to follow Zipf distribution with the parameter s of 1.2. Zipf distribution models a practical scenario with various device speeds such that the majority of devices are fast and a few devices are stragglers, while a medium number of devices are with middle-of-the-road speed. With $s > 1$, it satisfies the convergence of the generalized harmonic series.

To additionally introduce data heterogeneity, we set up the local data distribution for each clients with two methods: for datasets that are collected as a centralized dataset, for example `CIFAR-10`, `CIFAR-100` and `CINIC-10`, we sample local data partitions following Dirichlet distribution with the concentration parameter of 1. In Dirichlet distribution, with $\alpha \leq 1$, the samples will be highly concentrated in a few labels, and all the rest labels will have almost no samples, while with $\alpha > 1$, the samples will be dispersed almost equally among all the labels. For datasets directly collected from federated scenarios, for example `Federated EMNIST` and `Tiny Shakepeare`, we directly allocate one partition to a client to preserve native non-IID properties.

TABLE 2
Parameters in experimental evaluation.

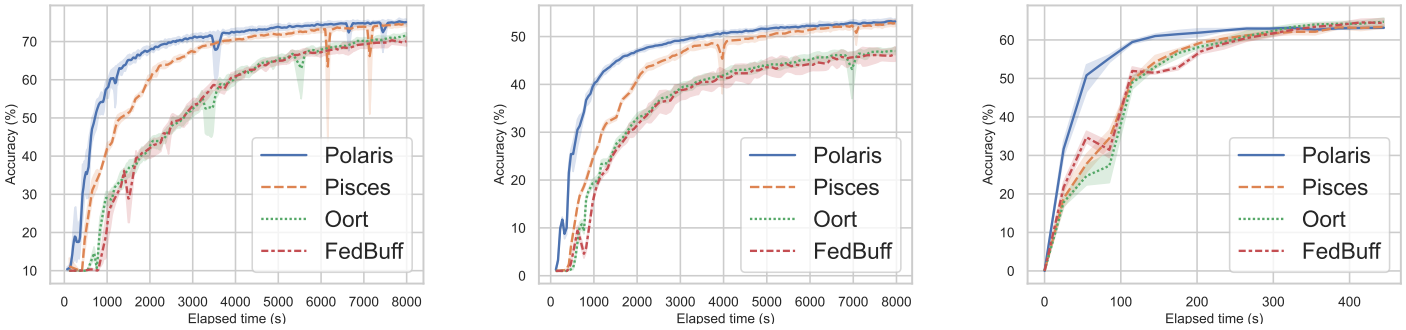
Dataset	`CIFAR-10` `CIFAR-100`	`CINIC-10`	`FEMNIST`	`Tiny Shakespeare`
Model	`Resnet-18`	`VGG-16`	`Lenet-5`	`Bert`
Partition size	$10^3/3*10^3$	$2*10^4$	–	–
Local epochs	5	1	5	5
Batch size	10	128	32	32
Optimizer	`SGD`	`SGD`	`SGD`	`SGD`
Learning rate	0.01/0.1	0.01	0.01	0.01
Momentum	0.9	0.5	0.9	0.9
Weight decay	10^{-4}	0	0	0
Scheduler	`LambdaLR`	`LambdaLR`	–	–
Gamma	0.1	–	–	–
Steps	80, 120	–	–	–

Datasets and Models. To evaluate *Polaris*' effectiveness on asynchronous federated learning training tasks, we conducted experiments over five datasets, falling into two categories:

Synthetic non-iid datasets. Here, we introduce three image classification datasets: the `CIFAR-10` dataset with 60k colored images in 10 classes, the `CIFAR-100` with 60k colored images in 100 classes, and a larger dataset, `CINIC-10` with 270k colored images in 10 classes. We train a `ResNet-16` model on `CIFAR-10` and `CIFAR-100`, and a `VGG-16` model on `CINIC-10`.

