Dynamic Staleness Control for Asynchronous Federated Learning in Decentralized Topology

Qianpiao Ma[1], Jianchun Liu[2], Qingmin Jia[1(✉)], Xiaomao Zhou[1], Yujiao Hu[1], and Renchao Xie[1,3]

[1] Purple Mountain Laboratories, Nanjing, Jiangsu, China
{maqianpiao,jiaqingmin,zhouxiaomao,huyujiao}@pmlabs.com.cn
[2] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China
jcliu17@ustc.edu.cn
[3] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
Renchao_xie@bupt.edu.cn

Abstract. Decentralized federated learning (DFL) has emerged as a promising paradigm for distributed machine learning over edge nodes (*i.e.*, workers) without relying on a centralized parameter server. Most existing DFL researches rely on synchronous communication among workers. However, due to edge heterogeneity and dynamic network conditions, synchronous DFL mechanisms may suffer from inefficient model training and poor scalability. Meanwhile, the existing asynchronous DFL (ADFL) mechanisms present the challenge of stale models among workers, leading to diminished training quality, especially on Non-IID data. In this paper, we propose a novel staleness-aware ADFL (SA-ADFL) mechanism, aiming to realize a trade-off between model training efficiency and quality by dynamic staleness control. Specifically, we provide rigorous theoretical proof for SA-ADFL and formulate the worker scheduling problem to minimize total model training time under flexible long-term staleness constraints. Then we decompose the original round-coupled problem into a series of single-round sub-problems by leveraging the Lyapunov optimization, enabling efficient worker selection to minimize training time in each round while ensuring staleness queue stability. Experimental results demonstrate that our SA-ADFL accelerates model training by approximately 52.9% while maintaining equivalent model training accuracy compared with the state-of-the-art mechanisms.

Keywords: Federated learning · Edge computing · Asynchronous · Decentralized · Dynamic

1 Introduction

With the increasing popularity of Internet of Things (IoT), a massive amount of data are generated from physical worlds every day [1]. Traditionally, these data

Z. Cai et al. (Eds.): WASA 2024, LNCS 14998, pp. 99–117, 2025.
https://doi.org/10.1007/978-3-031-71467-2_9

are transmitted to a remote cloud for processing, which will lead to potential privacy leakage and significant delay due to long-distance transmission. To this end, *edge computing* is proposed to push more computation capacity to the network edge, enabling efficient data processing locally. Besides, it motivates the application of federated learning (FL), which implements distributed machine learning over edge nodes (*i.e.*, workers) [2].

Based on the network topology, FL is generally categorized into two typical schemes. One is centralized FL (CFL) [2,3], where a centralized parameter server (PS) governs the model training among workers. Each worker performs local updates on its own data and uploads its local model to the parameter server. The parameter server then aggregates the local models into a global model and distributes it back to the workers. Although CFL is a well-established and widely used scheme, it has some drawbacks. First, the FL architecture is constrained by a star topology, which limits the scalability of the system. Second, as all workers have to communicate with the parameter server, it will become a communication bottleneck on suffering from the enormous amount of traffic workload, leading to the risk of single point failure. To overcome the limitation in CFL, decentralized federated learning (DFL) [4–9] is proposed, where workers exchange their local models with their neighbors through peer-to-peer communications. DFL is a promising scheme that can achieve high scalability, as it does not rely on a centralized parameter server. Moreover, DFL is more suitable for the emerging scenarios of IoT, where many intelligent and autonomous devices (*e.g.*, vehicles, robots) are deployed, which naturally form a decentralized network. However, the implementation of highly efficient DFL on mobile edge networks still faces some unique challenges:

- **Edge Heterogeneity:** In DFL, each worker needs to communicate with its heterogeneous neighbors with different locations, data sizes, computation capacities, and network quality [10]. This means that the coordination of the training process in DFL is more complicated and inefficient than that in CFL.
- **Edge Dynamic:** Due to the mobility of workers in edge computing, the connections between workers may be intermittent, creating the change of network topologies over time. Moreover, if a worker is under a poor communication condition, other workers may lose their connections with this worker.
- **Non-IID Data:** The data among workers is often non-independent-and-identically-distributed (Non-IID) [2] due to local data collection, which can significantly degrade FL performance. This challenge is more pronounced in DFL, since workers may have less data diversity than the parameter server in CFL, which can obtain models trained from all workers' data [11].

DFL can operate under two different communication schemes: synchronous or asynchronous. In the synchronous DFL (SDFL) scheme, each worker updates its local model on its own data and shares it with all its neighbors in each round. Then, it waits for the local models from all its neighbors and aggregates them to update its local model again. However, due to edge heterogeneity, it suffers from inefficient model training, as all workers have to wait for the slowest

or idle ones to finish their local updating and model transferring, known as the *straggler problem*. Moreover, it is vulnerable to the edge dynamics of the network, as the connections between workers may be unstable. Therefore, it is obviously impractical to expect all the neighbors to respond in each round, as some workers may lose their connectivity with others.

Therefore, it is necessary to relax the synchronization restrictions in DFL to cope with the heterogeneity and dynamic of edge network. Some researches propose the asynchronous DFL (ADFL) scheme, where each worker does not wait for the latest local models from all its neighbors to arrive. Instead, it aggregates the local models it has received so far and performs the next round of local updating as soon as it finishes the previous one. Asynchronous communication scheme is well suited for DFL scenarios because of the varying computation capacities and intermittent availability of workers. However, ADFL alleviates the straggler problem in SDFL on the expense of using out-of-date models, inevitably incurring *staleness* concern. Specifically, asynchronous communication scheme amplifies the negative effects of Non-IID data and leads to gradient divergence, as each worker aggregates the stale models from its neighbors, which may even make the model hard to converge [12].

Motivated by these challenges, we propose a dynamic staleness control mechanism for ADFL, aiming to strike a balance between model training efficiency and quality. Instead of performing local updating immediately after the last round as the existing ADFL researches, we control the local updating process for each worker to limit the staleness degree of the system. This allows us to exploit the advantages of asynchronous communication scheme to accelerate model training, while avoiding the drawbacks of low model quality caused by high-stale local models.

The main contributions of this paper are as follows:

- We formally describe the asynchronous federated learning in decentralized topology and analyze its convergence through rigorous theoretical proof. To achieve the dynacmic staleness control mechanism in ADFL, we formulate the worker scheduling problem with the objective of minimizing the total model training time under flexible long-term staleness constraints.
- We transform the original round-coupled problem into a series of single-round sub-problems by leveraging the Lyapunov optimization. Based on this, we determine the worker to perform updating in each round without compromising the staleness queue stability in the long run.
- Experimental results on the classical models and datasets show that, by deploying our mechanism, the model training can be greatly accelerated by about 52.9% while achieving the same accuracy compared with the state-of-the-art solutions.

2 Related Works

2.1 Communication Schemes of Decentralized Federated Learning

The communication schemes of DFL determine the behavior of the workers for performing local training and transferring models, which can be categorized into two schemes: synchronous and asynchronous (Table 1).

