Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.053462

ARTICLE

FedAdaSS: Federated Learning with Adaptive Parameter Server Selection Based on Elastic Cloud Resources

Yuwei Xu, Baokang Zhao*, Huan Zhou and Jinshu Su

School of Computer, National University of Defense Technology, Changsha, 410000, China

*Corresponding Author: Baokang Zhao. Email: bkzhao@nudt.edu.cn

Received: 30 April 2024 Accepted: 24 June 2024 Published: 20 August 2024

ABSTRACT

The rapid expansion of artificial intelligence (AI) applications has raised significant concerns about user privacy, prompting the development of privacy-preserving machine learning (ML) paradigms such as federated learning (FL). FL enables the distributed training of ML models, keeping data on local devices and thus addressing the privacy concerns of users. However, challenges arise from the heterogeneous nature of mobile client devices, partial engagement of training, and non-independent identically distributed (non-IID) data distribution, leading to performance degradation and optimization objective bias in FL training. With the development of 5G/6G networks and the integration of cloud computing edge computing resources, globally distributed cloud computing resources can be effectively utilized to optimize the FL process. Through the specific parameters of the server through the selection mechanism, it does not increase the monetary cost and reduces the network latency overhead, but also balances the objectives of communication optimization and low engagement mitigation that cannot be achieved simultaneously in a single-server framework of existing works. In this paper, we propose the FedAdaSS algorithm, an adaptive parameter server selection mechanism designed to optimize the training efficiency in each round of FL training by selecting the most appropriate server as the parameter server. Our approach leverages the flexibility of cloud resource computing power, and allows organizers to strategically select servers for data broadcasting and aggregation, thus improving training performance while maintaining cost efficiency. The FedAdaSS algorithm estimates the utility of client systems and servers and incorporates an adaptive random reshuffling strategy that selects the optimal server in each round of the training process. Theoretical analysis confirms the convergence of FedAdaSS under strong convexity and L-smooth assumptions, and comparative experiments within the FLSim framework demonstrate a reduction in training round-to-accuracy by 12%–20% compared to the Federated Averaging (FedAvg) with random reshuffling method under unique server. Furthermore, FedAdaSS effectively mitigates performance loss caused by low client engagement, reducing the loss indicator by 50%.

KEYWORDS

Machine learning systems; federated learning; server selection; artificial intelligence of things; non-IID data

1 Introduction

In recent years, artificial intelligence (AI) technology has made significant progress, and its application scope is constantly expanding, covering various fields such as smart homes, voice assistants, and intelligent decision-making. However, with the widespread promotion of AI applications, the issue of privacy protection has attracted widespread attention [1]. Many countries and regions have established corresponding laws and regulations to ensure user privacy, and strictly restrict the use and processing of data [2]. This undoubtedly poses a challenge to the development of artificial intelligence applications. In response to this concern, federated learning (FL) has gradually evolved into a privacy-preserving machine learning paradigm. In this paradigm, mobile devices and Internet of Things (IoT) clients distributed in different geographical regions can collaborate to train machine learning models while keeping their respective data on local devices. FL is widely used for a variety of tasks. Examples include speech recognition, handwriting recognition, and text prediction in the consumer Internet, as well as collaboration between medical, industrial [3], and communications [4] entities.

In FL training, there are numerous and widely distributed devices, coupled with a certain degree of heterogeneity in system performance, which leads to significant differences in the success probability of each client participating in training (hereinafter referred to as engagement). In addition, the client data participating in federated learning training varies in scale [5], and the data distribution also exhibits non-independent identically distributed (non-IID) characteristics [6]. These characteristics result in significant performance loss in federated learning, affecting round-to-accuracy performance, and there is an issue of optimization target deviation [7,8].

Existing FL optimization methods mainly aim to reduce communication costs and improve training efficiency by fine-tuning client selection strategies [9], local updates [10,11], and communication compression [12–16] strategies. However, these methods have not fully addressed the problem of low engagement. On the other hand, decentralized [17] and asynchronous methods [18–21] can increase engagement, but synchronization costs are high and convergence speed is slow.

The client selection mechanism has been widely studied and applied to improve training efficiency reduce communication overhead, and increase the robustness of FL training. The first is the system utility perspective. Client selection reduces communication pressure on the parameter server, reducing unnecessary communication and network load, especially in bandwidth-constrained environments. Mobile devices may have limited computing and storage resources. By selecting clients with sufficient resources for training, the training process can run smoothly without overburdening resource-constrained devices. The second is to improve the statistical performance of FL training. Client data may be heterogeneous, i.e., different clients may have different data characteristics. By selecting clients that represent the overall data distribution, a better-performing global model can be trained. At the same time, by selecting clients with high data quality and high computing power to participate in training, the convergence speed of the model can be accelerated.

With the development of 5G/6G networks and the integration of computing resources through cloud computing, edge computing and other methods, mobile user-oriented application development can design more flexible paradigms [22,23]. Based on the widespread distribution of cloud computing resources worldwide (as shown in Table 1), effective utilization of these resources can optimize the federated learning process. The parameter server of FL can be set and migrated by a specific server selection mechanism, which ensures that the overall usage cost does not increase. Selecting the optimal parameter server according to geographical distribution, network state, and computing resources can reduce the communication delay of FL training and reduce resource competition, which has been widely verified in general distributed tasks. On the other hand, we found that server selection can improve the FL training process from a statistical utility perspective. This is reflected in two aspects: first, it can extend the client selection strategy (co-selection), and second, it can be combined with client selection as an orthogonal method (server selection after client selection), thereby improving the overall client participation and the training efficiency of federated learning.

Table 1: The top 10 cloud service providers

#	Cloud service provider	Regions	Availability zones
1	Amazon web services (AWS)	33	105
2	Microsoft azure	64	126
3	Google cloud platform (GCP)	40	121
4	Alibaba cloud	30	89
5	Oracle cloud	48	58
6	IBM cloud	10	30
7	Tencent cloud	21	65
8	OVHcloud	17	37
9	DigitalOcean	9	15
10	Linode (Akamai)	20	20

We propose an adaptive server selection algorithm FedAdaSS, which differs from the traditional approach of having a unique fixed parameter server. We adopt an adaptive strategy to select the optimal server as the parameter server in each training round to optimize the training effect. The server selection mechanism can be incorporated into existing federated learning training algorithms as an orthogonal mechanism. In addition to improving FL from system performance as in classical distributed computing, we have also demonstrated that it can improve statistical performance while also expanding the client selection policy space.

Optimizing server selection is a challenging task. In theory, the optimal choice requires us to obtain global prior knowledge in advance. However, in the actual training process, we cannot know the current set of clients in advance. Relying solely on the optimal set of clients per round to select servers can reduce wall clock time, but it cannot effectively alleviate the problem of optimization target offset. To this end, this article proposes an adaptive random shuffle strategy that estimates the utility of the client system and server in each round and then approximately selects the nearest server to accelerate the training process.