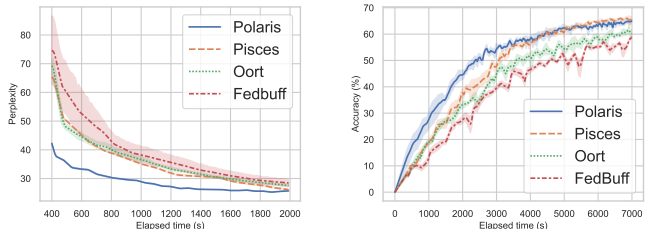
Real-world non-iid datasets. For image classification tasks, we use `Federated MNIST` (`FEMNIST`) dataset with 805k greyscale images in 62 categories built by partitioning the data in `Extended MNIST` based on the writer of the digit/character; For next word prediction tasks, we use `Tiny Shakespeare` dataset, which is built from The Complete Works of William Shakespeare via considering each speaking role in each play a different device. We train a `Lenet-5` model on `FEMNIST` and a `BERT` model on `Tiny Shakespeare`.

Parameters. For all training tasks, we use `SGD` as the local optimizer and summerize all configurations in Table 2. Notably, to guarantee model convergence, we assign larger partition sizes to clients for large image datasets such as `CIFAR-100` and `CINIC-10`. For real-world datasets `FEMNIST` and `Tiny Shakespeare`, as each data partition is directly collected from a real-world role, the partition size is various among each client.



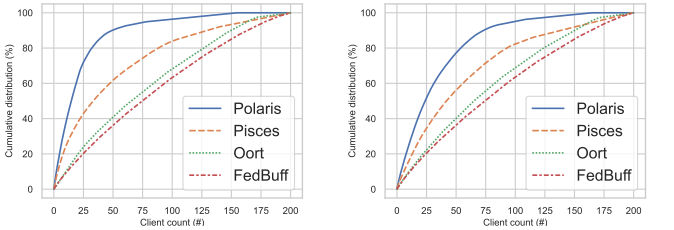
(a) ResNet-18 on CIFAR-10 (b) ResNet-18 on CIFAR-100 (c) VGG-16 on CINIC-10

Fig. 4. *Polaris* outperforms comparisons on synthetic datasets with dirichlet concentration factor of 1.



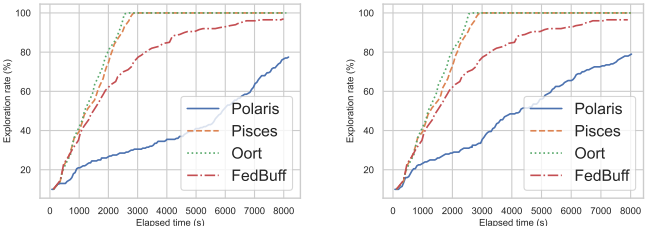
(a) BERT on Tiny Shakespeare (b) LeNET-5 on FEMNIST

Fig. 5. *Polaris* Outperforms Comparisons on Real-World datasets.



(a) ResNet-18 on CIFAR-10 (b) ResNet-18 on CIFAR-100

Fig. 7. The Cumulative Distribution Function for Client Selection with Dirichlet concentration factor of 1.



