Straggler Mitigation and Latency Optimization in Blockchain-based Hierarchical Federated Learning

Zhilin Wang, Xiangdong Hu, Qin Hu, Minghui Xu, Zehui Xiong

***Abstract*—Cloud-edge-device hierarchical federated learning (HFL) has been recently proposed to achieve communication-efficient and privacy-preserving distributed learning. However, there exist several critical challenges, such as the single point of failure and potential stragglers in both edge servers and local devices. To resolve these issues, we propose a decentralized and straggler-tolerant blockchain-based HFL (BHFL) framework. Specifically, a Raft-based consortium blockchain is deployed on edge servers to provide a distributed and trusted computing environment for global model aggregation in BHFL. To mitigate the influence of stragglers on learning, we propose a novel aggregation method, *HieAvg*, which utilizes the historical weights of stragglers to estimate the missing submissions. Furthermore, we optimize the overall latency of BHFL by jointly considering the constraints of global model convergence and blockchain consensus delay. Theoretical analysis and experimental evaluation show that our proposed BHFL based on HieAvg can converge in the presence of stragglers, which performs better than the traditional methods even when the loss function is non-convex and the data on local devices are non-independent and identically distributed (non-IID).**

***Index Terms*—Hierarchical federated learning, blockchain, stragglers, convergence analysis, latency optimization**

I. Introduction

As a representative paradigm of distributed machine learning, federated learning (FL) significantly reduces the cost of data transmission and protects data privacy [1]–[3]. In FL, local devices (i.e., FL clients) such as smartphones [4] and vehicles [5] upload the trained local models to the parameter server (i.e., aggregator) for global model aggregation. However, FL needs multiple rounds of global model aggregation to obtain the optimal model, which not only consumes substantial communication resources of local devices but may cause network congestion and thus long latency of receiving updates at the server. Therefore, communication efficiency becomes one of the major bottlenecks of FL.

Hierarchical federated learning (HFL) provides a promising solution to the above challenge [6]–[9]. The basic idea is to conduct multiple intermediate aggregations at proxy servers (e.g., edge servers) before global aggregation on the central server. Local devices upload the model updates to a closer proxy server for model aggregation, reducing their communication cost. As demonstrated in [10], HFL can effectively reduce communication latency. Nevertheless, HFL still faces many problems. First, HFL requires a central server for global model aggregation; once this central server fails, HFL cannot work anymore. In addition, proxy servers in the intermediate layer introduce a new attack surface, where the privacy leakage of local model updates and other malicious attacks (e.g., model poisoning attacks) become severe threats [11]–[14].

Blockchain [15], [16], as a distributed ledger technology, has been widely applied to the fields of distributed machine learning [17]–[19]. Considering that blockchain can establish a decentralized and trustless computing environment, we can similarly implement blockchain on proxy servers to take the place of the central server in HFL to reduce the risks of the single point of failure and malicious attacks. There exist some studies [20], [21] deploying blockchain in HFL to protect privacy and improve efficiency, termed blockchain-based HFL (BHFL); but they still rely on the central server to aggregate global models. Furthermore, applying blockchain on HFL can lead to extra latency during broadcasting, verification, and consensus to generate a new block. Although there exists research that optimizes the latency of BHFL by designing resource allocation mechanisms among devices [22], the influence of blockchain consensus on latency remains a challenge.

Apart from the latency issue, the challenge of stragglers also remains unaddressed in BHFL. Here the straggler refers to any participant, including local devices and proxy servers, that cannot submit the model updates in time due to insufficient computing resources or unstable network conditions. The communication efficiency would be directly affected by the stragglers, as waiting for updates from all clients can lead to significant time consumption. Besides, in the case of permanent stragglers that never rejoin FL, simply abandoning their updates can lead to poor performance of the global model, especially when the data of local devices are non-independent and identically distributed (non-IID) [23].

The existing studies mitigating the impact of stragglers on FL fall into three lines, i.e., utilizing *coded computing technology* [24]–[26], manipulating *delayed gradients* [27], [28], and employing *straggler-aware adaptation* [29], [30]. The coded computing method requires extra encoding/decoding processes and data transmission, which is not computation or communication efficient to be applied to various deep learning models. The scheme of manipulating delayed gradients can well address stragglers lacking computing power, where they can still submit partial gradients for model aggregation; however, this method cannot deal with stragglers caused by

Zhilin Wang is with the Department of Computer Science, Purdue University, Indianapolis, Indiana, USA. Email: wang5327@purdue.edu

Xiangdong Hu is with the Department of Computer Science, Georgia State University, Atlanta, Georgia, USA. Email: xhu20@student.gsu.edu.

Qin Hu (corresponding author) is with the Department of Computer Science, Georgia State University, Atlanta, Georgia, USA. Email: qhu@gsu.edu

Minghui Xu is with the School of Computer Science and Technology, Shandong University, China. E-mail: mhxu@sdu.edu.cn

Zehui Xiong is the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK. E-mail: z.xiong@qub.ac.uk

network disconnection or congestion, where the aggregator cannot obtain any data from stragglers within the required time. For straggler-aware adaptation, existing methods mitigate stragglers by reusing the client's most recent update if the current submission is unavailable; but these adaptations hinge on historical client updates and accurate profiling of "invariant" components, thus introducing estimation bias and extra storage overhead. In HFL, proxy servers can also become stragglers due to unexpected connection failures; since proxy servers are not responsible for model training, none of the above approaches premised on client-side participation can resolve proxy-server stragglers. In summary, there is no existing solution for the straggler challenge in HFL.

To fill these gaps, we propose a fully decentralized BHFL framework, which is proven to be convergence-guaranteed even with stragglers existing in local devices and edge servers. Specifically, we deploy a lightweight Raft-based consortium blockchain [31] on edge servers to provide a secure and trusted computing environment for HFL. To address the challenge of stragglers in BHFL, we design a novel aggregation algorithm, named *hierarchical averaging (HieAvg)*, to aggregate model updates submitted from local devices and edge servers at the edge aggregation and global aggregation phases, respectively. The basic idea of HieAvg is to estimate the missing weights with the differences between the historical weights of stragglers, and HieAvg can work well with non-IID data and non-convex loss functions. Further, to improve the system efficiency of BHFL, we optimize the overall latency of BHFL by balancing the performance of the global model and the time cost of blockchain consensus.

To the best of our knowledge, this is the first work to solve the problem of stragglers in BHFL. The proposed HieAvg is applicable to not only our considered BHFL framework but also the general HFL scenarios. The main contributions can be summarized below:

- We propose a decentralized BHFL framework that can converge even when there are stragglers in both local devices and edge servers with non-convex loss function and non-IID data.
- We design HieAvg, a novel model aggregation method for BHFL, to mitigate the negative impact of stragglers by utilizing their historical weights to estimate the missing weights, and its convergence is theoretically proved.
- We optimize the total latency of BHFL by deriving the optimal number of aggregation rounds on edge servers under the constraints of blockchain consensus time consumption and global model convergence.
- Rigorous theoretical analysis and extensive experiments are conducted to prove the convergence of BHFL with HieAvg and evaluate the validity and efficiency of our proposed schemes.

The rest of this paper is organized as below. We introduce the system model in Section II. Then, we analyze the convergence of BHFL with HieAvg in Section IV, and the latency optimization is shown in Section V. Next, we conduct experiments to support our framework and mechanisms in Section VI. The related work is discussed in Section VII. Finally, we conclude this work in Section VIII. The detailed proofs of theorems and lemmas are presented in the appendix.

II. Blockchain-based Hierarchical Federated Learning

In this section, we introduce our considered blockchain-based hierarchical federated learning (BHFL) framework, consisting of an HFL system and a blockchain system, where the blockchain is applied to improve efficiency and provide a trustworthy computing environment for the HFL system. Specifically, we discuss the overview of BHFL, the detailed descriptions of the HFL process and blockchain system, and the challenges of stragglers and latency in this framework.