Synchronous communication schemes are adopted by most of the existing DFL researches. For example, BrainTorrent [4] uses a fully connected topology for DFL, where each worker shares its local model with all other workers in each round. GossipFL [5] creates a sparse topology, where each worker only shares its local model with its neighbors in each round, saving bandwidth costs. MATCHA [6] applies matching decomposition to divide the original network topology into separate subgraphs and communicates over different subgraphs in different rounds. L2PL [7] develops a learning-driven method to dynamically build an optimal partially connected topology in each training round. FedHP [8] adaptively controls local updating frequency and network topology to support the heterogeneous workers. D-Cliques [9] introduces a novel topology that lowers gradient bias by grouping workers in connected cliques. These synchronous communication schemes have a common limitation: they require each worker to wait for the completion of each round, known as the synchronization point, before the next round of model updating. This leads to slow convergence, as the system depends on the efficiency of the slowest worker.

There are a few of works adopt asynchronous communication schemes for DFL, where workers can send and receive models without any synchronization point or coordination with other participants. For example, HADFL [13] proposes a framework that supports decentralized asynchronous training on heterogeneous workers. AsyNG [11] dynamically selects neighbors for each worker to balance the communication cost and model performance in ADFL. EF-HC [14] introduces an event-triggered communication framework for ADFL to restrict

Table 1. Key Notations.

Symbol	Semantics
$\mathcal{V}$	The set of workers $\{v_1, v_2, ..., v_N\}$
$\mathcal{V}_i$	The set of neighbor workers of v_i including itself
D_i/D	The data size on worker v_i/all workers
α_i	The proportion of the data size of worker v_i to the total data size
σ_j^i	The proportion of the data size of neighbor worker v_j to the total data size of workers in $\mathcal{V}_i$
F_i/F	The local loss function of v_i/global loss function
$\mathbf{w}_i(t)$	The local model of worker v_i at round t
$\mathbf{w}_j^i(t)$	The local model from worker v_j in worker v_i's cache at round t
$\hat{\mathbf{w}}_i(t)$	The aggregated local model of worker v_i at round t
$\tau_j^i(t)$	The staleness of model $\mathbf{w}_j^i(t)$
$x_i(t)$	The indicator for whether worker v_i is determined to perform local updating at round t.

aggregations to only when new models are available. However, these asynchronous communication schemes do not have fine-grained control over staleness, which leads to high-stale models that produce excessive inconsistent updates and large biases from the current gradient direction.

2.2 Deal with Staleness in Asynchronous Federated Learning

As mentioned before, staleness is a major concern that affects the performance of asynchronous communication schemes in FL. Many researches have focused on mitigating the adverse effect of staleness in CFL scenarios. For example, FedAsync [15] assigns smaller aggregation weights to stale models to lessen their impact on training. SAFA [16] proposes a simple approach to handle staleness, where the parameter server discards too stale models during the training process. ASO-Fed [17] deploys dynamic learning rates for workers according to the frequency of their participating in global updating, which can also alleviate the staleness concern. The authors in [18,19] propose technological means to compensate for delayed gradients based on approximate Taylor expansion. FedSA [12] introduces a semi-asynchronous FL mechanism, which ensures the models not to be too stale by involving multiple workers in global updating in each round.

However, the above methods are all applicable to asynchronous CFL (ACFL) scenarios, and there is few research on effective staleness control in ADFL. In fact, staleness control in ADFL is more challenging and complicated than in ACFL, due to the more complex and dynamic network topology then the star topology in ACFL. In this paper, we fully consider the unique challenges faced by ADFL, and propose an effective dynamic staleness control method to improve model training performance.

3 Asynchronous Decentralized Federated Learning

3.1 Federated Learning (FL)

We perform federated learning over a set of workers $\mathcal{V} = \{v_1, v_2, ..., v_N\}$. Each worker v_i trains a model on its local dataset $\mathcal{D}_i$, with the size of $D_i \triangleq |\mathcal{D}_i|$. Then the loss function of worker v_i is defined as

$$F_i(\mathbf{w}) \triangleq \frac{1}{D_i} \sum_{\mathcal{D} \in \mathcal{D}_i} \mathcal{L}(\mathbf{w}; \mathcal{D}), \tag{1}$$

where $\mathbf{w}$ is the parameter vector and $\mathcal{L}(\cdot)$ is the loss function, *e.g.*, cross-entropy, focal or hinge loss.

The global dataset over all workers is $\mathcal{D}$, with size $D = \sum_{v_i \in \mathcal{V}} D_i$. Let $\alpha_i = D_i/D$ denote the proportion of worker v_i's data size to the total data size. Then the global loss function is defined as

$$F(\mathbf{w}) \triangleq \sum_{v_i \in \mathcal{V}} \frac{D_i}{D} F_i(\mathbf{w}) = \sum_{v_i \in \mathcal{V}} \alpha_i f_i(\mathbf{w}). \tag{2}$$

Algorithm 1 Asynchronous Decentralized Federated Learning (ADFL)

1: **for** $t = 1$ to T **do**
2: **Processing at Each Worker** v_i**:**
3: ***Logging Thread:***
4: **while** true **do**
5: **if** Receive local model $\mathbf{w}_j^i(t)$ from worker v_j **then**
6: Add $\mathbf{w}_j^i(t)$ to its cache
7: Calculate staleness $\tau_j^i(t)$
8: Calculate v_i's total staleness $\hat{\Omega}_i(t) = \sum_{v_j \in \mathcal{V}_i} \tau_j^i(t)$
9: Send REPORT message containing $\hat{\Omega}_i$ to the coordinator
10: ***Updating Thread:***
11: **if** Receive EXECUTE message **then**
12: Obtain $\mathbf{w}_j^i(t)$ of each worker $v_j \in \mathcal{V}_i$ from caches
13: Aggregation local models to obain $\hat{\mathbf{w}}_i(t)$ by Eq. (4)
14: Obtain local model $\mathbf{w}_i(t+1)$ by Eq. (5)
15: Send $\mathbf{w}_i(t+1)$ to the neighbouring workers
16: Send READY message to the coordinator
17: **Processing at the Coordinator:**
18: ***Logging Thread:***
19: **while** true **do**
20: **if** Receive REPORT message from worker v_i **then**
21: Update $\Omega_i(t) = \hat{\Omega}_i(t)$
22: **if** Receive READY message from worker v_i **then**
23: $\hat{\mathcal{V}} = \hat{\mathcal{V}} \cup \{v_i\}$
24: ***Scheduling Thread:***
25: Determine v_i to perform local updating from $\hat{\mathcal{V}}$
26: Send EXECUTE message to v_i
27: $\hat{\mathcal{V}} = \hat{\mathcal{V}} - \{v_i\}$
28: $\Omega_i(t+1) = 0$
29: **for each** $v_j \in \hat{\mathcal{V}}$ **do**
30: $\Omega_j(t+1) = \Omega_j(t) + |\mathcal{V}_j|$

The objective is to find the optimal parameter vector $\mathbf{w}^*$ so as to minimize $F(\mathbf{w})$, *i.e.*, $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} F(\mathbf{w})$.

3.2 Asynchronous Decentralized Federated Learning Architecture

We introduce the asynchronous decentralized federated learning (ADFL) architecture as described in Algorithm 1. In addition to a set of workers, the ADFL architecture includes a coordinator that schedules the local updating of workers for staleness control.