(a) ResNet-18 on CIFAR-10 (b) ResNet-18 on CIFAR-100

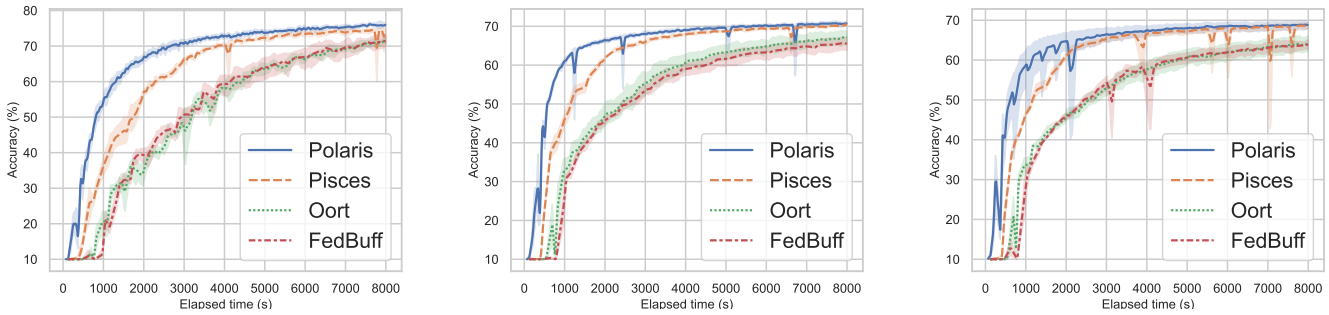
Fig. 6. Exploration rate vs. elapsed time with Dirichlet concentration factor of 1.

6.2 End-to-End Performance

Metrics and Visualization. We primarily measure the elapsed time taken to reach the target accuracy (perplaxity) or to converge for four algorithms including *Polaris*, Oort, Pisces and FedBuff. To rule out the noise incurred by randomness, each experiment was run 5 times with different random seeds in the sampler to ensure statistical significance. The results were reported in the figures, with a line indicating the mean and a shadow showing the 95% confidence interval. In addition, we analyzed the frequency of client involvement to gain insights into the root cause of the performance differences across all algorithms. This analysis was presented in the exploration rate and distribution plots.

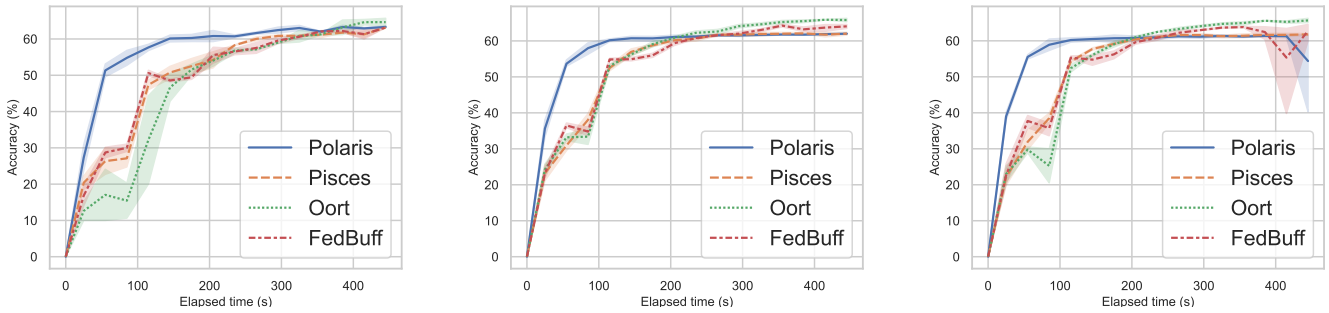
In Figures 4 and 5, we compare the performance of *Polaris* with that of the three other algorithms on the synthetic non-iid datasets and the real-world non-iid datasets. The blue line represents the performance of *Polaris*. Our results demonstrate that *Polaris* consistently achieves the best convergence performance across all datasets, with a faster convergence speed and a higher or similar accuracy (lower perplexity). Specifically, *Polaris* reaches the target accuracy or perplexity 1.5–2 times faster in terms of the wall-clock time when training various image classification models on Federated EMNIST, CINIC-10, CIFAR-10, CIFAR-100 datasets, as well as the next word prediction model BERT on the Tiny Shakespeare dataset.