A. System Overview

As shown in Fig. 1, the considered BHFL system comprises multiple edge servers and local devices, where a consortium blockchain runs on edge servers. Specifically, local devices and edge servers form several FL systems. Let $i \in \{1, 2, \cdots, N\}$ denote the edge server, where N is the total number of edge servers. For edge server i, there are J_i local devices connected, and we let $j \in \{1, 2, \cdots, J_i\}$ denote each local device connected with edge server i. These devices involved in BHFL are heterogeneous, which means they have various computing and communication resources, as well as different raw data distributions, i.e., non-independent and identically distributed (non-IID) data. Denote $k \in \{1, 2, \cdots, K\}$ as the edge aggregation round, i.e., the round of model aggregation on edge servers based on the local model updates from connected devices, where K is the total number of edge aggregation rounds; let $t \in \{1, 2, \cdots, T\}$ be the global aggregation round, i.e., the round of model aggregation on the blockchain system based on the submissions from edge servers, where T is the total number of global aggregation rounds. We use the pair (t, k) to denote edge aggregation round k in the global aggregation round t and use (i, j) to denote local device j connected to edge server i.

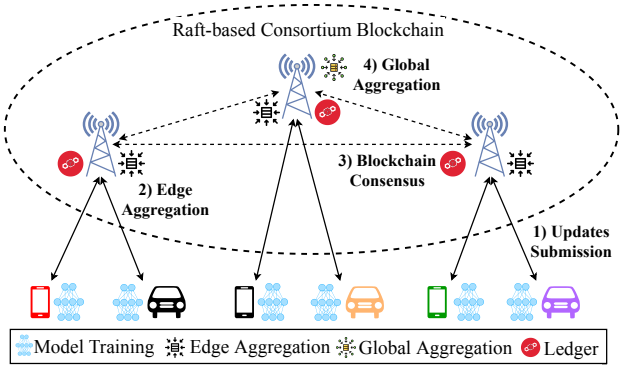


Fig. 1. Blockchain-based Hierarchical Federated Learning.

The workflow of our proposed BHFL system can be described below:

1) **Updates Submission**: after multiple local iterations, i.e., the gradients updating of Stochastic Gradient Descent

(SGD), local devices submit their trained local models to the connected edge servers.
2) **Edge Aggregation**: edge servers aggregate the received local models to get the edge models by our proposed HieAvg which will be detailed in Section III, and return them to local devices for the next round of training till finishing K rounds of edge aggregation.
3) **Blockchain Consensus**: while local devices are conducting model training using their own data, edge servers can perform the consensus algorithm in the upper blockchain network, where one edge leader will be elected before global aggregation. (see Section II-C for details).
4) **Global Aggregation**: after K rounds of edge aggregation, edge servers transmit their latest edge models to the edge leader for global model aggregation by HieAvg, and the edge leader will return the updated global model to edge servers for the next round of training till the BHFL model converges.

B. Hierarchical Federated Learning

In FL, the participants work together to solve a finite-sum optimization problem with SGD, while in hierarchical FL (HFL), the hierarchical SGD (H-SGD) is adopted [9]. The main difference between SGD and H-SGD is that H-SGD requires several rounds of intermediate aggregation before global aggregation.

In HFL, we can treat the framework of local devices and their connected edge server i as FL, and its objective function can be expressed as:

$$\underset{w_i^{t,k}\in\mathbb{R}^d}{\arg\min} \quad F_i(w_i^{t,k}) = \frac{1}{J_i}\sum_{j=1}^{J_i} F_{i,j}(w_{i,j}^{t,k}),$$

where $F_{i,j}(\cdot)$ is the loss function of local device j and $F_i(\cdot)$ is the loss function of edge server i; and $w_{i,j}^{t,k}$ is the weights of local device (i,j) in round (t,k) and $w_i^{t,k}$ is the weights of edge server i in round (t,k).

On local device (i,j), the model is updated by:

$$w_{i,j}^{t,k+1} = w_{i,j}^{t,k} - \eta^{t,k}\nabla F_{i,j}(w_{i,j}^{t,k}, \xi_{i,j}^{t,k}), \tag{1}$$

where $\eta^{t,k}$ is the learning rate in round (t,k); and $\nabla F_{i,j}(w_{i,j}^{t,k}, \xi_{i,j}^{t,k})$ is the gradient of $F_{i,j}(w_{i,j}^{t,k})$ with $\xi_{i,j}^{t,k}$ being the random data sample from the raw data of local device (i,j), and we can assume $\mathbb{E}_j[\nabla F_{i,j}(w_{i,j}^{t,k})] = \nabla F_i(w_i^{t,k})$ where $\mathbb{E}(\cdot)$ is the expectation notation.

On the edge leader, the objective function is defined as:

$$\underset{w^t\in\mathbb{R}^d}{\arg\min} \quad F(w^t) = \frac{1}{N}\sum_{i=1}^{N} F_i(w_i^t),$$

where $F(\cdot)$ is the global loss function, w^t is the global weights in round t and w_i^t is the weights of edge server i in round t with $w_i^t = w_i^{t,K}$.

As for the gradient of edge server i, we can assume $\mathbb{E}_i[\nabla F_i(w_i^t)] = \nabla F(w^t)$. After T rounds of global aggregation, we can get the final global model w^T, which should be as approximate as possible to the optimal global model w^* of $F(\cdot)$.

C. Blockchain

Generally, there are two ways for synchronizing the model updates in HFL so that the edge servers and local devices can get the latest models in the next round, i.e., *centralized* and *distributed*. In the centralized method, a cloud server is employed to send the latest global model to the edge servers, such as the client-edge-cloud framework presented in [10]; while in the distributed way, local model updates are directly shared among peering edge servers via broadcasting, and then each server can derive the global model based on the received updates from others. Though the distributed method can eliminate the single point of failure, broadcasting models among edge servers can be costly in communication resource consumption and latency.

To solve this challenge, blockchain is widely employed to assist in enhancing the efficiency and performance of HFL. In general, two essential requirements have to be satisfied. First, the deployment of blockchain in HFL cannot bring a significant latency increase to the global model convergence, including the latency of computing, communication, and blockchain consensus. Some of the existing studies use computing-intensive blockchain consensus, such as Proof of Work (PoW) [32], and require model sharing among participants before global model aggregation, making the time cost extremely high [33]. Second, the implemented blockchain system needs to be capable of protecting data privacy, i.e., local model updates and global models, from leakage.

For these considerations, we propose a consortium blockchain based on the Raft consensus protocol [34]. Since the consortium blockchain only allows authorized nodes to be included, the edge servers in this network can thus be trusted; further, as a leader-based consensus mechanism, Raft has been proven to be efficient and reliable. The main working process of the Raft-based consortium blockchain in BHFL can be summarized into the following three steps:

1) **Leader Election**: edge servers on the blockchain conduct the leader election process[1], which should be completed before submitting local models for global aggregation so that the latency of BHFL can be reduced compared to the existing work [33].
2) **Model Submission**: edge servers submit their latest edge models to the elected edge leader, and then the leader will aggregate those models to update the global model.
3) **Block Generation**: the leader generates a new block that contains all edge models from edge servers and the updated global model, and broadcasts this block to all edge servers on the blockchain.

Here the Raft-based consortium blockchain in the BHFL system is mainly employed to avoid the single point failure and provide a trustless intermediary for synchronizing model updates among edge servers. It is worth noting that the edge servers in the blockchain system are required to finish the leader election process before conducting global aggregation to reduce latency and communication overhead due to broadcasting. As for the details of the latency in blockchain consensus

[1]For brevity, we omit the detailed leader election process of Raft, which can be found in [34].

influencing the total system latency, which has been rarely explored in the existing studies, we will discuss them in Section V.

D. Challenges of BHFL

As mentioned before, there exist two main challenging factors in BHFL that impact learning performance, i.e., stragglers and latency. We denote $\mathcal{L}_e$ as the predefined waiting period of each edge aggregation round and $\mathcal{L}_g$ as the waiting time of each global aggregation round. The stragglers are BCFL participants that cannot upload models within the required waiting period in both the layers of local devices and edge servers. On the one hand, local devices may not submit model updates to edge servers in time; on the other hand, edge servers may miss the required deadline for submitting edge models. These stragglers can result in long latency for the model training. In addition to the latency caused by stragglers, the numbers of local devices, edge servers, edge aggregation rounds, and global aggregation rounds, as well as the blockchain consensus process, will influence the total time consumption.

To resolve the challenge brought by stragglers in BHFL, we first propose a novel model aggregation algorithm, named HieAvg, to mitigate the impact of stragglers in both layers of BHFL, which will be elaborated in Section III. Further, to deal with the challenge of latency, we design an optimization scheme detailed in Section V.