Each worker has a logging thread and an updating thread. Let $\mathbf{w}_i(t)$ denote the local model of worker v_i at round t. The logging thread (Lines 3–9) caches the local models sent by its neighbors, and reports its staleness to the coordinator. Note that if a worker receives multiple local models from the same neighbor before aggregating them, it only caches the latest one. Let $\mathbf{w}_j^i(t)$ denote the

local model from worker v_j in worker v_i's cache at round t. Note that since the workers perform model updating asynchronously, the received model $\mathbf{w}_j^i(t)$ in v_i's cache is not necessarily equal to the local model $\mathbf{w}_j(t)$ of v_j at round t [18]. Let $\tau_j^i(t)$ be the interval (called the *staleness* [15]) between the current round t, and the received model version from worker v_j at worker v_i's cache. The received model actually satisfies

$$\mathbf{w}_j^i(t) = \mathbf{w}_j(t - \tau_j^i(t)) \tag{3}$$

The logging thread of v_i also calculate its total staleness of received worker as $\hat{\Omega}_i(t) = \sum_{v_j \in \mathcal{V}_i} \tau_j^i(t)$, and then sends REPORT message containing $\hat{\Omega}_i$ to the coordinator. Let $\mathcal{V}_i$ denote the set of neighbor workers of v_i including itself. $x_i(t)$ is a boolean variable denoting whether worker v_i is determined to perform local updating at round t. Upon receiving an EXECUTE message from the coordinator at round t, *i.e.*, $x_i(t) = 1$, the updating thread (Lines 10–16) of worker v_i aggregates the cached local models by

$$\hat{\mathbf{w}}_i(t) = \frac{\sum_{v_j \in \mathcal{V}_i} D_j \mathbf{w}_j^i(t)}{\sum_{v_j \in \mathcal{V}_i} D_j} = \sum_{v_j \in \mathcal{V}_i} \sigma_j^i \mathbf{w}_j^i(t), \quad x_i(t) = 1, \tag{4}$$

where $\sigma_j^i = D_j / \sum_{v_j \in \mathcal{V}_i} D_j$ denote the proportion of the data size of worker v_j to the total data size of workers in $\mathcal{V}_i$. After the model aggregation, worker v_i performs local updating at round t by

$$\mathbf{w}_i(t+1) = \hat{\mathbf{w}}_i(t) - \eta \nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t)), \tag{5}$$

where η is the learning rate, ∇ is the gradient operator, $\xi_i(t)$ is a sample uniformly chosen from the local data on worker v_i at round t. Then worker v_i sends its local model $\mathbf{w}_i(t+1)$ to the neighboring workers and sends the READY message to the coordinator.

The coordinator has a logging thread and a scheduling thread. The logging thread (Lines 18–23) records the currently idle worker set $\hat{\mathcal{V}}$ and all workers' staleness in real time by the READY and REPORT messages. The scheduling thread (Lines 24–30) selects a worker from $\hat{\mathcal{V}}$ for the next local updating (see Sect. 5 for detail). If worker v_i is determined, the coordinator sends a EXECUTE message to v_i and remove v_i from set $\hat{\mathcal{V}}$. Accordingly, the total staleness $\Omega_i(t+1)$ of v_i in the next round $t+1$ is set to 0, and that of each other worker v_j in $\hat{\mathcal{V}}$ is increased by the number of their models in caches (*i.e.*, $\Omega_j(t+1) = \Omega_j(t) + |\mathcal{V}_j|$).

Note that the coordinator only exchanges messages (*i.e.*, REPORT, READY, and EXECUTE) containing minimal data with workers, which is an essential difference from the parameter server in CFL. As a result, we disregard its transmission time, which is negligible when compared to the model training and model transfer times. Moreover, in the event that a worker loses its connection with the coordinator due to dynamic network conditions, the coordinator excludes this worker from scheduling, ensuring that the overall training process of the system continues.

4 Convergence Analysis

4.1 Assumptions

We make the following widely used assumptions on the loss functions $F_i(\mathbf{w}), \forall v_i \in \mathcal{V}$ [20,21].

Assumption 1. $F_i(\mathbf{w})$ is L-smooth, *i.e.*, $\forall \mathbf{w}_1, \mathbf{w}_2$, $F_i(\mathbf{w}_2) - F_i(\mathbf{w}_1) \leq \langle \nabla F_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{L}{2}\|\mathbf{w}_2 - \mathbf{w}_1\|^2$.

Assumption 2. $F_i(\mathbf{w})$ is μ-strongly convex, *i.e.*, $\forall \mathbf{w}_1, \mathbf{w}_2$, $F_i(\mathbf{w}_2) - F_i(\mathbf{w}_1) \geq \langle \nabla F_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{\mu}{2}\|\mathbf{w}_2 - \mathbf{w}_1\|^2$.

4.2 Analysis of Convergence Bounds

Convergence of a Single Round of Local Updating. We first give the following lemma to describe the convergence of a single round of local updating. For ease of expression, we abbreviate $F(\mathbf{w}^*)$ as F^*, which represents the optimal value of the global loss function F. Similarly, $F_i^*(\mathbf{w}_i^*)$ is abbreviated as F_i^*, which denotes the optimal value of the loss function F_i on worker v_i.

Lemma 1. *Taking* $\eta < \frac{\mu}{2L^2}$, *by the local updating of Eq.* (5), *it holds that*

$$\mathbb{E}[F(\mathbf{w}_i(t+1))] - F^* \leq \rho(\mathbb{E}[F(\hat{\mathbf{w}}_i(t))] - F^*) + \delta_i,$$

where $\rho = 1 - \mu\eta$, *and* $\delta_i = \frac{\eta}{2}\xi_i + L\eta^2 g_i^*$.

Proof. According to Assumption 1, it is obvious that F is L-smooth. Combining with Eq. (5), it follows

$$\begin{aligned} F(\mathbf{w}_i(t+1)) - F^* &\leq F(\hat{\mathbf{w}}_i(t)) - F^* + \langle \nabla F(\hat{\mathbf{w}}_i(t)), \mathbf{w}_i(t+1) - \hat{\mathbf{w}}_i(t) \rangle \\ &\quad + \frac{L}{2}\|\mathbf{w}_i(t+1) - \hat{\mathbf{w}}_i(t)\|^2 \\ &= F(\mathbf{w}_i(t+1)) - F^* - \eta \langle \nabla F(\hat{\mathbf{w}}_i(t)), \nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t)) \rangle \\ &\quad + \frac{L\eta^2}{2}\|\nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t))\|^2. \end{aligned} \tag{6}$$

We derive the expectation of the third term of Eq. (6) as

$$\begin{aligned} &\mathbb{E}[\langle \nabla F(\hat{\mathbf{w}}_i(t)), \nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t)) \rangle] = \langle \nabla F(\hat{\mathbf{w}}_i(t)), \nabla F_i(\hat{\mathbf{w}}_i(t)) \rangle \\ =& \frac{1}{2}\left(\|\nabla F(\hat{\mathbf{w}}_i(t))\|^2 + \|\nabla F_i(\hat{\mathbf{w}}_i(t))\|^2 - \|\nabla F(\hat{\mathbf{w}}_i(t)) - \nabla F_i(\hat{\mathbf{w}}_i(t))\|^2 \right) \\ \geq& \frac{1}{2}\left(\|\nabla F(\hat{\mathbf{w}}_i(t))\|^2 + \|\nabla F_i(\hat{\mathbf{w}}_i(t))\|^2 - \xi_i^2 \right), \end{aligned} \tag{7}$$

where $\xi_i \triangleq \max \|\nabla F(\mathbf{w}) - \nabla F_i(\mathbf{w})\|$ is the maximum of the gradient divergences for $\forall \mathbf{w}$ according to [20]. By Assumption 2, it is obvious that F is μ-strongly convex. It follows

$$\|\nabla F(\hat{\mathbf{w}}_i(t))\|^2 \geq 2\mu(F(\hat{\mathbf{w}}_i(t)) - F^*). \tag{8}$$