Exploration vs. Exploitation. To investigate how the tradeoff between exploration and exploitation of sampling strategies affects asynchronous federated learning behavior, we recorded the explored clients as learning proceeds and plotted the exploration rate vs. wall-clock time in Figure 6. The x-axis represents the physical time, and the y-axis represents the percentage of the explored clients out of all 200 clients. Our analysis reveals that Oort and Pisces follow an ϵ-greedy exploration strategy, which reach out to new clients at the early stages of the learning process. FedBuff, on the other hand, performs uniform sampling, which results in a slower growth of the exploration rate compared to Pisces and Oort. However, we were surprised to find that *Polaris*, which outperformed all other methods, did not aggressively explore new clients. Instead, it took advantage of the explored clients and explored new clients occasionally, demonstrating a different approach to balancing exploration and exploitation. This finding challenges



(a) $\alpha = 0.5$ (b) $\alpha = 5$ (c) $\alpha = 10$

Fig. 8. *Polaris* shows advantages in training `ResNet-18` model on `CIFAR-10` datasets with different concentration factors (α).



(a) $\alpha = 0.5$ (b) $\alpha = 5$ (c) $\alpha = 10$

Fig. 9. *Polaris* shows advantages in training `VGG-16` model on `CINIC-10` datasets with different concentration factors (α).

previous assumptions about the importance of exploration, and rethinking its significance is necessary.

The distribution of client selection. To investigate how biased sampling strategies affect asynchronous federated learning behavior, we recorded the selection sets for all experiments and visualized the client selection in Figure 7. The x-axis represents all 200 clients, and the y-axis represents the accumulated sampling probability.

The baseline algorithm FedBuff, represented by the red line, uniformly samples clients and thus grows linearly. In contrast, the state-of-the-art biased asynchronous sampling method Pisces, represented by the orange line, exhibits a smooth upward curve, indicating its preference for clients with learning properties that accelerate global model convergence. Similarly, the state-of-the-art biased asynchronous sampling method Oort, represented by the green line, also shows a smoother upward curve, suggesting its preference for some clients with good learning properties. However, this bias is less pronounced than in Pisces.

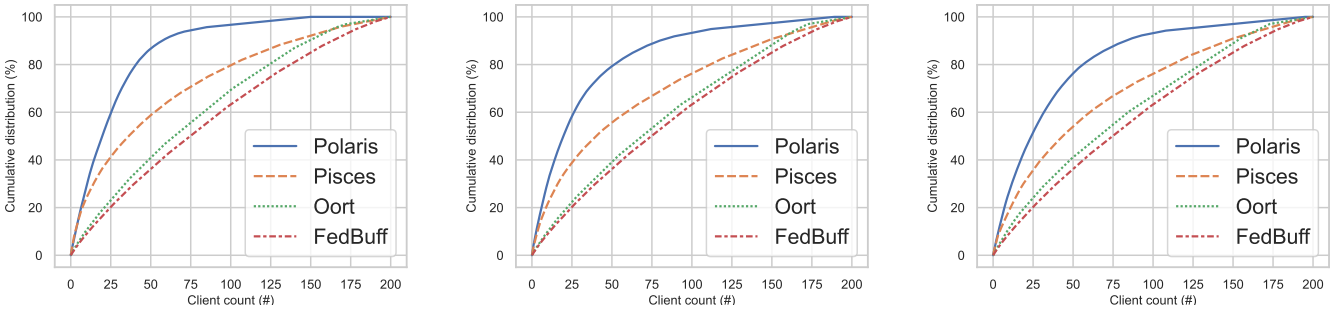
Our proposed algorithm *Polaris*, represented by the blue line, levels out towards the upper bound with a steep upward curve. This indicates that *Polaris* takes greater advantage of a subset of good clients, resulting in an improved learning behavior. These results confirm the practical benefits of biased sampling and highlight the effectiveness of our proposed algorithm.

***Polaris* as a plug-and-play module.** *Polaris* estimates the local data quality by analyzing the model updates from clients, which is a default part in clients reports. Unlike other estimators that rely on local loss calculations, *Polaris* does not require any additional information from clients beyond their local model updates. As a result, *Polaris* can be used as a plug-and-play module in asynchronous federated learning, and can be combined with various aggregation methods and local optimizations with minimal additional effort.