After rigorous theoretical analysis, we have confirmed that the FedAdaSS algorithm converges under strong convexity and L-smooth assumptions. Within the FLSim framework, we compared FedAdaSS with the Federated Averaging with random reshuffling (FedAvg+RR) and confirmed that optimizing the server selection mechanism can reduce the training #round-to-accuracy by 12%–20%. Under communication optimization strategies such as small number of participants and the large number of local steps, this optimization method shows significant advantages. In addition, when comparing different engagement levels, FedAdaSS can effectively mitigate the performance loss caused by low engagement, reducing the loss indicator by 50%.

The key contributions of our work are the following:

- We propose a basic framework for optimizing FL with elastic cloud resources, and analyze the optimization objectives and trade-offs for server selection.

- We propose the FedAdaSS algorithm, which is an adaptive server selection algorithm with dynamic client random shuffling to overcome the problems of dynamic joining and low commitment of clients, thereby increasing the statistical performance, and theoretically analyzing its convergence.
- We evaluate the adaptability of server selection to common FL optimization methods and the performance improvement it brings through comparative experiments.

2 Related Work

2.1 Federated Learning

Federated Learning is a distributed machine learning approach that allows multiple participants (typically devices or organizations) to work together to train a global machine learning model while maintaining the privacy of their respective data [6]. The key benefit of this approach is that it eliminates the need to centralize data in a single location, thereby reducing the risk of data leakage and reducing the need for centralized storage and computational resources [10].

Due to the size of the population and the diversity of user data and devices in FL, each round of training runs on a customer terminal set (with hundreds of participants), typically requiring hundreds of rounds (of a few minutes each) and several days to complete. For example, in the case of the Gboard keyboard, Google conducted weeks of federated training on natural language processing (NLP) models on 1.5 million devices [6,24]. Due to the large number of devices wide geographical distribution, and certain heterogeneity of system performance, each client shows great differences in the success probability of participating in training. Furthermore, client data involved in federated learning training differ in size and have a non-IID distribution. These factors cause federation learning performance to suffer a large loss, affect round-to-accuracy performance, and cause optimization target bias. To address these issues during the cross-device FL training process, existing methods can be classified into server-centric methods that focus on reducing communication overhead, and client-centric methods that focus on improving user engagement.

2.2 System Performance Optimization in FL

In the general paradigm of cross-device FL, the unique parameter server broadcasts the model parameters in each training round and aggregates the model parameters returned by the clients. However, due to resource constraints, methods such as multiple local updates, client selection, and communication compression have been proposed to meet the requirements of the parameter server.

The goal of the local update policy is to reduce the frequency of communication and to utilize the client's computational resources as much as possible [25], where each device performs multiple local steps before passing its updates back to the central server. One representative method is the Federated Averaging algorithm [20], which is an adaption of local-update to parallel stochastic gradient descent (SGD). The client involved in the training of each round runs a certain number of SGD steps based on its local data and sends back the local updated parameters to the server, then the pseudo-gradient for the global model is aggregated from each returned update. Recently, such methods have attracted a lot of attention, both in terms of theoretical guarantees [11,26], as well as in terms of optimization for real-world scenarios [1,27–29].

To address network bandwidth constraints and client heterogeneity, FL client selection is an emerging topic. FL client selection determines which client devices are selected in each training round. Effective FL client selection schemes can significantly improve model accuracy, increase fairness, improve robustness, and reduce training overhead [9]. For example, the paper [30] proposed a client selection framework called Oort, which improves the time-to-accuracy performance of model training by prioritizing those clients that have the greatest benefit in improving model accuracy and can perform training quickly. In addition, Oort allows developers to specify data distribution requirements during model testing and improve test duration efficiency by carefully selecting clients that meet those requirements.

Communication compression in FL is also a strategy that reduces data transmission between client devices and the central server during the training process. Quantization is a popular method that reduces the precision of model updates, making their transmission size smaller [31,32]. Sketching, as introduced by [33], provides a more concise alternative by summarizing updates using compact data structures. This enables approximate, yet significantly smaller representations of the original data.

2.3 Framework Optimization in FL

The training time scheduled by the parameter server is difficult to match with the idle time of the device, and the effective engagement rate of the client is low. Therefore, new frameworks of FL such as asynchronous distributed training [34] as well as decentralized federated learning methods have also been widely studied.

For asynchronous FL, the server updates the global model whenever it receives a local update. The authors in [35] found empirically that the asynchronous approach is robust to participants joining in the middle of training rounds and when the federation includes participating devices with heterogeneous processing capabilities. A new asynchronous federation optimization algorithm was proposed in [36] to improve the flexibility and scalability of federation learning, where each newly received local update is adaptively weighted according to its staleness, which is defined as the difference between the current epoch and the iteration to which the received update belongs. Furthermore, the authors also prove the convergence guarantee for a restricted family of non-convex problems. However, the current hyperparameters of the FedAsync algorithm still need to be tuned to ensure convergence in different environments. As such, the algorithm is still unable to generalize to the dynamic computational constraints of heterogeneous devices. Synchronous FL is still the most commonly used method today due to the uncertain reliability of asynchronous FL [37].

Decentralized federated learning is another way to maximize the possibility of the participant, which distributes the aggregation of model parameters between the neighboring participants [21,38,39]. Decentralized Federated Learning uses a P2P communication method, it deals with a dynamic and heterogeneous topology where participants often change their location or role in the federation. A study compared the communication efficiency of decentralized algorithms, such as gossip learning, with centralized FL [21]. It found that the best gossip variant had comparable overall performance to the best centralized FL algorithm. Another challenge in decentralized FL is imbalance heterogeneity, replacement of weight averaging with mutual knowledge distillation [40] was proposed to tackle class imbalance.

2.4 Limitations of Current Works

Existing system performance optimization methods, whose core goal is to reduce communication overhead, mainly use multiple local updates, client selection, and communication compression methods, but these methods are assumed to have better results when all clients effectively complete training and all have a high probability of participation. In the actual system, the client, due to its task, can not do more than an effective synchronization to the free time for training, so the total training engagement is low, slowing down the total training time; on the other hand, due to the client's data distribution of the phenomenon of non-IID, low engagement leads to the intensification of this imbalance phenomenon, which makes the distribution of training data is different from the distribution of the objective function, resulting in the training objective bias.

The methods proposed to solve the low engagement problem, such as asynchronous training and P2P training, have not been widely used in practical systems due to their characteristics, longer convergence times, and higher communication costs. Therefore, to solve the above contradiction, we propose a federated learning framework based on elastic cloud server resources, which improves the participation of clients and extends the policy framework of client selection by providing server selection when client selection, and on this basis, we propose a server selection method that supports dynamic client RR, FedAdaSS, which proves that server selection can be used as an orthogonal method to existing state-of-the-art (SOTA) methods and provides statistical performance improvement.