By using the AM-GM Inequality, we derive the expectation of the last term of Eq. (6) as

$$\begin{aligned}
&\mathbb{E}[\|\nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t))\|^2] \\
&\quad = \mathbb{E}[\|\nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t)) - \nabla F_i(\mathbf{w}_i^*; \xi_i(t)) + \nabla F_i(\mathbf{w}_i^*; \xi_i(t))\|^2] \\
&\quad \leq 2\mathbb{E}[\|\nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t)) - \nabla F_i(\mathbf{w}_i^*; \xi_i(t))\|^2] + 2\mathbb{E}[\|\nabla F_i(\mathbf{w}_i^*; \xi_i(t))\|^2],
\end{aligned} \tag{9}$$

where $\mathbf{w}_i^*$ is the optimal solution of F_i. According to Lemma 3 of [22], we have

$$\mathbb{E}[\|\nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t)) - \nabla F_i(\mathbf{w}_i^*; \xi_i(t))\|^2] \leq 2L(F_i(\hat{\mathbf{w}}_i(t)) - F_i^*). \tag{10}$$

Since F_i is μ-strongly convex for each $\forall v_i \in \mathcal{V}$, we have

$$F_i(\hat{\mathbf{w}}_i(t)) - F_i^* \leq \frac{1}{2\mu}\|\nabla F_i(\hat{\mathbf{w}}_i(t))\|^2. \tag{11}$$

By taking Eqs. (10) and (11) into Eq. (9), we have

$$\mathbb{E}[\|\nabla F_i(\hat{\mathbf{w}}_i(t); \xi_i(t))\|^2] \leq \frac{2L}{\mu}\|\nabla F_i(\hat{\mathbf{w}}_i(t))\|^2 + 2g_i^*. \tag{12}$$

where $g_i^* \triangleq \mathbb{E}[\|\nabla F_i(\mathbf{w}_i^*; \xi_i(t))\|^2]$ according to [22]. By taking Eqs. (7), (8) and (12) into Eq. (6), we have

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_i(t+1))] - F^* \\
&\quad \leq (1 - \mu\eta)(\mathbb{E}[F(\hat{\mathbf{w}}_i(t))] - F^*) - (\frac{\eta}{2} - \frac{L^2\eta^2}{\mu})\|\nabla F_i(\hat{\mathbf{w}}_i(t))\|^2 + \frac{\eta}{2}\xi_i^2 + L\eta^2 g_i^*.
\end{aligned} \tag{13}$$

Since $\eta < \frac{\mu}{2L^2}$, we have

$$\mathbb{E}[F(\mathbf{w}_i(t+1))] - F^* \leq \rho(\mathbb{E}[F(\hat{\mathbf{w}}_i(t))] - F^*) + \delta_i, \tag{14}$$

where $\rho = 1 - \mu\eta$ and $\delta_i = \frac{\eta}{2}\xi_i^2 + L\eta^2 g_i^*$.

A Key Lemma for Analysis. For ease of expression, we construct three sequences of matrices $\mathbf{X}(t)$, $\mathbf{Y}_j(t)$ and $\mathbf{Z}(t)$ for $t \geq 1$, where $\mathbf{X}(t)$ and $\mathbf{Y}_j(t)$ are $N \times N$ diagonal matrices, and $\mathbf{Z}(t)$ is a M-dimensional vector. Specifically, the $i(t)$-th diagonal element of $\mathbf{X}(t)$ is $x(t)$, and others are 1; the $i(t)$-th diagonal element of $\mathbf{Y}_j(t)$ is $y_j(t)$, and others are 0; the $i(t)$-th element of $\mathbf{Z}(t)$ is $z(t)$, and others are 0. Note that $x(t)$, $y_j(t)$ and $z(t)$ are are on the same row in $\mathbf{X}(t)$, $\mathbf{Y}_j(t)$ and $\mathbf{Z}(t)$, respectively, satisfying $\theta(t) = x(t) + \sum_{v_j \in V_i} y_j(t) \leq 1$. We first state a key lemma for our statement. Note that if $x_i(t) = 1$, we can drop the index i in $\tau_j^i(t)$, *i.e.*, rewrite as $\tau_j(t)$, since the worker v_i that performs the update in round t is determined. Let $\omega_j(t) = t - \tau_j(t) - 1 \geq 0$, $\tau_{max} = \max_{t,j}\{\tau_j(t)\}$ and $\theta_{max} = \max_t\{\theta(t)\}$.

Lemma 2. *Let* $\mathbf{Q}(t)$ *be a sequence of matrices for* $t \geq 0$. *If* $\mathbf{Q}(t) \leq \mathbf{X}(t)\mathbf{Q}(t-1) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(t)\mathbf{Q}(\omega_j(t)) + \mathbf{Z}(t)$, *where* $x_i(t) = 1$, *then we have*

$$\mathbf{Q}(T) \leq \prod_{t=0}^{T} \mathbf{P}(t)\mathbf{Q}(0) + \sum_{t=0}^{T} \mathbf{\Delta}(t),$$

where $\mathbf{P}(t)$ *is a* $M \times M$ *diagonal matrix and its* $i(t)$*-th diagonal element is* $\theta_{max}^{\frac{1}{1+\tau_{max}}}$, *and others are 1. That is,*

$$\mathbf{P}(t) = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \theta_{max}^{\frac{1}{1+\tau_{max}}} & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix},$$

$\mathbf{\Delta}(r)$ *is a* M*-dimensional vector*

$$\mathbf{\Delta}(t) = \begin{cases} [0, 0, ..., 0]^\top, t = 0 \\ (\mathbf{X}(t) + \sum\limits_{v_j \in \mathcal{V}_i} \mathbf{Y}(t) - \mathbf{E}) \sum\limits_{r=0}^{t-1} \mathbf{\Delta}(r) + \mathbf{Z}(t), t \geq 1. \end{cases}$$

Proof. Since $\theta_{max} < 0$, $\theta_{max}^{-\frac{\tau_{max}}{1+\tau_{max}}} > 1$. It follows that

$$\begin{aligned} x(t) + \sum_{v_j \in \mathcal{V}_i} y_j(t)\theta_{max}^{-\frac{\tau_{max}}{1+\tau_{max}}} \leq & (x(t) + \sum_{v_j \in \mathcal{V}_i} y_j(t))\theta_{max}^{-\frac{\tau_{max}}{1+\tau_{max}}} \\ \leq & \theta_{max} \cdot \theta_{max}^{-\frac{\tau_{max}}{1+\tau_{max}}} = \theta_{max}^{\frac{1}{1+\tau_{max}}} \end{aligned} \tag{15}$$

By using Eq. (15), we can derive the following relation:

$$\begin{aligned} & \mathbf{X}(t) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(t) \prod_{r=t-\tau_j(t)}^{t-1} [\mathbf{P}(r)]^{-1} \\ \leq & \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & x(t) + \sum_{v_j \in \mathcal{V}_i} y_j(t)\theta_{max}^{-\frac{\tau_{max}}{1+\tau_{max}}} & & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \theta_{max}^{\frac{1}{1+\tau_{max}}} & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix} = \mathbf{P}(t) \end{aligned} \tag{16}$$