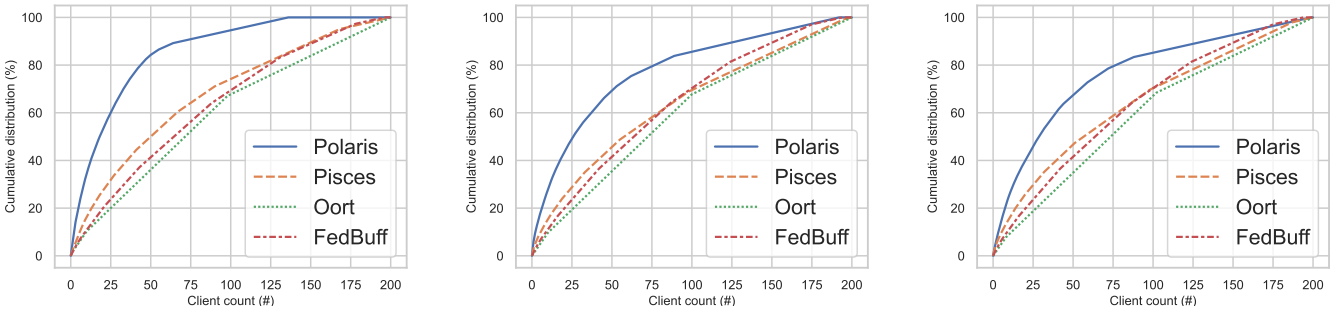
6.3 Ablation Studies

To conduct a comprehensive analysis of *Polaris*' effectiveness across various environments and configurations, we performed experiments for all four algorithms on datasets with varying levels of data heterogeneity (controlled by a concentration factor), smaller data partition sizes on local clients, and larger scales with a total of 2000 clients. In addition, we tested the sensitivity of the hyperparameter γ in *Polaris*.

Robustness against different levels of statistical heterogeneity. We conducted a series of experiments to evaluate the performance of *Polaris* on varying degrees of statistical heterogeneity. To control the heterogeneity levels in Dirichlet distribution, we applied concentration factors of 0.5, 5, and 10. These levels represent the extreme non-iid, mild non-iid, and approximating iid settings, respectively. Our results on `CIFAR-10` and `CINIC-10` datasets, as shown in Figure 8 and Figure 9, demonstrate that *Polaris* consistently

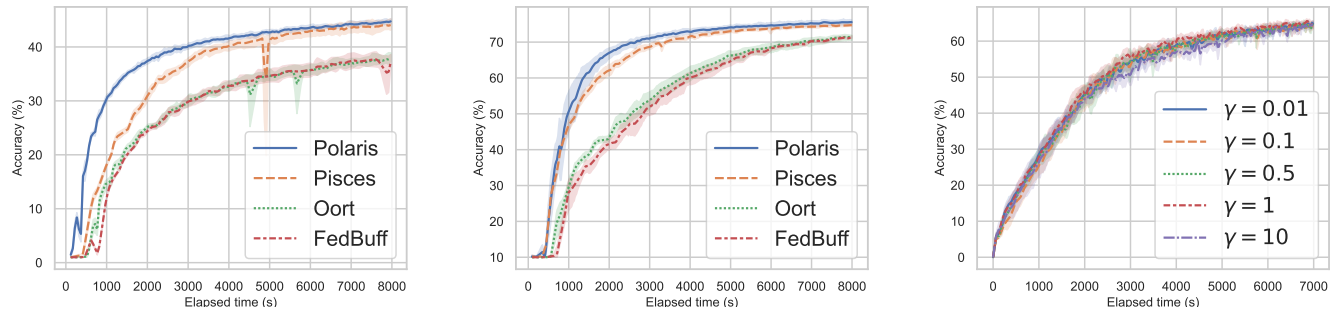


(a) CIFAR-10, $\alpha = 0.5$ (b) CIFAR-10, $\alpha = 5$ (c) CIFAR-10,$\alpha = 10$



(d) CINIC-10, $\alpha = 0.5$ (e) CINIC-10, $\alpha = 5$ (f) CINIC-10, $\alpha = 10$

Fig. 10. The Cumulative Distribution Function for Client Selection on CIFAR-10 and CINIC-10 with different Dirichlet concentration factors (α). *Polaris* does more biased sampling compared to Pisces, Oort and FedBuff. When data tend to be more non-iid (α becomes smaller), biased sampling strategy benefits.