3 Overview

In this paper, the standard formulation of Federated Learning is considered as a finite sum minimization problem:

$$\min_{x\in\mathbb{R}^d}\left[f(x)\triangleq\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}f_c(x)\right], \tag{1}$$

where $\mathcal{C}$ is the client set. $f_c(x) = \mathbb{E}_{\xi\sim\mathcal{D}_c}[l(x;\xi)]$ corresponds to the average local loss of the current model parameterized by $x \in \mathbb{R}^d$ over the training data $\mathcal{D}_c$ located on client c. In the real scenario, the participants are mainly geographically distributed and have numerous mobile devices. When these devices participate in training, it is difficult to ensure that each mobile client can continuously and stably participate in the training process due to the variety of usage scenarios and user behaviors.

We assume that the FL training organizer has automatic control over the creation and destruction of server resources. Modern cloud platforms have their application programming interface (API) to support the automatization, and many interfaces can hide the differences between multiple clouds and provide a unified resource lifecycle control interface. During the training process, there is a global orchestrator role that can be performed by a separate server. The orchestrator selects the server responsible for the next round of training (parameter broadcasting and aggregation) and the participating clients, i.e., the orchestrator is responsible for maintaining and updating the FL state information.

Since servers in the same region on the cloud platform have relatively consistent network and computing resources. For simplicity, we assume that each region of one cloud has one selectable server, as shown in Fig. 1, and the selection described in this article includes the entire lifecycle process of creation, maintenance, and eventual destruction.

As shown in Fig. 2, the basic framework of FL includes server selection. The FL task is planned by the orchestrator as a whole, where the server set is all cloud servers available for selection (as mentioned earlier, including the entire lifecycle management), and the client set is all mobile clients participating in FL. This article mainly discusses learning and training tasks in a synchronous federation that involves multiple global rounds. In each round, the orchestrator selects the parameter server for that round, which is responsible for parameter distribution and aggregation. Then, some kind of client selection mechanism selects some clients to participate in the training. After the selected server distributes the model parameters to the clients, the clients use local data to perform several rounds of local steps, typically the SGD process. Then the parameter server collects the results returned by the clients, usually the clients that successfully returned within a certain time threshold, and calculates the weighted average of the returned weights as the result of this round. The above steps are repeated until the model meets a certain requirement, such as a fixed number of rounds, or the accuracy reaches a threshold.

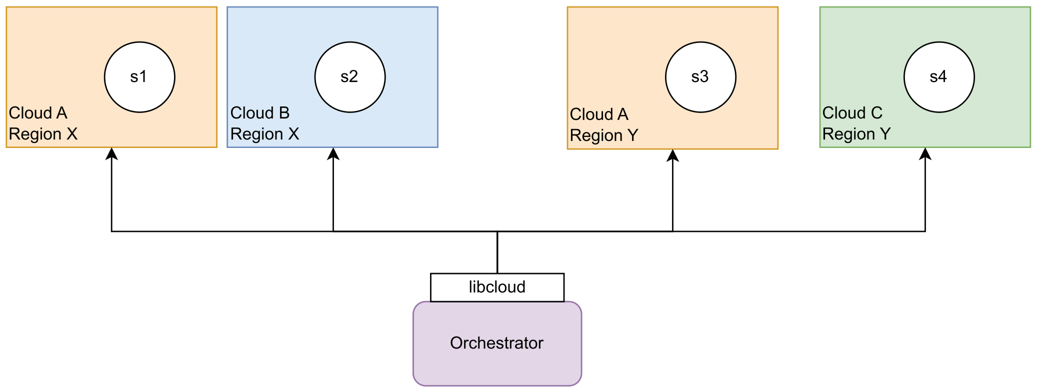

Figure 1: Elastic cloud resources controlled by orchestrator

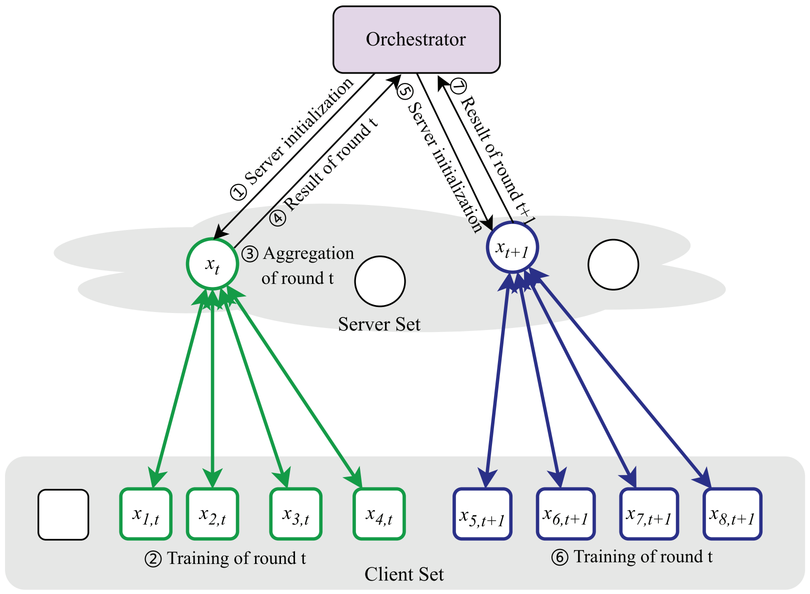

Figure 2: The architecture of federated learning with server selection

Due to factors such as location distance and network quality, the bandwidth between clients and servers in different regions is different, the success rate of participating in training is also different, and the distribution of server training success rate among different clients is also different. We can assume that the success rate of clients under the same server is approximately a power-law distribution. Server selection based on optimized latency can improve the wall time of FL training. At the same time, server selection can cooperate with client selection, which not only expands the policy space of client selection but also further improves the effect of client selection with the existing state of the art, thus statistically improving training efficiency.

4 Server Selection Method in Federated Learning

4.1 General Optimization Objectives of FL

In the system design process of federated learning, the first metric to focus on is time-to-accuracy, which has two main influencing factors, (i) System utility factors: the training process includes the actual available performance of both client-side and server-side participants, and the quality of network among them. These physical metrics will affect the actual running time of the system. (ii) Algorithm and data factors: by adjusting the hyper-parameters in the training process, as well as the use of client selection algorithms, as a way to affect the round-to-accuracy performance of the training, to further reduce the training time in the same environment and system setup.

To achieve the above objectives. From a system optimization perspective, the main constraints are the unreliability of computing resources of clients and restricted network resources. Client selection methods are firstly involved to ensure that the training can be carried out continuously, and secondly, to reduce the communication overhead, methods such as *local update* and *communication compression* are further used.