It is obvious that Lemma 2 is true when $t = 0$. We assume that the induction hypothesis holds for all t from 0 to $T-1$, *i.e.*,

$$\mathbf{Q}(t) \leq \prod_{r=0}^{t} \mathbf{P}(r)\mathbf{Q}(0) + \sum_{r=0}^{t} \mathbf{\Delta}(r), \quad \forall t \in \{0, 1, ...T-1\}. \tag{17}$$

When $t = T$, we derive that

$$\begin{aligned}
&\mathbf{Q}(T) \leq \mathbf{X}(T)\mathbf{Q}(T-1) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(T)\mathbf{Q}(\omega_j(T)) + \mathbf{Z}(T) \\
\leq &\mathbf{X}(T)\left[\prod_{t=0}^{T-1} \mathbf{P}(t)\mathbf{Q}(0) + \sum_{t=0}^{T-1} \mathbf{\Delta}(t)\right] + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(T)\left[\prod_{t=0}^{\omega_j(T)} \mathbf{P}(t)\mathbf{Q}(0) + \sum_{t=0}^{\omega_j(T)} \mathbf{\Delta}(t)\right] + \mathbf{Z}(T) \\
= &\left[\mathbf{X}(T) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(T) \prod_{t=T-\tau_j(T)}^{T-1} [\mathbf{P}(t)]^{-1}\right] \prod_{t=0}^{T-1} \mathbf{P}(t)\mathbf{Q}(0) \\
&+ \left[\mathbf{X}(T) \sum_{t=0}^{T-1} \mathbf{\Delta}(t) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(T) \sum_{t=0}^{\omega_j(T)} \mathbf{\Delta}(t)\right] + \mathbf{Z}(T) \\
\leq &\mathbf{P}(T) \prod_{t=0}^{T-1} \mathbf{P}(t)\mathbf{Q}(0) + \left[\mathbf{X}(T) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(T)\right] \sum_{t=0}^{T-1} \mathbf{\Delta}(t) + \mathbf{Z}(T) \\
= &\prod_{t=0}^{T} \mathbf{P}(t)\mathbf{Q}(0) + \sum_{t=0}^{T} \mathbf{\Delta}(t). \qquad (18)
\end{aligned}$$

Thus, we complete the induction and prove the correctness of Lemma 2.

Convergence Bound of SA-ADFL. Combining with Lemma 1 and Lemma 2, we analyze the convergence bound of SA-ADFL as the following theorem. For ease of expression, we denote $\mathbf{A} = [\alpha_1, ...\alpha_N]$ as the data size weight vector of all workers, and denote

$$\mathbf{S} = \begin{pmatrix} \sigma_1^1 & \cdots & \sigma_1^N \\ \vdots & \ddots & \vdots \\ \sigma_N^1 & \cdots & \sigma_N^N \end{pmatrix} \qquad (19)$$

as the data size weight matrix of the neighboring workers. Since no global model is maintained in the decentralized topology, we consider the weighted $\overline{\mathbf{w}}_T = \sum_{v_i \in \mathcal{V}} \alpha_i \mathbf{w}_T^i$ of all local models.

Theorem 1. *$\mathbf{w}_0$ is the initial model on each worker. After inter-cluster aggregation Eq. (4) is performed T times, the weighted model $\overline{\mathbf{w}}_T$ satisfies*

$$\mathbb{E}[F(\overline{\mathbf{w}}_T)] - F^* \leq \kappa(F(\mathbf{w}_0) - F^*) + \delta,$$

where $\kappa = \sum_{v_i \in \mathcal{V}} \alpha_i \rho^{\frac{\psi_i T}{1+\tau_{max}}}$, $\rho = 1 - \mu\eta$, $\delta = \mathbf{A} \odot \mathbf{LS\Delta}$, $\mathbf{L} = [\frac{1-\rho^{\psi_1 T}}{1-\rho}, \frac{1-\rho^{\psi_2 T}}{1-\rho}, ..., \frac{1-\rho^{\psi_N T}}{1-\rho}]$ and $\mathbf{\Delta} = [\delta_1, \delta_2, ...\delta_N]^\top$. $\odot$ is the Hadamard product symbol.

Proof. Since F is convex and $\sigma_j^i \in (0,1)$, combining with Eq. (3), for $\forall v_i \in \mathcal{V}$, it holds that

$$\begin{aligned} F(\hat{\mathbf{w}}_i(t)) - F^* &\leq \sum_{v_j \in \mathcal{V}_i} \sigma_j^i F(\hat{\mathbf{w}}_i(t)) - F^* = \sum_{v_j \in \mathcal{V}_i} \sigma_j^i (F(\mathbf{w}_j^i(t)) - F^*) \\ &= \sum_{v_j \in \mathcal{V}_i} \sigma_j^i (F(\mathbf{w}_j(t - \tau_j^i(t))) - F^*). \end{aligned} \tag{20}$$

According to Lemma 1, we derive that

$$\mathbb{E}[F(\hat{\mathbf{w}}_i(t))] - F^* \leq \rho \sum_{v_j \in \mathcal{V}_i} \sigma_j^i (\mathbb{E}[F(\hat{\mathbf{w}}_j(t - \tau_j^i(t) - 1))] - F^*) + \sum_{v_j \in \mathcal{V}_i} \sigma_j^i \delta_j. \tag{21}$$

Let $Q_i(t) = F(\hat{\mathbf{w}}_i(t)) - F^*$ and $\mathbf{Q}(t) = [Q_1(t), ..., Q_M(t)]^\top$. The recursive relation for $\forall v_i \in \mathcal{V}$ is transformed into

$$\mathbf{Q}(t) \leq \mathbf{X}(t)\mathbf{Q}(t-1) + \sum_{v_j \in \mathcal{V}_i} \mathbf{Y}_j(t)\mathbf{Q}(\omega_j(t)) + \mathbf{Z}(t), \tag{22}$$

where

$$\mathbf{X}(t) = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \rho\sigma_i^i & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix}, \mathbf{Y}_j(t) = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \rho\sigma_j^i & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 0. \end{pmatrix},$$

and $\mathbf{Z}(t) = [0, 0, ... \sum_{v_j \in \mathcal{V}_i} \sigma_j^i \delta_j, ..., 0]^\top$. According to Lemma 2,

$$\mathbf{Q}(T) \leq \prod_{t=0}^{T} \mathbf{P}(t)\mathbf{Q}(0) + \sum_{t=0}^{T} \mathbf{\Delta}(t), \tag{23}$$

where $\mathbf{P}(t)$ is a $M \times M$ diagonal matrix. Specifically, the $i(t)$-th diagonal element of $\mathbf{P}(t)$ is $(\rho \sum_{v_j \in \mathcal{V}_i} \sigma_i^j)^{\frac{1}{1+\tau_{max}}} = \rho^{\frac{1}{1+\tau_{max}}}$, and others are 1. $\mathbf{\Delta}(t)$ satisfies the following recursive relation:

$$\mathbf{\Delta}(t) = \begin{cases} [0, 0, ..., 0]^\top, t = 0 \\ \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \rho & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 0 \end{pmatrix} \sum_{r=0}^{t-1} \mathbf{\Delta}(r) + \begin{pmatrix} 0 \\ \vdots \\ \sum\limits_{v_j \in \mathcal{V}_i} \sigma_j^i \delta_j \\ \vdots \\ 0 \end{pmatrix}, t \geq 1. \end{cases} \tag{24}$$