(a) *Polaris* keeps advantages with smaller local data partitions CIFAR-100.

(b) *Polaris* is scalable when clients number increases to 2000 on CIFAR-10.

(c) *Polaris* is robust against various hyperparameter γ.

Fig. 11. *Polaris* is scalable and robust to setting changes and keeps advantages over Pisces, Oort and FedBuff.

outperforms the other algorithms in terms of faster convergence to the target accuracy on both datasets.

To further investigate the selected client distribution, we plot the results in Fig 10. From the figure, we can observe that *Polaris* still exhibits some biased sampling compared to other algorithms. As the degree of statistical heterogeneity increases, *i.e.* the concentration factor decreases, increasing the bias level leads to better sampling performance. Therefore, we conclude that *Polaris* is robust and effective in handling different levels of statistical heterogeneity.

Robustness against different local data sizes. We evaluated the performance of *Polaris* on a setting with smaller local data size by testing it on the CIFAR-100 dataset with a local size of 2000, instead of the original size of 3000. As shown in Figure 11a, while a smaller data partition can lead to a lower accuracy, *Polaris* still maintained its advantage over the comparison algorithms. This result demonstrates the robustness of *Polaris* in the scenarios where local data sizes may vary.

Scalability. We evaluate *Polaris* on a larger scale of federated learning setting, with a total of 2000 clients and 50 clients selected at each round. As before, when 50% of sam-

pled clients report, the central server performs aggregation and updates the global model. The result, shown in Figure 11b, demonstrates that *Polaris* maintains its superiority over FedBuff and Pisces even in the larger scale setting.

Hyperprameter sensitivity. We conducted experiments to evaluate the sensitivity of the hyperparameter γ in *Polaris*. The data samples used in `FEMNIST` are collected from real-world users, making it an extremely non-iid dataset. Varying the value of α from 0.01 to 10, we observe from Figure 11c that the performance of *Polaris* remains stable, and there is no need for fine-tuning the hyperparameter within the range. This property of *Polaris* is particularly advantageous as it can be applied to various settings without the need for tedious hyperparameter tuning. Therefore, we conclude that *Polaris* is a robust and practical algorithm for asynchronous federated learning.

7 Related Work

Due to limited communication bandwidth and device availability in large scale settings, federated learning algorithms usually sample a small subset of clients in each round. By default, the server selects clients uniformly at random since FedAvg [1] and FedBuff [3] are proposed. In practice, however, uniform sampling suffers from slow model convergence with respect to wall-clock time due to unpredictably high degrees of statistical and system heterogeneity.

To tackle the problem, multiple adaptive client sampling algorithms are presented. Mostly, researchers focus on improving the necessary communication round to converge with the methods that first evaluate clients' statistical properties, such as gradients [11]–[13] and losses [14], and select more "important" ones. While these methods reduce communication rounds, they do not efficiently improve learning performance with respect to wall-clock time.

To address the issue, extensive research efforts that prioritize clients with fast response times and high quality of data have been made. Among these methods, Oort is the leading one. To guide client selection, Oort ranks each client with a utility score that considers the client's response time and data quality and selects the top group. While it performs well in synchronous settings, it faces challenges in asynchronous scenarios, in which staleness exists.

Recently, Pisces [4] is proposed as a solution for asynchronous federated learning. It profiles clients statistical utility using a similar formula as Oort. Different from Oort, Pisces records the staleness instead of the response duration to evaluate speed utility, and exponentially punishes high stalenesses. Yet, it's limited as a heuristic and lacks theoretical support from an optimization perspective.