III. Hierarchical Averaging Aggregation Method

In this section, we elaborate on our proposed hierarchical averaging (HieAvg) aggregation method. The basic idea of HieAvg is to use the historical weights of stragglers to estimate their delayed weights. Generally speaking, there are two main parts of HieAvg: the first part is the basic aggregation method when edge servers would like to collect model submissions from all clients no matter whether there is any straggler or not; and the second is the straggler mitigation method with two steps which will be detailed below.

A. Basic Aggregation Methods of HieAvg

In the case that edge servers and the edge leader wait for weights from all connected devices without dealing with the stragglers' impact on BHFL convergence, they may use the following aggregation methods to update the edge models and the global model.

1) Edge Aggregation: Let $M_i^{t,k}$ be the number of devices that can submit weights to edge server i in round (t,k) within the time requirement, and let $S_i^{t,k}$ denote the number of stragglers among local devices in round (t,k). Thus, we have $J_i = M_i^{t,k} + S_i^{t,k}$, and can get the model of edge server i in round (t,k) as

$$w_i^{t,k} = \frac{1}{J_i}\sum_{j=1}^{J_i} w_{i,j}^{t,k} = \frac{1}{J_i}\Big(\sum_{m=1}^{M_i^{t,k}} w_{i,m}^{t,k} + \sum_{s=1}^{S_i^{t,k}} w_{i,s}^{t,k}\Big), \quad (2)$$

where $w_{i,m}^{t,k}$ and $w_{i,s}^{t,k}$ are the in-time and delayed local weights in round (t,k), respectively.

2) Global Aggregation: We use the following equation to update the global model in round t:

$$w^t = \sum_{i=1}^{N} \frac{J_i}{\sum_{i=1}^{N} J_i} w_i^t = \sum_{m=1}^{M^t} \frac{J_m^t}{\sum_{i=1}^{N} J_i} w_m^t + \sum_{s=1}^{S^t} \frac{J_s^t}{\sum_{i=1}^{N} J_i} w_s^t, \quad (3)$$

where M^t is the number of edge servers submitting updates to the edge leader timely in round t, and S^t is the number of stragglers among edge servers in round t; J_m^t and J_s^t are the numbers of local devices connected to edge server m submitting models in time and that of straggler s in round t, respectively; w_m^t and w_s^t are the in-time and delayed edge weights in round t, respectively.

Note that even if there is no straggler, i.e., $S_i^{t,k} = 0$ and $S^t = 0$, the BHFL system can also apply the above equations to aggregate models.

Remark: Based on the basic aggregation methods of HieAvg, we can find that it differs from FedAvg [1] in the following two aspects: i) HieAvg does not require the data size in edge aggregation, avoiding additional data disclosure of local devices; and ii) HieAvg uses the ratio of J_i to the total number of local devices in global aggregation, which is more suitable for HFL, and FedAvg cannot be applied in this situation.

B. Straggler Mitigation of HieAvg

In this part, we detail the design of HieAvg to mitigate the stragglers' impact, including the steps of *cold boot* and *estimation of delayed weights*.

1) Cold Boot: To better estimate the stragglers' delayed submissions, the BHFL system has to collect enough historical data in the process of cold boot. Denoting T_c as the number of model submission rounds, we require all participants, including local devices and edge servers, to finish at least two rounds of model submission, i.e., $T_c \geq 2$, so that the necessary amount of information can be collected. Ideally, all devices can submit models in time for the first T_c rounds, and the step of cold boot can be described in Algorithm 1. The edge servers need to wait for submissions from local devices for T_c global aggregation rounds (Line 1). During cold boot, we use (2) and (3) to aggregate the models on edge servers and the edge leader, respectively (Lines 2-13).

In the case that one device loses connection after the first round of model submission while other devices can continue working, if the device is reconnected and submits weights after multiple rounds, the resubmitted weights will be considered as the historical weights.

2) Estimation of Delayed Weights: After cold boot, we design the scheme of delayed weight estimation to mitigate the impact of stragglers by estimating their delayed weights via the historical weights.

Estimation of Delayed Local Weights: Since the edge servers have the historical weights of stragglers, we can use those weights to estimate the delayed weights of stragglers. We have to ensure that the difference between the estimated weights and the real delayed weights is as small as possible. To that aim, we design an approximate method by utilizing the difference in historical weights of stragglers to estimate their

Algorithm 1 Cold Boot in HieAvg

Require: T_c, N, K, J_i, $\eta^{t,k}$
Ensure: w^{T_c}
1: **for** $t \in \{1, \cdots, T_c\}$ **do**
2: **for** $i \in \{1, \cdots, N\}$ parallelly **do**
3: **for** $k \in \{1, \cdots, K\}$ **do**
4: **for** $j \in \{1, \cdots, J_i\}$ parallelly **do**
5: $w_{i,j}^{t,k} \leftarrow$ updated by (1)
6: **end for**
7: $w_i^{t,k} \leftarrow$ updated by (2)
8: **end for**
9: $w_i^t \leftarrow w_i^{t,K}$
10: **end for**
11: $w^t \leftarrow$ updated by (3)
12: **end for**
13: **return** w^{T_c}

Algorithm 2 Estimation of Delayed Weights in HieAvg

Require: T, T_c, N, K, J_i, γ_0, λ, $\eta^{t,k}$
Ensure: w^T
1: **for** $t \in \{T_c + 1, \cdots, T\}$ **do**
2: **for** $i \in \{1, \cdots, N\}$ parallelly **do**
3: **for** $k \in \{1, \cdots, K\}$ **do**
4: $\overline{w}_i^{t,k} \leftarrow$ updated by (4)
5: $w_i^{t,k} \leftarrow \overline{w}_i^{t,k}$
6: **end for**
7: $w_i^t \leftarrow w_i^{t,K}$
8: **end for**
9: $\overline{w}^t \leftarrow$ updated by (5)
10: $w^t \leftarrow \overline{w}^t$
11: **end for**
12: **return** w^T

delayed weights in round (t, k). The estimation of delayed weights is:

$$\overline{w}_{i,s}^{t,k} = w_{i,s}^{t,k-1} + \mathbb{E}_k[\Delta_{i,s}^{t,k-1}],$$

where $\Delta_{i,s}^{t,k-1} = w_{i,s}^{t,k-1} - w_{i,s}^{t,k-2}$, and $\mathbb{E}_k[\Delta_{i,s}^{t,k-1}]$ is the expectation of $\Delta_{i,s}^{t,k-1}$ used to avoid large estimation bias.

Then, the estimated $w_i^{t,k}$ can be written as:

$$\overline{w}_i^{t,k} = \frac{1}{J_i}\left[\sum_{m=1}^{M_i^{t,k}} w_{i,m}^{t,k} + \sum_{s=1}^{S_i^{t,k}} \gamma_{i,s}^{t,k}(w_{i,s}^{t,k-1} + \mathbb{E}_k[\Delta_{i,s}^{t,k-1}])\right], \tag{4}$$

where $\gamma_{i,s}^{t,k} = \gamma_0 \lambda^{k'}$ is the decay factor used to scale estimated delayed weights with $\gamma_0 \in (0, 1)$ being the initial decay factor, $\lambda \in (0, 1)$ being the scalar, and $k' \geq 1$ being the missing edge aggregation rounds of stragglers.

Estimation of Delayed Edge Weights: As for stragglers among edge servers, we use the same estimation method for dealing with stragglers among local devices. Then, we can get the estimated w^t by the following equation:

$$\overline{w}^t = \sum_{m=1}^{M^t} \frac{J_m^t}{\sum_{i=1}^N J_i} w_m^t + \sum_{s=1}^{S^t} \frac{\gamma_s^t J_s^t}{\sum_{i=1}^N J_i}(w_s^{t-1} + \mathbb{E}_t[\Delta_s^{t-1}]), \tag{5}$$

where $\gamma_s^t = \gamma_0 \lambda^{t'}$ is the delay factor with t' being the missing global aggregation rounds of stragglers; and $\Delta_s^{t-1} = w_s^{t-1} - w_s^{t-2}$.

The process of estimating delayed weights in HieAvg is detailed in Algorithm 2. If there are stragglers, the corresponding edge server and the edge leader will use the estimation method to update the models (Line 4 and Line 9). Please note that this algorithm is used to handle situations where there exist stragglers; while if there are no stragglers, the model updating will be the same as Algorithm 1. Here we can see that the time complexity of HieAvg, including Algorithms 1 and 2, is $\mathcal{O}(T \times N \times K \times J)$.