From the client's perspective, the primary constraint is the lack of engagement; due to the uncontrollable synchronization of training time and the passive selection mechanism from the parameter server, the client's arithmetic availability and training time cannot occur simultaneously, resulting in the client arithmetic that participates in federated learning to be a very small fraction of the total arithmetic. In addition, due to possible single-point-of-failure issues and privacy considerations, the use of distributed algorithms can provide better hence protection. Therefore, clients expect to use asynchronous or decentralized methods to achieve this goal.

4.2 Optimal Server Selection Requirements

All servers that can be created by cloud platforms in different regions are taken as a server set, and the orchestrator selects one server as a parameter server, which is responsible for parameter broadcasting, collecting local update results from different clients, and calculating the final result of this round. To illustrate the importance of server selection, we discuss the requirements for optimal server selection in a state-of-the-art client selection scenario.

Concerning all data samples for training planning, Random Reshuffling of the training data at the beginning of each epoch is a successful technique for optimizing the empirical risk minimization process in standard SGD process analysis. In FL training, the current state-of-the-art approach also introduces the RR process, but the difference is that this process is defined on the client arrangement, as shown in Fig. 3, which requires that in a meta-epoch consisting of multiple rounds of global training, each client is selected at most once, and in the next meta-epoch all clients are randomly reshuffled to enter training.

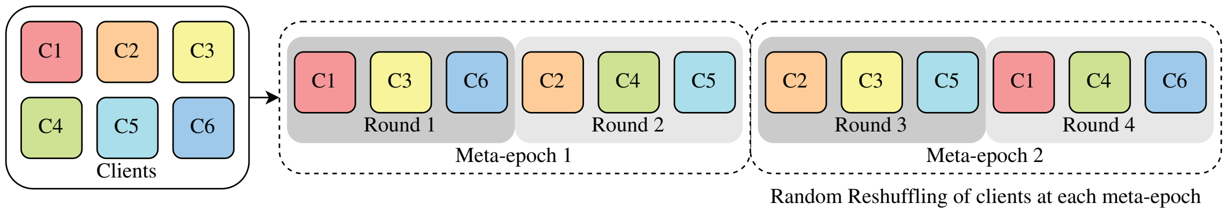

Figure 3: Random reshuffling of clients in FL training scheduling

To further increase the round-to-accuracy performance, we can utilize the *importance sampling* method to prioritize high-utility clients [30,41], where the importance of each data point's contribution to training can be quantified using the L2-norm of the gradient. Assuming that each client x has a training sample set of B_x, the importance of each client can be defined as

$$U_x \triangleq |B_x| \sqrt{\frac{1}{|B_x|} \sum_{k \in B_X} \| \nabla f(k) \|^2}. \tag{2}$$

Optimal server selection can be quantified as selecting the most efficient server. After selecting clients by RR and quantifying the utility per client, we can compute the utility of each server. Due to the different geographic distributions of clients and servers, the network states of different client-server combinations are inconsistent. Assuming that the success rate of the training process between different clients and servers is defined as $P(x, y)$, and assuming that the set of clients in each round is C_t, and the selected server corresponding to each round is y_r, the final utility of a server and client co-selection in one meta-epoch is defined as the expectation the importance value sum of all client participated in training:

$$\text{Util} \triangleq \sum_{t=1}^{R} \sum_{x \in C_r} P(x, y) U_x, \quad \bigcup_{t=1}^{R} C_t = \mathcal{C} \text{ and } C_i \cap C_j = \emptyset, \forall i \neq j. \tag{3}$$

The client sets in each round of one meta-epoch do not intersect, and the union of the client sets C_t in each round covers all clients.

The optimal server selection mechanism should be the one that maximizes the utility in Eq. (3). In addition, the timing of the federated learning training provided by the parameter server from the client's perspective should be matched as closely as possible with the time available for the client's arithmetic, thus minimizing the partial participant problem. This requirement is equivalent to maximizing the number of FL client participants, consistent with the optimization goal.

Algorithm 1: FedAdaSS: Adaptive Server Selection in Federated Learning

```
1:  Input: client step size $\eta > 0$, client set that could join the training $\mathcal{C}$, Server set $\mathcal{S}$, sample size $K$,
     initials model $x_0 \in \mathbb{R}^d$, number of rounds $T$
2:  Procedure at Orchestrator
3:  $C_{ignored} \leftarrow \emptyset$
4:  for each round $t = 0, 1, \ldots, T$ do
5:      $C_{active} \leftarrow$ UpdateActiveClientSet($\mathcal{C}$)          ▷Adaptive clients random reshuffling
6:      if $\left|C_{active} \backslash C_{ignored}\right| < K$ then
7:          $C_{ignored} \leftarrow \emptyset$
8:      end if
9:      select a cohort $C_t \subset C_{active} \backslash C_{ignored}$, $|C_t| = K$ uniformly at random
10:     $s_t \leftarrow$ ServerSelect($C_t, \mathcal{S}$)
11:     initialize server $s_t$ with current $x_t$ and $C_t$
12:     get $x_{t+1}$ from procedure of selected server and clients
13:     $C_{ignored} \leftarrow C_{ignored} \cup C_t$
14: end for
15: functionSERVERSELECT($C_t$, $\mathcal{S}$)
```

(Continued)

Algorithm 1 (continued)

```
16:         for all $s \in \mathcal{S}$ do
17:             $Util(s) \leftarrow \sum_{c \in C_t} P(c,s)\,Util(c)$
18:         end for
19:         $s_{t,*} \leftarrow \arg\max_{s \in \mathcal{S}} Util(s)$        ▷Server selection with utility estimation
20:         return $s_{t,*}$
21:     end function
22:     Procedure of Selected Server and Clients at Round t
23:     broadcast $x_t$ to all clients from sever $s_t$
24:     for each $c \in C_t$ in parallel do
25:         compute $x_t^c$ with local training
26:         $Util(c) \leftarrow |B_x|\sqrt{\frac{1}{|B_x|}\sum_{k \in B_x}\|\nabla f(k)\|^2}$
27:         send $\Delta_t^c \leftarrow x_t^c - x_t$ and $Util(c)$ to server $s_t$
28:     end for
29:     $x_{t+1} \leftarrow x_t - \frac{\eta_g}{|C_t|}\sum_{c \in C_t}\Delta_t^c$
30:     push the $x_{t+1}$ and $Util(c)$ of all clients in $C_t$ to orchestrator
```

4.3 *Trade-Offs and Adaptive Server Selection Method*

4.3.1 *Adaptive Clients Random Reshuffling*

In cross-device FL deployment, client joining and exiting are unpredictable. We cannot obtain the client set in advance, nor can we guarantee that it will remain unchanged during training. As a result, it is not possible to reshuffle all clients beforehand and select the optimal server in advance. There is no strict meta-epoch boundary in this case, so we cannot obtain a meta-epoch client permutation in advance according to the RR defined on the static client set. Therefore, we introduce an adaptive client random reshuffling method in FedAdaSS.