In this work, we systematically study the sampling problem in asynchronous federated learning and propose *Polaris* as an analytical solution from an optimization perspective. To efficiently navigate client sampling, *Polaris* calculates the optimal sampling strategy based on the reported staleness and model updates at each communication round. As learning proceeds, the server updates the sampling probability in each round and always performs the optimal sampling. Additionally, *Polaris* does not require clients to perform extra local calculation, which Oort and Pisces do. This benefit provides better privacy preservation and reduces computation burdon for remote devices, thus encouraging more potential clients to participate.

8 Concluding Remarks

In this study, we analyzed the sampling strategy in asynchronous federated learning from an optimization perspective, and proposed a theoretical sound solution, *Polaris* sampler. First, our analysis established the theoretical relationship between physical time and client sampling probability, accounting for statistical heterogeneity and staleness, and provided convergence bounds. Further, we formulated the sampling task as a geometric optimization problem. Additionally, our empirical analysis confirmed that model update is on a par with other loss based methods in estimating local data quality but does not require extra communication overhead.

Based on our findings, we proposed a novel client sampling algorithm named *Polaris*, which outperformed existing methods on multiple datasets, including `CIFAR-10`, `CIFAR-100`, `CINIC-10`, `Federated MNIST`, and `Tiny Shakespeare`. Our experiments demonstrated that the single hyperparameter γ is not sensitive within a considerably wide range and does not require extensive fine-tuning. Additionally, extensive ablation studies confirmed the scalability and robustness of *Polaris* against different local data partitions and data heterogeneities.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication efficient learning of deep networks from decentralized data," in *Proc. Int'l Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017.

[2] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proc. NeurIPS Workshop on Optimization for Machine Learning (OPT)*, 2019.

[3] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *Proc. Int'l Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2022, pp. 3581–3607.

[4] Z. Jiang, W. Wang, B. Li, and B. Li, "Pisces: Efficient federated learning via guided asynchronous training," in *Proc. of the 13th Symposium on Cloud Computing (SoCC)*, 2022, pp. 370–385.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[6] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *Proc. IEEE INFOCOM*, 2022.

[7] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *Proc. Int'l Conference on Learning Representations (ICLR)*, 2020.

[8] S. U. Stich, "Local SGD Converges Fast and Communicates Little," *Proc. Int'l Conference on Learning Representations (ICLR)*, 2019.

[9] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *Proc. Int'l Conference on Learning Representations (ICLR)*, 2021.

[10] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection." in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2021, pp. 19–35.

[11] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *Transactions on Machine Learning Research (TMLR)*, vol. 2022, no. 8, 2020.

[12] E. Rizk, S. Vlaski, and A. H. Sayed, "Federated learning under importance sampling," in *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[13] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, "Fast-convergent federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2021.
[14] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," in *Proc. Int'l Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2022.

Yufei Kang received her B.Engr. degree in Telecommunication Engineering in 2019 from Xidian University, China. Currently, She is working toward the doctoral degree in the iQua group at the University of Toronto. Her research interests include cloud computing, edge computing and federated learning.

Baochun Li received his B.Engr. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 1995 and his M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, in 1997 and 2000. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a Professor. He holds the Bell Canada Endowed Chair in Computer Engineering since August 2005. His current research interests include cloud computing, security and privacy, distributed machine learning, federated learning, and networking.

Dr. Li has co-authored more than 460 research papers, with a total of over 25000 citations, an H-index of 88 and an i10-index of 340, according to Google Scholar Citations. He was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems in 2000, the Multimedia Communications Best Paper Award from the IEEE Communications Society in 2009, the University of Toronto McLean Award in 2009, the Best Paper Award from IEEE INFOCOM in 2023, and the IEEE INFOCOM Achievement Award in 2024. He is a Fellow of the Canadian Academy of Engineering, a Fellow of the Engineering Institute of Canada, and a Fellow of IEEE.