In fact, there exist two types of stragglers for both the stragglers among local devices and edge servers in BHFL: *permanent stragglers* and *temporary stragglers*. Without loss of generality, we take the stragglers among edge servers as an example to clarify this point. First, if the stragglers will never return to join the BHFL training process due to the loss of communication connection or location change at global round t, then we can use the above method to estimate the updates of stragglers starting from round t to the end of training. We call this kind of stragglers permanent stragglers. Second, if the stragglers will return after $t' \geq 1$ rounds, we can still first use the above method to estimate the updates of stragglers during rounds t to $t + t'$, and once the stragglers return in round $t + t' + 1$, they can submit their latest models. These stragglers are named temporary stragglers. Intuitively, permanent stragglers are more harmful to BHFL than temporary stragglers since the bias will be larger if the stragglers disappear. Thus, we treat the permanent stragglers as the worst case in the following section of convergence analysis.

IV. Convergence Analysis of HieAvg-based BHFL

In this section, we analyze the convergence of BHFL with HieAvg. We first introduce the necessary assumptions for theoretical proof and then discuss the convergence of edge aggregation and global aggregation subsequently.

A. Assumptions

Here we introduce two assumptions that are important for the proof of convergence. The first one indicates the property of the loss function employed in our proposed BHFL framework, which has also been widely included in the existing studies [9], [22], [35]. The second ensures that the model updating process will not lead to a significant bias.

Assumption 1. *(Lipschitz-smoothness) The loss function $F(\cdot)$ is continuously differentiable and the gradient function of $F(\cdot)$ is Lipschitz continuous with Lipschitz constant $L > 0$, which means $||\nabla F(w) - \nabla F(\overline{w})||^2 \leq L||w - \overline{w}||^2$ for all $w, \overline{w} \in \mathbb{R}$. It also implies that*

$$F(w) - F(\overline{w}) \leq \nabla F(w)^T(\overline{w} - w) + \frac{L}{2}||w - \overline{w}||^2,$$

where $||\cdot||^2$ is the l_2 norm.

Assumption 2. *(Bounded Variance) Three types of bounded variance are assumed:*

1) Bounded Variance of Weight Difference:

$$\mathbb{E}_k[||(w_{i,j}^{t,k} - w_{i,j}^{t,k-1}) - \Delta_{i,j}||^2] \leq \delta_{i,j}^2,$$
$$\mathbb{E}_t[||(w_i^t - w_i^{t-1}) - \Delta_i||^2] \leq \delta_i^2,$$

where $\Delta_{i,j} = \mathbb{E}_k[\Delta_{i,j}^{t,k-1}]$ *and* $\Delta_i = \mathbb{E}_t[\Delta_i^t]$*; and* $\delta_{i,j}, \delta_i \in \mathbb{R}^+$.

2) Bounded Variance of Estimated Gradients:

$$\mathbb{E}_j[||\nabla F_{i,j}(\overline{w}_{i,j}^{t,k}) - \nabla F_i(\overline{w}_i^{t,k})||^2] \leq \delta'^2,$$
$$\mathbb{E}_i[||\nabla F_i(\overline{w}_i^t) - \nabla F(\overline{w}^t)||^2] \leq \delta''^2,$$

where $\nabla F_i(\overline{w}_i^{t,k}) = \mathbb{E}_j[\nabla F_{i,j}(\overline{w}_{i,j}^{t,k})]$ *and* $\nabla F(\overline{w}^t) = \mathbb{E}_i[\nabla F_i(\overline{w}_i^t)]$*; and* $\delta', \delta'' \in \mathbb{R}^+$.

3) Bounded Variance of Estimated Delayed Weights:

$$\mathbb{E}_i[||\overline{w}_i^{t,k} - \overline{w}^{t,k}||^2] \leq \overline{\delta}^2,$$
$$\mathbb{E}_t[||\overline{w}^t - \overline{w}||^2] \leq \overline{\delta}'^2,$$

where $\overline{w}^{t,k} = \mathbb{E}_i[\overline{w}_i^{t,k}]$ *is the auxiliary variable inspired by [36], [37]; and* $\overline{w} = \mathbb{E}_t[\overline{w}^t]$*; and* $\overline{\delta}, \overline{\delta}' \in \mathbb{R}^+$.

Assumption 2.1 is unique in this work since we use the difference of weights to estimate the delayed weights, which is assumed to have bounded variance; and Assumptions 2.2 and 2.3 are about the estimated weights, guaranteeing that the estimation method will not lead to significant bias.

It is worth noting that the learning rate $\eta^{t,k} = \frac{1}{\eta_0 + d(tK+k)}$ is assumed to be dynamic, where η_0 is the initial learning rate for all the local devices and d is the decay rate. Besides, we have no assumption on the convexity of the loss function; however, since the non-convex case is more challenging, we will analyze the convergence of HieAvg in BHFL with the non-convex loss function in the below.

B. Convergence of HieAvg

Before we discuss the convergence of HieAvg on both layers, we need to introduce two useful lemmas which will be applied in the proof of convergence.

Lemma 1. *Under Assumption 2, by applying HieAvg on edge servers, the difference between the estimated edge model in round* $(t, k+1)$ *and that in round* (t, k) *is bounded by*

$$\overline{w}_i^{t,k+1} - \overline{w}_i^{t,k} \leq \overline{\delta} - \eta^{t,k}\nabla F_i(\overline{w}_i^{t,k}) - \gamma_0 \frac{S_i^{t,k}}{J_i}(\Delta_{i,j} + \delta_{i,j}^2),$$

where $\frac{S_i^{t,k}}{J_i}$ *denotes the proportion of stragglers among local devices connected to edge server* i *in round* (t, k).

Lemma 2. *Under Assumption 2, by applying HieAvg on the edge leader, the difference between the estimated global model in round* $t+1$ *and that in round* t *is bounded by*

$$\overline{w}^{t+1} - \overline{w}^t \leq \overline{\delta}' - [\frac{\mathbb{E}_s[J_s^t]}{N\mathbb{E}_i[J_i]} + \gamma_0 \frac{S^t}{N}(\Delta_i + \delta_i^2)] - \frac{K\mathbb{E}_s[J_s^t]}{N\mathbb{E}_i[J_i]}\eta^{t,k}\nabla F(\overline{w}^t),$$

where $\frac{S^t}{N}$ *is the proportion of stragglers among edge servers in round* t.

Both Lemma 1 and Lemma 2 imply that the difference in estimated weights will be affected by the previous weight differences, the delayed weights, and the proportion of stragglers. By now, however, it is still unclear how these factors influence the convergence of HieAvg based on the above two lemmas, which will be explored in the following two subsections.

1) Convergence on Edge Servers: We first investigate the convergence of HieAvg on edge servers. By analyzing the convergence on edge servers, we can see the effectiveness of our proposed HieAvg algorithm.

Theorem 1. *Under Assumption 1 and Assumption 2, with dynamic learning rate* $\eta^{t,k}$*, the number of stragglers* $S_i^{t,k}$*, and the number of connected local devices* J_i*, if* $\eta^{t,k} > \frac{1}{L+2}$ *with* $L > 0$ *and* $\gamma_0 \frac{\mathbb{E}_k[S_i^{t,k}]}{J_i}(\Delta_{i,j} + \delta_{i,j}) - \overline{\delta} \geq 0$*, by applying Lemma 1, the convergence of HieAvg on edge server* i *is bounded by*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[||\nabla F_i(\overline{w}_i^{t,k})||^2] \leq \frac{2[F_i(w_i^0) - F_i(w_i^*) + \frac{2\mathbb{E}_k[\eta^{t,k}]\delta'^2}{L\mathbb{E}_k[\eta^{t,k}] + 2\mathbb{E}_k[\eta^{t,k}] - 1}]}{(L\mathbb{E}_k[\eta^{t,k}] + 2\mathbb{E}_k[\eta^{t,k}] - 1)\sqrt{K}} + \frac{(2+L)[\gamma_0 \frac{\mathbb{E}_k[S_i^{t,k}]}{J_i}(\Delta_{i,j} + \delta_{i,j}) - \overline{\delta}]}{L\mathbb{E}_k[\eta^{t,k}] + 2\mathbb{E}_k[\eta^{t,k}] - 1},$$

where w_i^0 *is the initial weights of edge server* i *and* w_i^* *is the optimal weights of edge server* i.