From Eq. (24), we can derive that

$$\mathbf{A}\sum_{t=0}^{T}\mathbf{\Delta}(t) = [\alpha_1, ..., \alpha_N]\begin{pmatrix} \frac{1-\rho^{\psi_1 T}}{1-\rho}\sum_{v_j\in\mathcal{V}_1}\sigma_j^1\delta_j \\ \frac{1-\rho^{\psi_2 T}}{1-\rho}\sum_{v_j\in\mathcal{V}_2}\sigma_j^2\delta_j \\ \vdots \\ \frac{1-\rho^{\psi_N T}}{1-\rho}\sum_{v_j\in\mathcal{V}_M}\sigma_j^N\delta_j \end{pmatrix}$$
$$= \sum_{v_i\in\mathcal{V}}\alpha_i\frac{1-\rho^{\psi_i T}}{1-\rho}\sum_{v_j\in\mathcal{V}_i}\sigma_j^i\delta_j = \mathbf{A}\odot\mathbf{LS\Delta}, \tag{25}$$

$\mathbf{L} = [\frac{1-\rho^{\psi_1 T}}{1-\rho}, \frac{1-\rho^{\psi_2 T}}{1-\rho}, ..., \frac{1-\rho^{\psi_N T}}{1-\rho}]$ and $\mathbf{\Delta} = [\delta_1, \delta_2, ...\delta_N]^\top$. Therefore, we have

$$\mathbb{E}[F(\overline{\mathbf{w}}_T)] - F^* \le \sum_{v_i\in\mathcal{V}}\alpha_i(\mathbb{E}[F(\hat{\mathbf{w}}_i(T))] - F^*) = \sum_{v_i\in\mathcal{V}}\alpha_i Q_i(T) = \mathbf{AQ}(T)$$
$$\le \mathbf{A}\prod_{t=0}^{T}\mathbf{P}(t)\mathbf{1}_M(F(\mathbf{w}_0) - F^*) + \mathbf{A}\sum_{t=0}^{T}\mathbf{\Delta}(t)$$
$$= \sum_{v_i\in\mathcal{V}}\alpha_i\rho^{\frac{\psi_i T}{1+\tau_{max}}}(F(\mathbf{w}_0) - F^*) + \mathbf{A}\odot\mathbf{LS\Delta}. \tag{26}$$

Setting $\kappa = \sum_{v_i\in\mathcal{V}}\alpha_i\rho^{\frac{\psi_i T}{1+\tau_{max}}}$ and $\delta = \mathbf{A}\odot\mathbf{LS\Delta}$, we complete the proof.

5 Staleness-Aware ADFL (SA-ADFL)

5.1 Problem Formulation

In each round, the coordinator selects a worker in temp set $\hat{\mathcal{V}}$ to execute the local updating. Recall that $x_i(t)$ is the selection indicator for worker v_i at round t, where $x_i(t) = 1$ signifies v_i is selected and $x_i(t) = 0$ otherwise. The arrival time of worker v_i's READY message at round t is represented by $R_i(t)$. The training time of a single round on v_i is given by $A_i = a_i D_i$, where a_i is the completion time needed by v_i to train one data sample. We denote the completion time of round t as $H(t)$. Since each worker performs local updating asynchronously, $H(t)$ satisfies the following recurrence relation: $H(t+1) = \max\{R_i(t) + A_i - H(t), 0\}$. Let $\boldsymbol{x} = \{x_i(t)|v_i \in \mathcal{V}, t \in [1, T]\}$ denote the worker scheduling strategy in the whole training process, the problem is formulated as follows:

$$\textbf{(P1)}: \min_{\boldsymbol{x}} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[H(t)] \tag{27a}$$
$$\textbf{s.t.} \quad \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\Omega_i(t)] \le \overline{\Omega}, \qquad v_i \in \mathcal{V} \tag{27b}$$
$$\Omega_i(t) \le \Omega_{max}, \qquad v_i \in \mathcal{V}, t \in [1, T] \tag{27c}$$
$$\sum_{v_i\in\mathcal{V}} x_i(t) = 1, \qquad t \in [1, T] \tag{27d}$$
$$x_i(t) \in \{0, 1\}, \qquad \forall v_i \in \mathcal{V}, t \in [1, T]. \tag{27e}$$

Constraint (27c) is the long-term staleness constraint for each worker. Constraint (27d) requires that all workers' staleness do not exceed an upper limit Ω_{max} at each round. Constraint (27e) represents that a worker is selected to perform local updating at each round. The objective is to determine strategy $\boldsymbol{x}$ so as to minimize the total training time while adhering to both long-term and individual staleness constraints.

5.2 Lyapunov Optimization Framework

To enable online scheduling without any future information while satisfying the total staleness constraints of workers, we construct a virtual queue $Q_i(t)$ for each worker v_i to indicate the gap between the cumulative staleness till round t and the given budget $\overline{\Omega}$, evolved as

$$Q_i(t+1) = \max\{Q_i(t) + \Omega_i(t) - \overline{\Omega}, 0\}. \tag{28}$$

To characterize the stability of the training procedure, we first define the Lyapunov function as $L(\Theta(t)) = \frac{1}{2}\sum_{i=1}^{N} Q_i(t)^2$, where $\Theta(t) = \{Q_i(t)\}_{v_i \in \mathcal{V}}$ contains the backlogs of all the virtual queues. To keep the virtual queues stable (*i.e.*, to enforce the staleness constraints) by persistently pushing the Lyapunov function towards a lower value, we introduce one-round Lyapunov drift as

$$\begin{aligned}
\Delta L(\Theta(t)) &\triangleq \mathbb{E}[L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)] = \frac{1}{2}\sum_{i=1}^{N}\mathbb{E}[Q_i(t+1)^2 - Q_i(t)^2|\Theta(t)] \\
&= \frac{1}{2}\sum_{i=1}^{N}\mathbb{E}[(Q_i(t) + \Omega_i(t) - \overline{\Omega})^2 - Q_i(t)^2|\Theta(t)] \\
&= \sum_{i=1}^{N}\mathbb{E}[(\Omega_i(t) - \overline{\Omega})|\Theta(t)]Q_i(t) + \frac{1}{2}\sum_{i=1}^{N}(\Omega_i(t) - \overline{\Omega})^2.
\end{aligned} \tag{29}$$

Through employing Lyapunov optimization theory, our objective is to minimize a supremum bound on the following drift-plus-penalty expression in each training round:

$$\Delta L(\Theta(t)) + V\mathbb{E}[H(t)|\Theta(t)] \leq V\mathbb{E}[H(t)|\Theta(t)] + \sum_{i=1}^{N}\mathbb{E}[(\Omega_i(t) - \overline{\Omega})|\Theta(t)]Q_i(t) + \Gamma, \tag{30}$$

where $\Gamma = \frac{1}{2}\sum_{i=1}^{N}(\Omega_{max} - \overline{\Omega})^2$ is a constant, and the parameter $V \geq 0$ controls the time-staleness deficit tradeoff, *i.e.*, how much we shall emphasize the training time minimization compared to the staleness deficit.