As shown in Fig. 4, before each round of training, FedAdaSS will update the current set of clients, remove the clients selected in previous rounds, and randomly select clients from the remaining set for this round. Server selection will also depend on the results of the current round of client selection. We approximate that choosing the optimal server in each round achieves a global relative superiority. The algorithm implements random reshuffle by dynamically maintaining the remaining unselected clients. trained clients are added to the set $C_{ignored}$ at the end of each round, and K clients are randomly selected from the set $C_{active} \backslash C_{ignored}$ at the beginning of the round. when $C_{active} \backslash C_{ignored}$ is empty, it can be approximated as the end of a global epoch, and reset $C_{ignored}$ to the empty set and continue the above steps.

4.3.2 *Server Selection with Utility Estimation*

For the utility computation of clients and sever, $Util(c)$ takes the value from the gradient obtained in the current round, and we can not get the utility of each client in advance before choosing the current server and training. However, due to the adaptive reshuffling process, each client only participates in training once per epoch, while the sequential relationship between client utilities, due to the positive correlation between utility and data size, does not change significantly over the multiple training epochs. Thus, our updated client utility for each epoch can be used as an estimate of that client's utility for the next epoch of server selection.

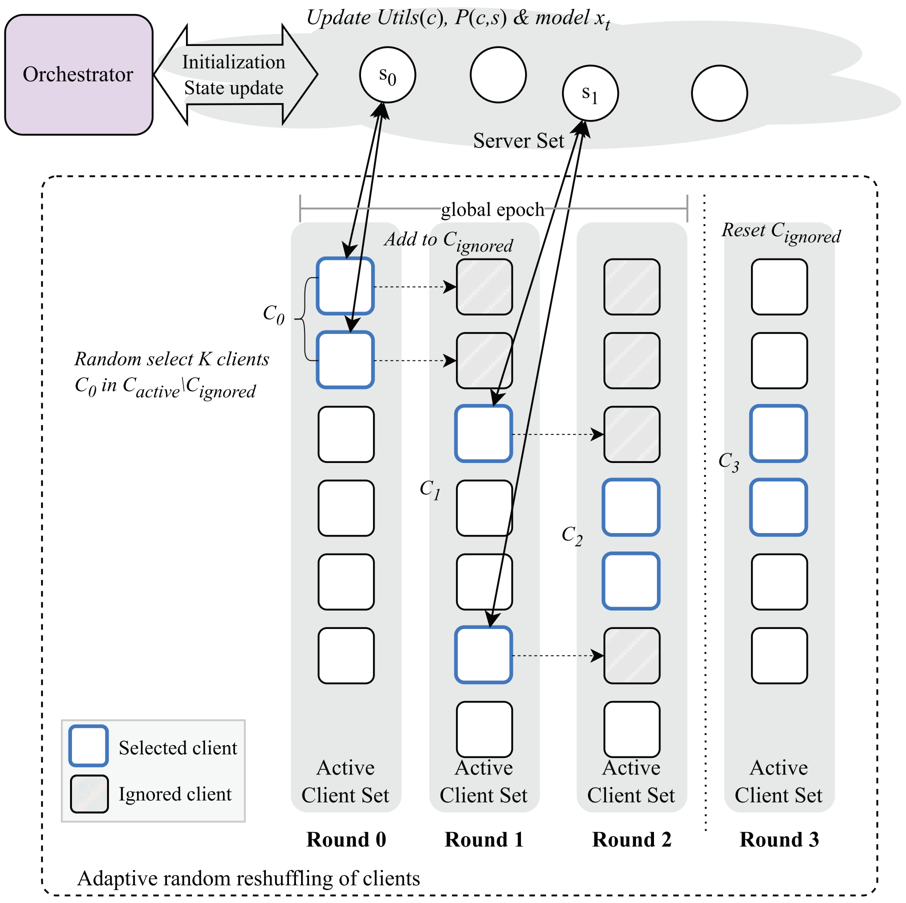

Figure 4: The main loop and adaptive random reshuffling of clients in FedAdaSS

4.3.3 Division of Roles and Cooperation in the Procedure

After introducing the two main computational processes in the Algorithm 1, we will introduce the interaction process in the FedAdaSS algorithm. As described in Section 3, we split the roles in FedAdaSS into orchestrator, parameter server, and client.

The orchestrator is responsible for maintaining the client collection and the server collection. The server collection is obtained during initialization, and the currently available clients are updated before each round of client selection. The orchestrator arranges each round of FL training in turn. In a given round t, the orchestrator first computes and selects the server s_t and clients C_t of the current round, and initializes the server s_t using the result and the parameters of the model of the previous round. Its communication overhead is not large, corresponding to a constant multiple (maybe including the intermediate state of optimization) of the communication between the server and one client. Then, FedAvg is executed between the server and the client, and the orchestrator retrieves the training results for that round from the server.

The server is initialized by the orchestrator through the cloud platform API, and the server obtains the client set of the current round and the current model parameters during initialization. Thereafter, the server process is similar to that in generic FL, where the server broadcasts the current model parameters x_t to all clients in C_t. Each client performs multiple local steps, returning resulting model parameters C_t^c and client utility $Util(c)$. The server then computes $x_{t+1} \leftarrow x_t - \eta_g/|C_t| \cdot \sum_{c\in C_t} \Delta_t^c$ and returns all results to the orchestrator. The orchestrator then ends the life cycle of the server. The system implementation and training method of the client are consistent with the general federated learning process, and there is no need to provide redundant additional descriptions here.

5 Convergence Analysis

5.1 Preliminaries

The loss function of client c is composed of single losses $f_c^j(x)$ where j corresponds to j-th data points with the current model parameterized by x. We assume that client c has access to an oracle that, when given input (j, x), returns the gradients $\nabla f_c^j(x)$ as an output. We donate $[l] \triangleq \{1, 2, \ldots, l\}$ for any $l \in \mathbb{N}$. To show the convergence of our methods, we adopt the standard assumptions in convex optimization, which are commonly used in the previous works [42].