The above inequality provides a theoretical upper bound for the averaging expectation of squared gradient norms of $F_i(\cdot)$, which indicates that the loss function of edge i can converge to a critical point with enough edge aggregation rounds, smaller learning rate, and fewer stragglers among local devices. Besides, Theorem 1 can be employed to analyze the convergence of traditional FL, which is guaranteed to converge even with the existence of stragglers if K is well selected.

2) Convergence on the Edge Leader: Now, we can discuss the global convergence on the blockchain.

Theorem 2. *Under Assumption 1 and Assumption 2, with dynamic learning rate* $\eta^{t,k}$*, the number of stragglers* S^t*, and the number of connected local devices* J_i*, if* $\eta^{t,k} \geq \frac{1}{L + \frac{2K\mathbb{E}_t[J_s^t]}{N\mathbb{E}_i[J_i]}}$ *with* $L > 0$ *and* $\frac{\mathbb{E}_t[J_s^t]}{N\mathbb{E}_i[J_i]} + \gamma_0 \frac{\mathbb{E}_t[S^t]}{N}(\Delta_i + \delta_i^2) - \overline{\delta}' \geq 0$*, by applying Lemma 2, the convergence of HieAvg on the edge leader is bounded by*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[||\nabla F(\overline{w}^t)||^2] \leq \frac{2[F(w^0) - F(w^*) + \frac{\sqrt{K}\mathbb{E}_t[\eta^{t,k}]\mathbb{E}_t[J_s^t]}{N\mathbb{E}_i[J_i]}\delta''^2]}{\sqrt{T}(2\sqrt{K}\frac{\mathbb{E}_t[\eta^{t,k}]\mathbb{E}_t[J_s^t]}{N\mathbb{E}_i[J_i]} + L\mathbb{E}_t[\eta^{t,k}] - 1)} + \frac{(2+L)[\frac{\mathbb{E}_t[J_s^t]}{N\mathbb{E}_i[J_i]} + \gamma_0\frac{\mathbb{E}_t[S^t]}{N}(\Delta_i + \delta_i^2) - \overline{\delta}']}{2\sqrt{K}\frac{\mathbb{E}_t[\eta^{t,k}]\mathbb{E}_t[J_s^t]}{N\mathbb{E}_i[J_i]} + L\mathbb{E}_t[\eta^{t,k}] - 1},$$

where w^0 is the initial global weights; for convenience, we use Ω to represent the upper bound at the right side of the above inequality.

Based on the above theorem, we can see that the averaging expectation of squared gradient norms of $F(\cdot)$ has an upper bound, which implies that BHFL with HieAvg can converge. Besides, by analyzing Theorem 1 and Theorem 2, we can obtain the following two corollaries to further explain the convergence performance of HieAvg.

Corollary 1. *Given the fixed values of other influence factors, the convergence performance can be better achieved with more edge aggregation rounds (K).*

Corollary 1 indicates that we can speed up global convergence with more edge aggregation rounds. This is because if there are more rounds of edge aggregation, each edge server can get a model with a smaller loss, which accelerates the convergence of the global model during the phase of global aggregation.

Corollary 2. *Given the fixed values of other influence factors, the convergence performance can be better achieved with fewer stragglers in local devices and edge servers ($S_i^{t,k}$ and S^t).*

Corollary 2 demonstrates the influence of stragglers on the convergence of HieAvg. The occurrence of stragglers is usually caused by unpredictable network conditions, and it is nearly impossible to eliminate their effects on model training completely; but with our proposed HieAvg, the convergence of BHFL is guaranteed.

In summary, the HieAvg algorithm is convergence-guaranteed with a non-convex loss function and non-IID data even when there are stragglers among local devices and edge servers in BHFL.

V. Latency Optimization of BHFL

In this section, we target to resolve the challenge of latency in BHFL by studying the latency optimization of our proposed framework.

A. Latency Model

1) Latency of Local Devices: By applying Shannon's theory [38], we can calculate the data transmission rate of local device j connected to edge server i in round (t,k) as $r_{i,j}^{t,k} = B_{i,j}^{t,k}\log_2(1+\frac{u_{i,j}^{t,k}\pi_{i,j}^{t,k}}{\epsilon^2})$, where $B_{i,j}^{t,k}$ is the bandwidth of local device (i,j) in round (t,k); $u_{i,j}^{t,k}$ and $\pi_{i,j}^{t,k}$ are the transmission power and channel power gain, respectively; and ϵ is the Gaussian noise. Then, the transmission time of one communication round between the local device j and edge server i can be calculated by $\mathcal{LM}_{i,j}^{t,k} = \frac{D_{i,j}^{t,k}}{r_{i,j}^{t,k}}$, where $D_{i,j}^{t,k}$ is the size of local model updates.

The computing latency before each round of edge aggregation can be computed as $\mathcal{LP}_{i,j}^{t,k} = \frac{C_{i,j}^{t,k}}{f_{i,j}^{t,k}}$, where $C_{i,j}^{t,k}$ is the total CPU cycles required to complete the training in edge round (t,k) and $f_{i,j}^{t,k}$ is the unit CPU cycles of local device (i,j).

Since the communication between the local device j and edge server i includes both model downloading and model update submission, the total latency[2] on local devices during one round of edge aggregation is

$$\mathcal{L}_{lc} = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{J_i}(2\mathcal{LM}_{i,j}^{t,k} + \mathcal{LP}_{i,j}^{t,k}).$$

2) Latency of Edge Servers: On edge servers, they are mainly responsible for intermediate model aggregation and transmission. Here we omit the time consumption of model aggregation since it is negligible compared to that of model transmission. Similarly to the calculation of the communication latency of local devices, we can get the total communication latency of servers as

$$\mathcal{L}_{gb} = 2\sum_{t=1}^{T}\sum_{i=1}^{N}\mathcal{LM}_i^t,$$

where $\mathcal{LM}_i^t$ is the communication time cost for model uploading and downloading of edge server i.

3) Latency of Blockchain Consensus: Let $\mathcal{L}_{bc}$ be the latency of blockchain consensus in each global round, and denote

$$\mathcal{L}_g = K\max_{t\le T_c}(\mathcal{LM}_{i,j}^{t,k} + \mathcal{LP}_{i,j}^{t,k}),$$

as the waiting period for round t. Then $\mathcal{L}_{bc} \le \mathcal{L}_g$ becomes a constraint for the Raft-based blockchain system to guarantee that its deployment brings no increase to the overall latency of BHFL.

4) Total Latency: The total latency, denoted as $\mathcal{L}$, is the sum of the latency of local devices and edge servers. Note that the latency of blockchain consensus is not included in the total latency because the blockchain consensus has been completed during K rounds of edge aggregation as required above. Thus, we have:

$$\mathcal{L} = \mathcal{L}_{lc} + \mathcal{L}_{gb} = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{J_i}(2\mathcal{LM}_{i,j}^{t,k} + \mathcal{LP}_{i,j}^{t,k}) + 2\sum_{t=1}^{T}\sum_{i=1}^{N}\mathcal{LM}_i^t.$$

For a rough qualitative analysis, we assume that the number of local devices connected to each edge server is the same, which is denoted by J; and we can assume the latency of each local device is fixed in each round, and thus we use $\mathcal{LM}_{i,j}$ to represent the communication latency of local device (i,j); similarly, we use $\mathcal{LP}_{i,j}$ and $\mathcal{LM}_i$ to stand for the computing latency of local device (i,j) and the latency of edge server i, respectively. Then, we can simplify the above equation as

$$\mathcal{L} \approx TNJK(2\mathbb{E}[\mathcal{LM}] + \mathbb{E}[\mathcal{LP}]) + 2TN\mathbb{E}[\mathcal{LM}'],$$

where $\mathbb{E}[\mathcal{LM}] = \mathbb{E}_i[\mathbb{E}_j[\mathcal{LM}_{i,j}^{t,k}]]$, $\mathbb{E}[\mathcal{LP}] = \mathbb{E}_i[\mathbb{E}_j[\mathcal{LP}_{i,j}^{t,k}]]$ and $\mathbb{E}[\mathcal{LM}'] = \mathbb{E}_i[\mathcal{LM}_i^t]$. We can see that $\mathcal{L}$ and K are positively proportional, and thus we can conclude that reducing the frequencies of edge aggregation can lead to lower communication

[2]This latency can also utilize the value of the slowest device, and the main solution proposed in this section can be applied similarly.

latency. However, we know that larger K will improve the convergence performance of BHFL according to Theorem 2. Thus, K should be determined by jointly considering the performance and latency of BHFL.