Then the original round-coupling optimization problem **P1** can be further decomposed into the following step-by-step online scheduling problem:

$$\textbf{(P2)}: \min_{x_i(t),\forall v_i\in\mathcal{V}} \sum_{i=1}^{N}(\Omega_i(t)-\overline{\Omega})Q_i(t)+VH(t) \tag{31a}$$

$$\textbf{s.t.} \quad \Omega_i(t)\le\Omega_{max}, \qquad v_i\in\mathcal{V} \tag{31b}$$

$$\sum_{v_i\in\mathcal{V}} x_i(t)=1, \tag{31c}$$

$$x_i(t)\in\{0,1\}, \qquad \forall v_i\in\mathcal{V}. \tag{31d}$$

We can solve **P2** to obtain a worker from $\hat{\mathcal{V}}$ to perform local updating at each round t (Line 25 in Algorithm 1).

5.3 Performance Analysis

Theorem 2. *Following the strategy obtained by Algorithm 1, the long-term system training time satisfies:*

$$\sum_{t=0}^{T-1}\mathbb{E}[H(t)]<H^*+\frac{T\Gamma}{V}$$

where $H^*=\sum_{t=0}^{T-1}H^*(t)$ *is the minimum training completion time obtained by the optimal strategy to* ***P1****.*

Proof. From Theorem 4.5 in [23], for an arbitrary $\delta>0$, there exists a stationary and randomized strategy for **P2**, which decides independent of the current queue backlogs, such that the inequalities $\sum_{t=0}^{T-1}\mathbb{E}[H(t)]\le H^*+\delta$ and $\mathbb{E}[\Omega_i(t)-\overline{\Omega}]\le\delta$ are satisfied. From Eq. (30), we have

$$\Delta L(\Theta(t))+V\mathbb{E}[H^*(t)|\Theta(t)]\le V\mathbb{E}[H(t)|\Theta(t)]+\sum_{i=1}^{N}\mathbb{E}[(\Omega_i(t)-\overline{\Omega})|\Theta(t)]Q_i(t)+\Gamma \tag{32}$$

Summing the inequality over $t\in\{0,1,...,T-1\}$, we have:

$$\mathbb{E}[L(\Theta(t))-L(\Theta(0))]+V\sum_{t=0}^{T-1}\mathbb{E}[H^*(t)]\le V(H^*+\delta)+\delta T\sum_{i=1}^{N}Q_i(t)+T\Gamma \tag{33}$$

By letting δ go to zero and considering the fact that $L(\Theta(t))\ge 0$ and $L(\Theta(0))=0$, we derive that

$$\sum_{t=0}^{T-1}\mathbb{E}[H(t)]<H^*+\frac{T\Gamma}{V} \tag{34}$$

Theorem 2 shows that the strategy obtained by Algorithm 1 achieves $O(\frac{1}{V})$-near of the optimal strategies without compromising the staleness queue stability in the long run.

6 Performance Evaluation

6.1 System Setup

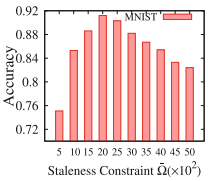
Environment. We implement a large-scale decentralized federated learning system using PySyft [24], a Python library for privacy-preserving deep learning under the PyTorch framework. PySyft allows the virtual worker creation for FL training, where each worker simulates an individual machine and trains a local model on its own dataset. We simulate a typical edge computing system with 100 workers randomly deployed in a 100 m×100 m region. The mobility patterns of edge nodes are modeled similar to [25] and the communication environment is setting according to [21].

Models and Datasets. We train classical CNN [26] models on two datasets (MNIST [27] and CIFAR-10 [28]). MNIST consists of 60,000 handwritten digits for training and 10,000 for testing, while CIFAR-10 includes 50,000 images for training and 10,000 for testing, and both of them have ten different types of objects. The CNN network architecture consists of two 5×5 convolution layers (20, 50 channels for MNIST and 32, 64 channels for CIFAR-10), each of which is followed by 2×2 max pooling, two fully-connected layers (800, 500 units for MNIST and 1600, 512 units for CIFAR-10), and a softmax layer with 10 units. The model sizes are 1.64MB (CNN on MNIST) and 3.35MB (CNN on CIFAR-10), respectively. Similar to the existing works [12], we implement the Non-IID data among workers by label skewed partition. Specifically, the data in MNIST (or CIFAR-10) labeled as '0' are distributed to workers v_1-v_{10}, the data labeled as '1' are distributed to workers v_{11}-v_{20},..., and the data labeled as '9' are distributed to workers v_{91}-v_{100}.

Benchmarks. To demonstrate the advantages of asynchronous decentralized federated learning mechanism with staleness control, we compare our SA-ADFL mechanism with two typical DFL benchmarks. 1) FedHP [8]: A typical synchronous decentralized federated learning mechanism. 2) HADFL [13]: A asynchronous decentralized federated learning mechanism without staleness control.

6.2 Evaluation Results

The Impact of Staleness Constraint. Figure 1 illustrates the impact of staleness constraint $\overline{\Omega}$ on accuracy after 5000 s of training. The range of $\overline{\Omega}$ spans from 500 to 5000. And we set the maximum staleness $\Omega_{max} = 3\overline{\Omega}$ empirically. Our observations reveal that accuracy declines when staleness is either too small or too large. Specifically, when staleness is too small, local updates on workers become limited, resulting in idle computation resources. Conversely, excessive staleness leads to stale models causing high gradient divergence and reduced model accuracy. For instance, in the left plot of Fig. 1, during MNIST training, accuracy peaked at 91.2% when $\overline{\Omega} = 2000$. Consequently, we adopt $\overline{\Omega} = 2000$ for subsequent experiments.



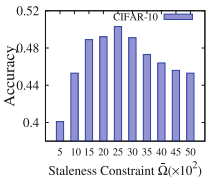
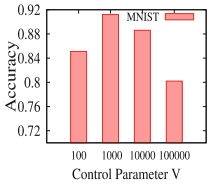


Fig. 1. Accuracy Reached under Staleness Control. *Left*: MNIST; *Right*: CIFAR-10.



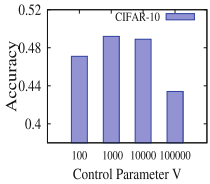


Fig. 2. Accuracy Reached under Control Parameter V. *Left*: MNIST; *Right*: CIFAR-10.

The Impact of Control Parameter. Figure 2 illustrates the impact of the control parameter V on accuracy after 5000 s of training. Notably, a larger value of V prioritizes training time, while a smaller value emphasizes staleness control. We observe that when $V = 1000$, the model achieves its highest accuracy. For instance, in the right plot of Fig. 2, when training on the CIFAR-10, the accuracy reaches a maximum of 49.2% at $V = 1000$. Consequently, we select $V = 1000$ for subsequent experiments.

Performance Comparison. We compare our SA-ADFL with two other typical DFL mechanisms: HADFL and FedHP. As shown in Figs. 3 and 4, the loss and accuracy curves of HADFL exhibit more pronounced jitter compared to those of SA-ADFL and FedHP due to its asynchronous communication among workers without staleness control. In contrast, the curves of FedHP remain stable due to its adoption of synchronous communication among all workers. Our SA-ADFL strikes a balance between fully synchronous and asynchronous communication by staleness control, resulting in superior DFL performance. For example, in Fig. 3, after 5000 s of training, SA-ADFL achieves a stable training accuracy of 91.2%, outperforming HADFL (83.4%) and FedHP (76.2%). Furthermore, the training time required to reach 75% accuracy is 1735 s for SA-ADFL, 3680 s for HADFL, and 4878 s for FedHP. SA-ADFL reduces training time by approximately 52.9% and 64.4% while achieving the same accuracy compared with HADFL and FedHP, respectively.