Assumption 1. The functions $f_c^j(x)$ are L_c-smooth for all $c \in \mathcal{C}, j \in [|\mathcal{D}_c|]$; i.e., there is an $L_c > 0$ such that for any c, j, x, y

$$\|\nabla f_c^j(x) - f_c^j(x)\| \leq L_c\|x - y\|. \tag{4}$$

Assumption 2. The functions $f_c^j(x)$ are μ_c-strongly convex for all $c \in \mathcal{C}, j \in [|\mathcal{D}_c|]$; i.e., there is an $\mu_c > 0$ such that for any c, j, x, y

$$\langle \nabla f_c^j(x), y - x\rangle \leq -\left(f_c^j(x) - f_c^j(y) + \frac{\mu_c}{2}\|x - y\|\right). \tag{5}$$

FedAdaSS introduces the RR of clients, and analysis of existing RR shows that it cannot converge to a certain exact value. Through analysis of the shuffling radius upper bound brought by RR, it can be proved that it converges to a certain neighborhood related to σ^2_{rad}. We use the notions of *shuffling variance*, introduced by Mishchenko et al. [43] for the analysis, Given a static stepsize $\eta > 0$ and a permutation π of $[n]$, the intermediate points $x_*^1, x_*^2, \ldots, x_*^n$ is

$$x_*^i \triangleq x_* - \eta \sum_{j=1}^{i} \nabla f_{\pi_j}(x_*), \quad i = 1, \ldots, n. \tag{6}$$

The shuffling radius $\sigma^2_{\text{rad}}(\eta)$ is

$$\sigma^2_{\text{rad}}(\eta) \triangleq \max_{i\in[n]} \left[\frac{1}{\eta^2}\mathbb{E}_\pi\left[D_{f_{\pi_i}}(x_*^i, x_*)\right]\right], \tag{7}$$

where $D_f(x, y) = f(x) - f(y) - \langle \nabla f(y), x - y\rangle$ is the Bregman divergence associated with f. Due to our use of dynamic client rearrangement, the corresponding shuffling radius is equivalent to the RR process of each data point. With the combination of L-smooth of f and the definition of x_*^i, we could get the bound of shuffling radius (for a detailed proof, see Appendix A).

Lemma 1. (Shuffling radius bounds). For all $\eta > 0$, we get the upper bound of shuffling radius σ^2_{rad}.

$$\sigma^2_{\text{rad}} \leq \frac{nL_{\max}}{2}\left(n\|\nabla f(x_*)\|^2 + \frac{1}{2}\sigma_*^2\right), \tag{8}$$

where σ_*^2 is the gradient variance at the optimum $\sigma_*^2 \triangleq \frac{1}{n}\sum_{i\in[n]}\|\nabla f_i(x_*) - \nabla f(x_*)\|^2$. Since f is $L_{\max}$-smooth and $\nabla f(x_*) = 0$, we get that

$$\sigma_{\mathrm{rad}}^2 \leq \frac{nL_{\max}}{4}\sigma_*^2. \tag{9}$$

5.2 Convergence Guarantees

After defining the variance quantities and assuming that each f_c^j is μ-strongly convex which is commonly satisfied in machine learning applications as in l_2 regularized linear regression and l_2 regularized logistic regression, we can get the result that the optimization term decreases linearly. The exponential is the product of the number of data points of each client N, the number of communication rounds in each global epoch R, and the number of global epochs T. Since the cohort and data points on each client are sampled without replacement for each round, the statistical term scales proportionally to the squared step size η^2. The formal statement of the theorem follows.

Theorem 1. Assume that functions f_c^j are L_c-smooth and μ_c-strongly convex for each c and j. If $\eta \leq \frac{1}{L_{\max}}$, then the iterates generated by the Algorithm 1 satisfy

$$\mathbb{E}\left[\|x_T - x_*\|^2\right] \leq (1-\eta\mu)^{NRT}\|x_0 - x_*\|^2 + \frac{2\eta^2}{\mu}\sigma_{\mathrm{rad}}^2. \tag{10}$$

The results show the exponential rate of convergence to a neighborhood of size $\frac{2\eta^2}{\mu}\sigma_{\mathrm{rad}}^2$ and can be adjusted to accommodate dynamic client shuffling, with the only requisite modification being the alteration of the rates and convergence analysis. Detailed proof can be found in the Appendix B.

6 Evaluation

6.1 Experimental Methodology

To demonstrate that FedAdaSS can effectively provide both random reshuffling and partial participant mitigation in large-scale client scenarios, we perform experimental verification in addition to theoretical analysis. Due to the lack of large-scale client validation conditions, we use simulation experiments for validation and expand FLSim [44] to provide a server selection mechanism as the experimental framework, simulating 10 candidate servers in the experiment. We used the CIFAR10 dataset to simulate 100 clients and split the data into IID and non-IID methods for experiments. The non-IID group splits the sample size according to a power law distribution. To account for the heterogeneity of clients, as shown in Fig. 5, we generate the training success probability of each client to one server according to the power law distribution. In the comparative experiment, to demonstrate the performance of server selection under different network conditions, we set the average success rate of client generation to $p = 75\%$–95%.

6.2 Comparisons Results

To demonstrate the improvement of statistical utility in FL process by server selection and to enhance existing client selection methods, we choose the state-of-the-art method, i.e., client RR (FedAvg+RR), as a comparison. The overall results show that FedAdaSS can reduce the number of rounds to reach the same accuracy value (#round-to-accuracy) by approximately 12% to 20% under the same environmental parameters,

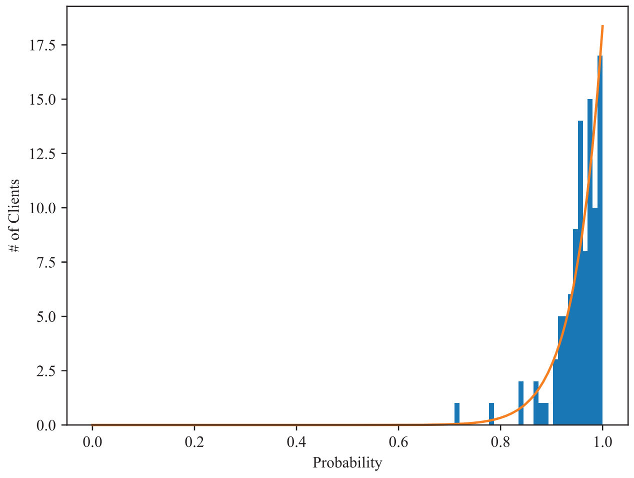

Figure 5: Distribution of probability of successful communication

Next, we will compare different environmental parameters (IID/non-IID, average success rate *p*, number of clients per round k, local steps N) to study the characteristics and causes of server selection performance improvement.

6.2.1 *Increasing Training Speed*

Fig. 6 compares the decline curves of the loss function for different numbers of participants in each training round. It can be seen that as the number of participants increases, the loss function decreases faster. This indicates that increasing the number of participants can reduce the #round-to-accuracy. Under the same k setting, it can be seen that FedAdaSS accelerates training speed by 5%–10%, which is more significant at low engagement levels.