B. Latency Optimization

In this part, we formulate the latency optimization problem by reducing latency and maintaining the convergence performance of BHFL at the same time. Based on the above analysis, we know that $\mathcal{L}$ is a linear function of K if we use the expectations of communication and computing latency to calculate $\mathcal{L}$, so we can approximately get the optimal K by solving the following optimization problem:

$$\begin{aligned} \arg\min_{K} \quad & \mathcal{L} \\ \text{s.t. } & \mathrm{C1}: \Omega \leq \overline{\Omega}, \\ & \mathrm{C2}: \mathcal{L}_{bc} \leq \mathcal{L}_{g}, \\ & \mathrm{C3}: K \in \mathbb{N}^{+}, \end{aligned}$$

where C1 is the constraint of convergence performance, ensuring that BHFL can have a good performance to meet the requirement $\overline{\Omega}$; and C2 constrains the waiting time in each global round by considering the time consumption of blockchain consensus; and C3 indicates that K should be a positive integer. Then the above optimization problem becomes a simple integer linear programming with inequality constraints, which can be resolved using classical solutions with polynomial complexity, such as CVXPY [39], to find the optimal number of edge aggregation rounds, i.e., K^*.

VI. Experimental Evaluation

In this section, we evaluate our proposed HieAvg algorithm and latency optimization scheme via extensive experiments. We first introduce the experimental settings and then present the experimental results with discussions.

A. Experimental Settings

BHFL Basic Setting: Unless specified otherwise, we use the following basic setting in our experiments. We simulate a BHFL framework with five edge servers, where each edge server is connected to five local devices. There are two edge aggregation rounds between two rounds of global aggregation, i.e., $K = 2$. Each local device owns at most one class of data. We assume there are 20% stragglers in each layer, which means that one edge server cannot submit the edge model timely in each round of global aggregation and one local device connected to each edge server fails to upload the local model timely in each edge aggregation round, respectively. Besides, we set $\gamma_0 = 0.9$ and $\lambda = 0.9$ for HieAvg.

Stragglers: For permanent stragglers, they stop submitting model updates after 40 rounds. And temporary stragglers miss submissions in multiple single rounds but will continue to submit in the next round after the missing round.

Dataset: We use MNIST [40] as the example dataset in BHFL, which contains 70,000 handwritten digits from 0 to 9. When there are no stragglers, the accuracy is about 87.75%.

Machines and Platforms: We develop our proposed BHFL framework based on Python 3.7 and TensorFlow 2.9 on Google Colab, Raspberry Pi 4 Model B, and AWS EC2. Specifically, we test the convergence of BHFL on Colab with an A100 GPU and explore the latency of communication for model synchronization with Raspberry Pi and AWS EC2.

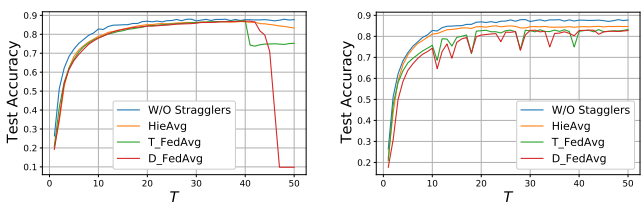
Learning Models: Based on TensorFlow, we create a CNN-based deep learning model with two convolutional layers, one max pooling layer, one flattening layer, and one dense layer. The batch size is 32, the local iteration is one epoch, and the initial learning rate is 0.001 with the decay rate $d = 0.90$.

Benchmarks of Aggregation Methods: We consider three benchmarks based on federated averaging (FedAvg) [1] to compare with our proposed BHFL framework with HieAvg from the convergence perspective. The first benchmark considers no stragglers, which is named as *W/O Stragglers*. For the second solution dealing with stragglers, only the timely submissions from local devices and edge servers will be included in edge aggregation and global aggregation, which is termed as *T_FedAvg*. The third one uses the weights submitted in the last round as the weights of stragglers in round k or t, which is called *D_FedAvg*.

Computing and Communication: We calculate the computing and communication latency on machines and platforms according to the equations in Section V-A. Specifically, we simulate the model training process of one local device on Raspberry Pi, and we let the Raspberry Pi communicate with EC2 to get the latency of communication.

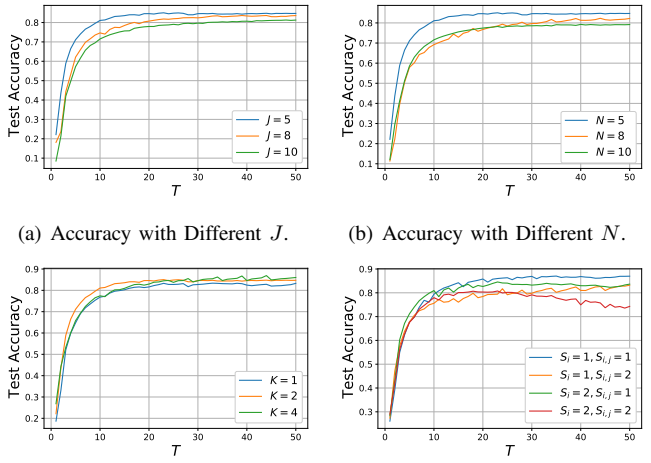
B. Experimental Results

1) Evaluation of Convergence: We first compare the performance of different algorithms in handling both permanent and temporary stragglers. The results are shown in Fig. 2. From Fig. 2(a) involving permanent stragglers, we can see that compared to the ideal case, i.e., W/O Stragglers, the other three algorithms have various losses of accuracy. The accuracy of T_FedAvg decreases a lot, and D_FedAvg fails to converge, while our proposed HieAvg can still have relatively good accuracy in handling permanent stragglers. In Fig. 2(b) dealing with temporary stragglers, all algorithms can achieve good accuracy, but the convergence of HieAvg is smoother and faster than T_FedAvg and D_FedAvg. These two sets of experiments illustrate that different kinds of stragglers affect global convergence, but the proposed HieAvg performs better in both cases.



(a) Permanent Stragglers. (b) Temporary Stragglers.

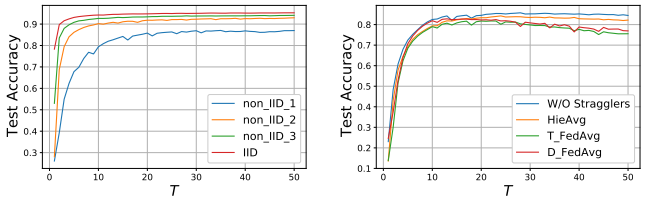
Fig. 2. Comparison with Different Aggregation Algorithms.



(a) Accuracy with Different J. (b) Accuracy with Different N.

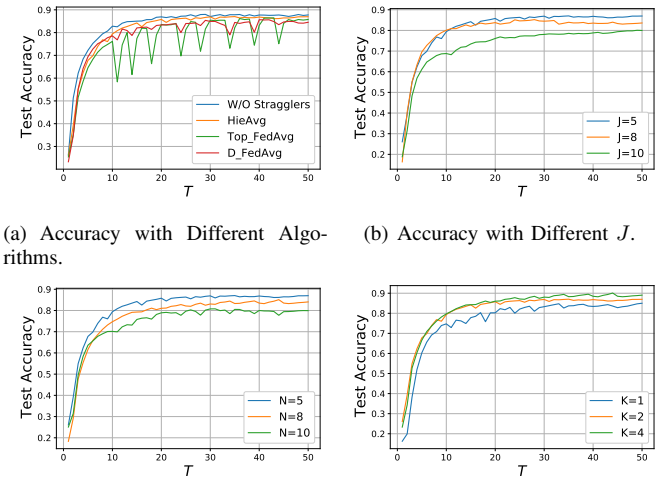
(c) Accuracy with Different K. (d) Accuracy with Different S_i and $S_{i,j}$.

Fig. 3. Influences of Parameters on BHFL Training.



(a) Different Data Distributions. (b) Comparison with Inconsistent J_i.

Fig. 4. Influences of Data Distribution and Local Device Distribution on BHFL Training.



(a) Accuracy with Different Algorithms. (b) Accuracy with Different J.

(c) Accuracy with Different N. (d) Accuracy with Different K.