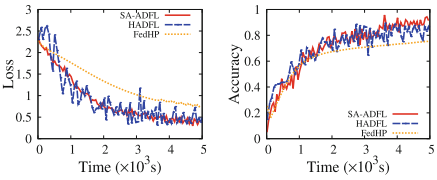


Fig. 3. Loss/Accuracy vs. Round (MNIST). *Left*: Loss; *Right*: Accuracy.

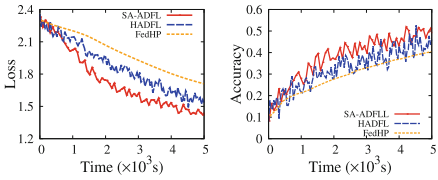


Fig. 4. Loss/Accuracy vs. Round (CIFAR-10). *Left*: Loss; *Right*: Accuracy.

7 Conclusion

In this paper, we have proposed a novel staleness-aware asynchronous decentralized federated learning (SA-ADFL) mechanism to balance the model training efficiency and quality in edge computing. We have rigorously proved the convergence of SA-ADFL and formulated a worker scheduling problem to minimize total model training time given flexible long-term staleness constraints. By leveraging Lyapunov optimization, we have transferred the original round-coupled problem into single-round sub-problems, ensuring efficient worker selection and staleness queue stability. Our experimental results have demonstrated the superiority of SA-ADFL compared with the state-of-the-art solutions.

Acknowledgement. The corresponding author of this paper is Qingming Jia. This work was supported in part by the National Science Foundation of China (NSFC) under Grants 92367104 and 92267301, in part by the National Key Research and Development Program of China under Grant 2023YFB2704200, in part by the Hefei Municipal Natural Science Foundation under Grant 2022013, in part by the Jiangsu Province Excellent Postdoctoral Program under Grant JB23085, and in part by the Purple Mountain Talents-Jiangning Baijia Lake Plan Program under Grant 74072203-3.

References

1. Hu, Y., et al.: Industrial internet of things intelligence empowering smart manufacturing: a literature review. IEEE Internet of Things J. (2024)
2. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, A.Y.B.: Communication-efficient learning of deep networks from decentralized data. AISTATS, 1273–1282 (2017)
3. Li, M., et al.: Scaling distributed machine learning with the parameter server. In: 11th USENIX Symposium on Operating Systems Design and Implementation OSDI, vol. 14, pp. 583–598 (2014)
4. Roy, A.G., Siddiqui, S., Pölsterl, S., Navab, N., Wachinger, C.: Braintorrent: a peer-to-peer environment for decentralized federated learning, arXiv preprint arXiv:1905.06731 (2019)
5. Tang, Z., Shi, S., Li, B., Chu, X.: Gossipfl: a decentralized federated learning framework with sparsified and adaptive communication. IEEE Trans. Parallel Distrib. Syst. **34**(3), 909–922 (2022)
6. Wang, J., Sahu, A.K., Yang, Z., Joshi, G., Kar, S.: MATCHA: speeding up decentralized SGD via Matching Decomposition Sampling. In: Sixth Indian Control Conference (ICC), vol. 2019, pp. 299–300 (2019)
7. Xu, H., Chen, M., Meng, Z., Xu, Y., Wang, L., Qiao, C.: Decentralized machine learning through experience-driven method in edge networks. IEEE J. Sel. Areas Commun. **40**(2), 515–531 (2022)
8. Liao, Y., Xu, Y., Xu, H., Wang, L., Qian, C.: Adaptive configuration for heterogeneous participants in decentralized federated learning. In: IEEE INFOCOM 2023-IEEE Conference on Computer Communications, pp. 1–10. IEEE (2023)
9. Bellet, A., Kermarrec, A.-M., Lavoie, E.: D-cliques: compensating for data heterogeneity with topology in decentralized federated learning. In: 41st International Symposium on Reliable Distributed Systems (SRDS), vol. 2022, pp. 1–11. IEEE (2022)

10. Yao, Z., Liu, J., Xu, H., Wang, L., Qian, C., Liao, Y.: Ferrari: a personalized federated learning framework for heterogeneous edge clients. IEEE Trans. Mobile Comput. (2024)
11. Chen, M., Xu, Y., Xu, H., Huang, L.: Enhancing decentralized federated learning for non-iid data on heterogeneous devices. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE), vol. 2023, pp. 2289–2302. IEEE
12. Ma, Q., Xu, Y., Xu, H., Jiang, Z., Huang, L., Huang, H.: FedSA: a semi-asynchronous federated learning mechanism in heterogeneous edge computing. IEEE J. Sel. Areas Commun. **39**(12), 3654–3672 (2021)
13. Cao, J., Lian, Z., Liu, W., Zhu, Z., Ji, C.: Hadfl: heterogeneity-aware decentralized federated learning framework. In: 58th ACM/IEEE Design Automation Conference (DAC), vol. 2021, pp. 1–6. IEEE (2021)
14. Zehtabi, S., Hosseinalipour, S., Brinton, C.G.: Decentralized event-triggered federated learning with heterogeneous communication thresholds. In: 2022 IEEE 61st Conference on Decision and Control (CDC), pp. 4680–4687. IEEE (2022)
15. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization, arXiv preprint arXiv:1903.03934 (2019)
16. Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S.A.: SAFA: a semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. (2020)
17. Chen, Y., Ning, Y., Slawski, M., Rangwala, H.: Asynchronous online federated learning for edge devices with non-iid data. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 15–24. IEEE (2020)
18. Zheng, S., et al.: Asynchronous stochastic gradient descent with delay compensation. In: International Conference on Machine Learning, pp. 4120–4129. PMLR (2017)
19. Zhu, H., Kuang, J., Yang, M., Qian, H.: Client selection with staleness compensation in asynchronous federated learning. IEEE Trans. Veh. Technol. **72**(3), 4124–4129 (2022)
20. Wang, S., et al.: Adaptive federated learning in resource constrained edge computing systems. IEEE J. Sel. Areas Commun. **37**(6), 1205–1221 (2019)
21. Ma, Q., Xu, Y., Xu, H., Liu, J., Huang, L.: Feduc: a unified clustering approach for hierarchical federated learning. IEEE Trans. Mobile Comput. (2024)
22. Nguyen, L., Nguyen, P.H., Dijk, M., Richtárik, P., Scheinberg, K., Takác, M.: Sgd and hogwild! convergence without the bounded gradients assumption. In: International Conference on Machine Learning, pp. 3750–3758. PMLR (2018)
23. Neely, M.: Stochastic network optimization with application to communication and queueing systems. Springer Nature (2022)
24. Ziller, A., et al.: PySyft: a library for easy federated learning. In: Rehman, M.H., Gaber, M.M. (eds.) Federated Learning Systems. SCI, vol. 965, pp. 111–139. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70604-3_5
25. Ouyang, T., Zhou, Z., Chen, X.: Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing. IEEE J. Sel. Areas Commun. **36**(10), 2333–2345 (2018)
26. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: From theory to algorithms. Cambridge university press (2014)
27. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
28. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Citeseer (2009)