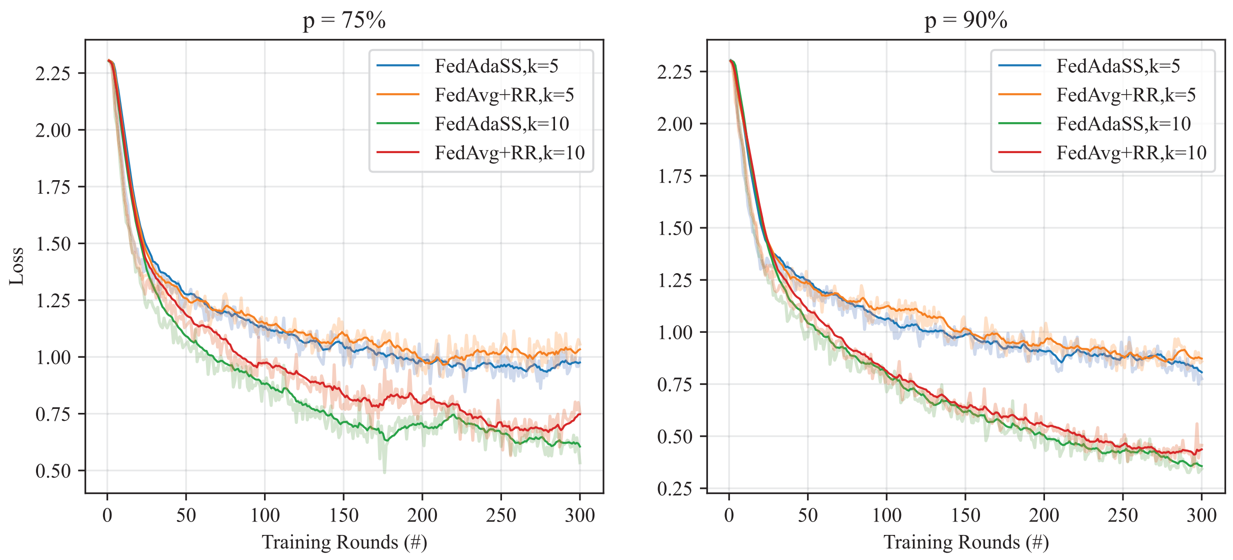

Figure 6: Training loss of FedAdaSS and FedAvg+RR

In addition, as shown in Fig. 7, under the setting of non-IID, a lower number of training clients has a greater impact on convergence. At this time, adding server selection can also accelerate convergence speed by 10%–20%. This improvement is more significant with the fewer number of clients and less engagement.

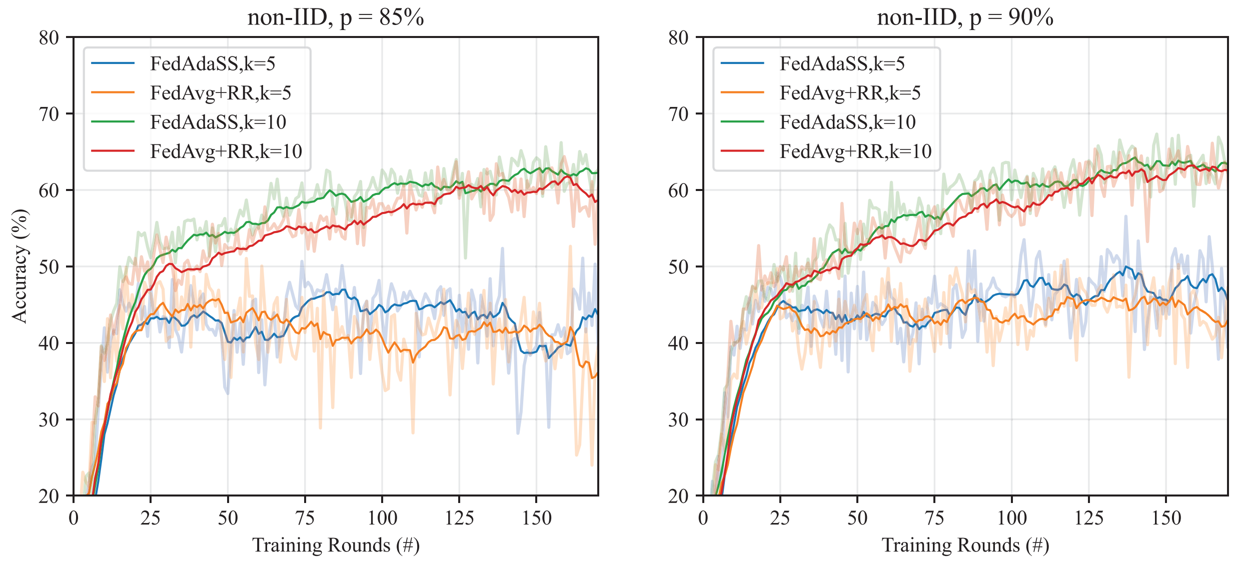

Figure 7: Accuracy of aggregation of each training round on non-IID data and with $p = 75\%$ and $p = 90\%$

6.2.2 *Mitigating Low Engagement of Clients*

Fig. 8 compares the accuracy training curves under the same machine learning parameter configuration but different communication success probability settings. It can be seen that compared to the FedAvg+RR method, the FedAdaSS method has a smaller negative impact on the decrease in engagement. Due to the dropout strategy, the actual number of client participants is the same under different methods. However, under the same method, the impact of low engagement may not have been added to the training, resulting in a reduction in effective training data. It can be seen that after joining the server selection, the decrease in accuracy under the same setting was alleviated by increasing the participation opportunities of these low-probability clients, resulting in a 3% increase in tie values. In addition, under the high local steps setting set to reduce communication overhead, the impact of low engagement on #round-to-accuracy is more pronounced, resulting in an increase of about 50% in #round-to-accuracy. At this point, using FedAdaSS can reduce this loss to about 20%.

In summary, FedAdaSS can firstly improve client engagement compared to the singular server method, thus increasing the number of clients involved in training per round, thus reducing rounds to accuracy. Second, FedAdaSS improves the fairness of each client's participation, allowing the random reshuffling process to cover a larger number of clients, making it possible to achieve a better gradient representation in a global epoch, and thus reducing round-to-accuracy. And since the number of local steps in FL is relatively large, adding RR can effectively improve the training performance. Therefore, it is demonstrated that FedAdaSS can be effectively combined with existing methods to accelerate the training process of federated learning.