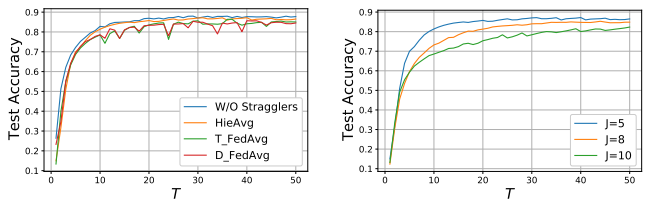
Fig. 5. Test Accuracy with Only Local Device Stragglers.

Next, we explore the impact of different parameter settings, including the numbers of local devices, edge servers, edge aggregation rounds, and stragglers in two layers, on HieAvg with temporary stragglers. By changing J, we obtain the results shown in Fig. 3(a), which indicates that HieAvg converges faster when there are fewer local devices. By adjusting the number of edge servers, as shown in Fig. 3(b), we can get a similar conclusion. This is because increasing the number of local devices and edge servers will aggravate the imbalance of data distribution when the total data volume of MNIST is fixed under the non-IID situation, thus leading to performance degradation. Later, we analyze the influence of K on the accuracy. Fig. 3(c) implies that more edge aggregation rounds help improve the accuracy because the more frequent edge aggregation allows each edge server to better integrate the data characteristics of local devices for local optimization. With the varying number of stragglers, the results are reported in Fig. 3(d), showing that as the number of stragglers increases, the model performance decreases. However, even in the case of 40% being stragglers (i.e., $S_i = 2, S_{i,j} = 2$), HieAvg can still achieve an accuracy of 0.74.

We then investigate the performance of HieAvg with more heterogeneity involved, including different data distributions and inconsistent numbers of local devices at edge servers. For varying data distributions, We adjust the number of image classes in MNIST that each local device holds. For example, non_IID_1 means that each local device has at most 1 class of images. The results are presented in Fig. 4(a), which indicates that when the data distribution is more unbalanced, the model performance is worse. For inconsistent numbers of local devices connected to each edge server, we aim to test the aggregation effectiveness of HieAvg. The results in Fig. 4(b) show that the BHFL framework with HieAvg can still achieve better performance than the benchmark algorithms.

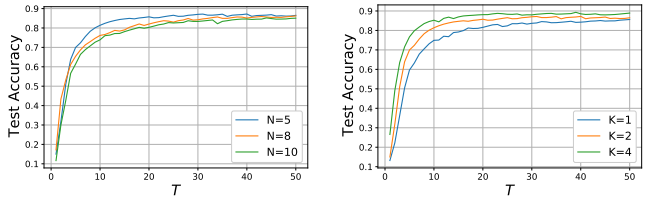
Finally, we test the convergence performance of HieAvg in mitigating temporary stragglers in only one layer, i.e., the local devices or edge servers. From Fig. 5(a) and Fig. 6(a), we can see HieAvg performs better in both cases compared to other algorithms. Furthermore, by varying the values of J, N, and K, we can find that smaller J and N, as well as larger K result in higher accuracy, which is consistent with the results in Fig. 3. These experimental results indicate that HieAvg can also efficiently handle only local device stragglers or only edge server stragglers.

2) Evaluation of Latency: First, we calculate the computing latency and communication latency on Raspberry Pi and EC2, and the averaged results of three Raspberry Pis are shown in Fig. 7(a). We can see that the more images on a local device, the higher the latency. This is because the time spent to process more data samples in the local training process will increase. In our basic setting, there are five edge servers and five local devices for each server, so each local device has 2,400 images, and thus, the corresponding latency is around 1.67s. The size of our employed CNN model updates is about 20KB, and the averaged transmission time between Raspberry Pi and EC2 is about 0.51s in the ideal scenario. Inspired by [10], we can assume that the latency among edge servers is 0.05s. These parameters are used to solve the latency optimization problem. The latency of Raft-based blockchain consensus will directly influence the optimal value of K^*, and the detailed results are illustrated in Fig. 7(b), which shows that the longer the consensus latency, the more the optimal edge aggregation rounds. Thus, we may adjust K to offset the influence of blockchain consensus latency on the overall latency.



(a) Accuracy with Different Algorithms.

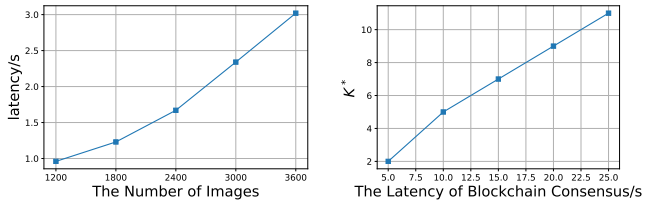
(b) Accuracy with Different J.



(c) Accuracy with Different N.

(d) Accuracy with Different K.

Fig. 6. Test Accuracy with Only Edge Server Stragglers.



(a) Latency vs. Data Size.

(b) K^* vs. Consensus Latency

Fig. 7. Evaluation Results of Latency Optimization.

VII. Related Work

Recently, there is an increasing number of studies on HFL. Lim *et al.* [7] propose an HFL framework to reduce node failures and device dropout, and design the resource allocation and incentive mechanisms to improve the learning efficiency based on game theory. Liu *et al.* [10] propose a client-edge-cloud HFL framework running with the HierFAVG aggregation algorithm and demonstrate that communication efficiency can be improved by introducing the hierarchical architecture in FL. Wang *et al.* [9] provide theoretical analysis about the convergence of HFL based on Stochastic Gradient Decent (SGD) and emphasize the importance of local aggregation before global aggregation. In [8], the focus is on protecting participants' privacy in HFL with flexible and decentralized control. Newest advancements have pushed HFL towards greater scalability and novel knowledge sharing paradigms. A more fundamental shift is seen in framework Hierarchical Knowledge Structuring (HKS) [41] which aggregates client-side *logits* and organizes them into a multi-granularity "knowledge codebook" via hierarchical clustering, allowing clients to distill knowledge at different levels of abstraction to better balance personalization and generalization.

With the emergence of blockchain technology, researchers propose the blockchain-based federated learning framework to address the challenges of FL, such as the single point of failure, incentive, and privacy preservation [17], [18]. There are also some studies applying blockchain in HFL. In [20], HFL participants are fragmented into multiple microchains to guarantee security and privacy for large-scale IoT intelligence. In [21], blockchain is used to verify the model updates from edge servers. Nguyen *et al.* [22] design a resource allocation mechanism among local devices to assist the latency optimization of BHFL. In this evolving landscape, our framework's choice of a lightweight Raft-based consensus is a deliberate design trade-off, prioritizing minimal latency for performance-critical applications over the feature-rich but potentially more latent smart contract-based ecosystems.

As for the challenges of stragglers, related research can be classified into three categories: *coded federated learning (CFL)*, *delayed gradient*, and *straggler-aware adaptation*. CFL is proposed in [24] to speed up FL running the linear regression task; the basic idea is that local clients transmit coded data to the central server at the beginning of training so that the server can compute coded gradients to compensate for the missing gradients of stragglers. In [25], CodedFedL mitigates the impact of stragglers for linear and non-linear regression. Although CFL performs well in tolerating stragglers, it requires extra data transmission and computing, leading to risks of privacy leakage and excessive resource consumption; moreover, most CFL-based methods are model-dependent and thus hard to generalize to diverse deep models. For delayed gradient methods, AD-SGD is proposed to minimize the difference between delayed and optimal gradients [27]; Xu *et al.* [28] propose live gradient compensation to utilize one-step delayed gradients. However, such methods only handle stragglers with poor computing power where partially trained gradients are still available; if stragglers arise from network disconnections, the server may receive no update in that round. For straggler-aware adaptation approaches, the training process is adapted for lagging devices or missing submissions. A memory enhancement approach, MIFA [29], is proposed to solve the problem of stragglers by using their recently submitted updates to correct missing updates; but this approach can lead to significant estimation bias since it only relies on the latest updates which cannot accurately reflect the overall optimization trend. FLuID [30] aims at workload reduction by identifying "invariant neurons" whose activations change little during training and then applying "Invariant Dropout" to exclude them when constructing the sub-model for straggler devices . The server then sends a smaller sub-model to lagging clients, proactively lowering both computation and communication without relying on coded redundancy or delayed gradients. However, FLuID relies on correctly identifying invariant neurons, where noisy or non-stationary training can misclassify useful neurons and degrade accuracy, with round-specific sub-models increasing protocol complexity. These limitations motivate the design of HieAvg, which estimates missing weights from historical differences at the aggregator without altering client models or depending on their most recent updates.

Optimizing end-to-end latency is critical for practical FL deployment. A common strategy, which our work adopts, is to optimize algorithmic hyperparameters, such as our optimization of the edge aggregation rounds, K. Concurrently, recent research has expanded this scope towards a more holistic

approach. Shaon *et al.* [42] propose the PAFL framework for wireless multi-hop networks, formulating a joint optimization problem that minimizes latency by simultaneously tuning the transmit power and CPU frequency of client and relay nodes, as well as network routing paths. In [43], energy consumption is optimized, which is intrinsically linked to latency, by exploring techniques such as adaptive learning rates and communication sparsification. Our work on optimizing K is complementary to these fine-grained, lower-layer approaches, as the optimal value of K is dependent on the per-round latency that these techniques aim to minimize.

To overcome the above shortcomings in the existing methods, we propose a novel aggregation method, HieAvg. It can be easily applied to more common cases of FL with non-IID data and even non-convex loss functions to solve the problem of stragglers in a cost-efficient manner.

VIII. Conclusion

In this paper, we propose a decentralized BHFL framework and design a novel aggregation algorithm HieAvg to ensure the convergence of BHFL even when there are stragglers in both local devices and edge servers, where the data is non-IID and the loss function can be non-convex. We also optimize the overall latency of BHFL by jointly considering the requirement of global model convergence and blockchain consensus latency. Theoretical analysis for the convergence of HieAvg is provided and extensive experiments are conducted to demonstrate the validity and superiority of our proposed schemes. In the future, we would like to design incentive and privacy protection mechanisms to further improve the performance of BHFL.

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 *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[4] N. Tabassum, M. Ahmed, N. J. Shorna, U. R. Sowad, M. Mejbah, and H. Haque, "Depression detection through smartphone sensing: A federated learning approach." *International Journal of Interactive Mobile Technologies*, vol. 17, no. 1, 2023.

[5] M. Fu, Y. Shi, and Y. Zhou, "Federated learning via unmanned aerial vehicle," *IEEE Transactions on Wireless Communications*, 2023.

[6] S. M. Azimi-Abarghouyi and V. Fodor, "Scalable hierarchical over-the-air federated learning," *IEEE Transactions on Wireless Communications*, 2024.

[7] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 536–550, 2021.

[8] A. Wainakh, A. S. Guinea, T. Grube, and M. Mühlhäuser, "Enhancing privacy via hierarchical federated learning," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2020, pp. 344–347.

[9] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Demystifying why local aggregation helps: Convergence analysis of hierarchical sgd," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[10] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[11] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[12] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[13] Z. Wang, Q. Hu, and X. Zou, "Can we trust the similarity measurement in federated learning?" *arXiv preprint arXiv:2311.03369*, 2023.

[14] G. Pichler, M. Romanelli, L. R. Vega, and P. Piantanida, "Perfectly accurate membership inference by a dishonest central server in federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[16] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[17] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," *arXiv preprint arXiv:1808.03949*, 2018.

[18] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[19] L. Chen, D. Zhao, L. Tao, K. Wang, S. Qiao, X. Zeng, and C. W. Tan, "A credible and fair federated learning framework based on blockchain," *IEEE Transactions on Artificial Intelligence*, 2024.

[20] R. Xu and Y. Chen, "μdfl: A secure microchained decentralized federated learning fabric atop iot networks," *IEEE Transactions on Network and Service Management*, 2022.

[21] P. Zhang, Y. Hong, N. Kumar, M. Alazab, M. D. Alshehri, and C. Jiang, "Bc-edgefl: A defensive transmission model based on blockchain-assisted reinforced federated learning in iiot environment," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3551–3561, 2021.

[22] D. C. Nguyen, S. Hosseinalipour, D. J. Love, P. N. Pathirana, and C. G. Brinton, "Latency optimization for blockchain-empowered federated learning in multi-server edge computing," *arXiv preprint arXiv:2203.09670*, 2022.

[23] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[24] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded federated learning," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[25] S. Prakash, S. Dhakal, M. Akdeniz, A. S. Avestimehr, and N. Himayat, "Coded computing for federated learning at the edge," *arXiv preprint arXiv:2007.03273*, 2020.

[26] R. Schlegel, S. Kumar, E. Rosnes *et al.*, "Codedpaddedfl and codedsecagg: Straggler mitigation and secure aggregation in federated learning," *arXiv preprint arXiv:2112.08909*, 2021.

[27] X. Li, Z. Qu, B. Tang, and Z. Lu, "Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients," *arXiv preprint arXiv:2102.06329*, 2021.

[28] J. Xu, S.-L. Huang, L. Song, and T. Lan, "Live gradient compensation for evading stragglers in distributed learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[29] X. Gu, K. Huang, J. Zhang, and L. Huang, "Fast federated learning in the presence of arbitrary device unavailability," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 052–12 064, 2021.

[30] D. M. Irene Wang, Prashant J. Nair, "Fluid: Mitigating stragglers in federated learning using invariant dropout," *arXiv preprint arXiv:2307.02623*, 2023.

[31] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 172–181, 2019.

[32] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 643–673.

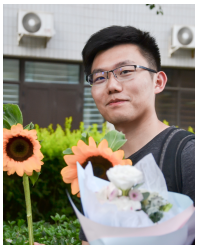
[33] Z. Wang and Q. Hu, "Blockchain-based federated learning: A comprehensive survey," *arXiv preprint arXiv:2110.02182*, 2021.

[34] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, 2014, pp. 305–319.
[35] X. Liu, Z. Zhong, Y. Zhou, D. Wu, X. Chen, M. Chen, and Q. Z. Sheng, "Accelerating federated learning via parallel servers: A theoretically guaranteed approach," *IEEE/ACM Transactions on Networking*, 2022.
[36] S. U. Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.
[37] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.
[38] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
[39] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
[40] C. J. B. Yann LeCun, Corinna Cortes, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.
[41] A. M. A. Wai Fong Tam, Qilei Li, "Hierarchical knowledge structuring for effective federated learning in heterogeneous environments," *arXiv preprint arXiv:2504.03505*, 2025.
[42] D. C. N. Shaba Shaon, Van-Dinh Nguyen, "Latency optimization for wireless federated learning in multihop networks," *arXiv preprint arXiv:2506.12081*, 2025.
[43] A. James, "Energy optimization techniques for federated learning on edge devices," 03 2025.

Zhilin Wang received his Ph.D. degree in Computer Science from Purdue University. His research interests include federated learning, security & privacy, distributed optimization, and blockchain. He's actively contributed as a reviewer for prestigious academic journals and conferences such as IEEE TPDS, IEEE IoTJ, Elsevier JNCA, IEEE TCCN, and IEEE ICC.

Xiangdong Hu received the B.Eng degree in Computer Science from the NingboTech University, Ningbo, China, in 2024. He is currently pursuing his Ph.D. degree in Computer Science at Georgia State University. He was actively involved in various programming competitions, such as ICPC, CCPC, and CCCC. His research interests include AI security & privacy, Multi Large Language Models, and federated learning.

Qin Hu received her Ph.D. degree in Computer Science from the George Washington University in 2019. She is currently an Assistant Professor with the Department of Computer Science, Georgia State University. She has served on the Editorial Board of two journals, the Guest Editor for multiple journals, the TPC/Publicity Co-chair for several workshops, and the TPC Member for several conferences. Her research interests include wireless and mobile security, edge computing, blockchain, and federated learning.



Minghui Xu received the BS degree in Physics from the Beijing Normal University, Beijing, China, in 2018, and the PhD degree in Computer Science from The George Washington University, Washington DC, USA, in 2021. He is currently a Professor in the School of Computer Science and Technology, Shandong University, China. His current research focuses on blockchain, distributed computing, and quantum computing.

Zehui Xiong is currently a Professor in the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, United Kingdom. Prior to that, he was a researcher with Alibaba-NTU Joint Research Institute, Singapore. He received the PhD degree in Nanyang Technological University, Singapore. He was the visiting scholar at Princeton University and University of Waterloo. His research interests include wireless communications, network games and economics, blockchain, and edge intelligence. He has published more than 140 research papers in leading journals and flagship conferences and many of them are ESI Highly Cited Papers. He has won over 10 Best Paper Awards in international conferences and is listed in the World's Top 2% Scientists identified by Stanford University. He is now serving as the editor or guest editor for many leading journals including IEEE JSAC, TVT, IoTJ, TCCN, TNSE, ISJ, JAS.