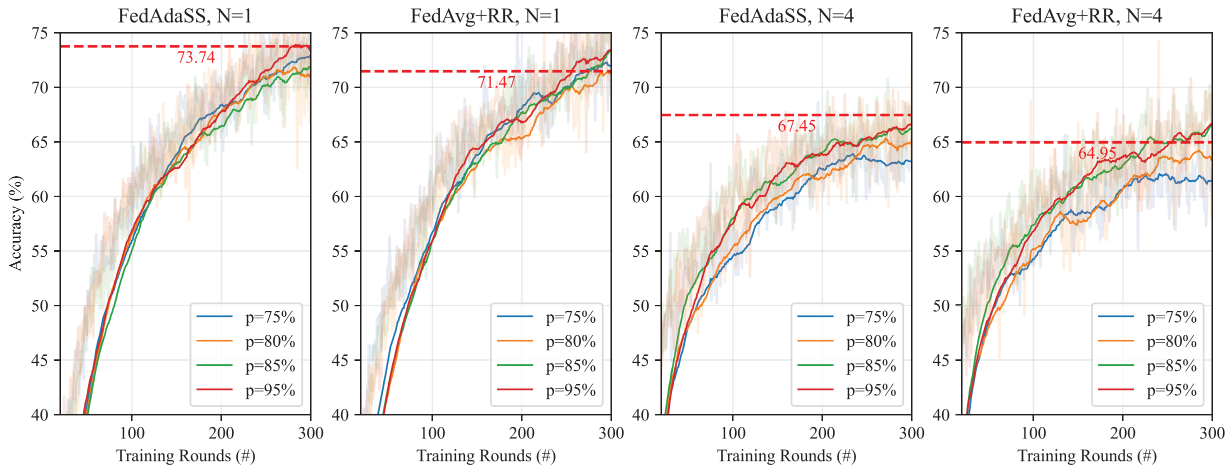

Figure 8: Final average accuracy of the training and round-to-accuracy with different settings of the probability of successful communication ($p = 75\%$–95%)

7 Conclusions

Federated learning training involves a large number of geographically distributed devices, and the heterogeneity of system performance leads to significant differences in clients' engagement levels. In addition, the client's data have different scales and distributions, which are not independent and identically distributed, resulting in significant performance loss in federated learning, affecting the round-to-accuracy performance, and there is an optimization target bias problem. We propose a basic framework for optimizing using optional computing power servers in federated learning and analyze the optimization objectives and trade-offs for server selection. A self-adaptive server selection algorithm FedAdaSS with random shuffling is proposed to overcome the dynamic joining and low engagement issues of clients, and its effectiveness is theoretically analyzed. Through comparative experiments, we demonstrate that FedAdaSS can reduce round to accuracy and alleviate performance losses caused by low engagement.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: This work was supported in part by the National Natural Science Foundation of China under Grant U22B2005, Grant 62372462.

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Yuwei Xu, Baokang Zhao; data collection: Huan Zhou, Baokang Zhao; analysis and interpretation of results: Yuwei Xu, Huan Zhou; draft manuscript preparation: Yuwei Xu, Baokang Zhao, Jinshu Su. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data used in the experimental process was generated using the FLSim simulator, which is perfectly reproducible following the instructions of Section 6.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Bao G, Guo P. Federated learning in cloud-edge collaborative architecture: key technologies, applications and challenges. J Cloud Comput. 2022;11:4.
2. Kotsehub N, Baughman M, Chard R, Hudson N, Patros P, Rana O, et al. FLoX: federated learning with FaaS at the edge. In: 2022 IEEE 18th International Conference on e-Science (e-Science); 2022; Salt Lake City, UT, USA: IEEE. p. 11–20.
3. Wang X, Garg S, Lin H, Hu J, Kaddoum G, Piran MJ, et al. Toward accurate anomaly detection in industrial internet of things using hierarchical federated learning. IEEE Internet Things J. 2022 May;9:7110–9. doi: 10.1109/JIOT.2021.3074382.
4. Wang X, Hu J, Lin H, Garg S, Kaddoum G, Piran MJ, et al. QoS and privacy-aware routing for 5G-enabled industrial internet of things: a federated reinforcement learning approach. IEEE Trans Ind Inform. 2022 Jun;18:4189–97. doi: 10.1109/TII.2021.3124848.
5. Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, et al. Towards federated learning at scale: system design. Proc Mach Learni Syst. 2019;1:374–88.
6. Banabilah S, Aloqaily M, Alsayed E, Malik N, Jararweh Y. Federated learning review: fundamentals, enabling technologies, and future applications. Inform Process Manag. 2022 Nov;59:103061. doi: 10.1016/j.ipm.2022.103061.
7. Sattler F, Wiedemann S, Muller KR, Samek W. Robust and communication-efficient federated learning from non-i.i.d. data. IEEE Trans Neural Netw Learn Syst. 2020 Sep;31:3400–13. doi: 10.1109/TNNLS.5962385.
8. Luo K, Li X, Lan Y, Gao M. GradMA: a gradient-memory-based accelerated federated learning with alleviated catastrophic forgetting. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2023; Vancouver, BC, Canada: IEEE. p. 3708–17.
9. Fu L, Zhang H, Gao G, Zhang M, Liu X. Client selection in federated learning: principles, challenges, and opportunities. IEEE Internet Things J. 2023;10:1
10. Gholami P, Seferoglu H. DIGEST: fast and communication efficient decentralized learning with local updates. IEEE Trans Mach Learn Commun Netw. 2024;1.
11. Glasgow M, Yuan H, Ma T. Sharp bounds for federated averaging (Local SGD) and continuous perspective. In: International Conference on Artificial Intelligence and Statistics; 2022; PMLR. p. 9050–90.
12. Yang HH, Liu Z, Quek TQS, Poor HV. Scheduling policies for federated learning in wireless networks. IEEE Trans Commun. 2020 Jan;68:317–33. doi: 10.1109/TCOMM.26.
13. Foley P, Sheller MJ, Edwards B, Pati S, Riviera W, Sharma M, et al. OpenFL: the open federated learning library. Phys Med Biol. 2022 Nov;67:214001. doi: 10.1088/1361-6560/ac97d9.
14. Oh K, Zhang M, Chandra A, Weissman J. Network cost-aware geo-distributed data analytics system. IEEE Trans Parallel Distrib Syst. 2022 Jun;33:1407–20. doi: 10.1109/TPDS.2021.3108893.
15. Chahoud M, Otoum S, Mourad A. On the feasibility of federated learning towards on-demand client deployment at the edge. Inf Process Manag. 2023 Jan;60:103150. doi: 10.1016/j.ipm.2022.103150.
16. Grudzień M, Malinovsky G, Richtárik P. Improving accelerated federated learning with compression and importance sampling. arXiv preprint arXiv:2306.03240, 2023. doi:10.48550/ARXIV.2306.03240.
17. Li Z, Lu J, Luo S, Zhu D, Shao Y, Li Y, et al. Towards effective clustered federated learning: a peer-to-peer framework with adaptive neighbor matching. IEEE Trans Big Data. 2022:1–16.
18. Guan Y, Liu X, Ren T, Niu J. Enabling communication-efficient federated learning via distributed compressed sensing. In: IEEE INFOCOM 2023-IEEE Conference on Computer Communications; 2023; New York City, NY, USA: IEEE. p. 1–10.
19. Zhang F, Liu X, Lin S, Wu G, Zhou X, Jiang J, et al. No one idles: Efficient heterogeneous federated learning with parallel edge and server computation. In: International Conference on Machine Learning; 2023; Honolulu, Hawaii, USA: PMLR. p. 41399–413.

20. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics; 2017; PMLR. p. 1273–82.

21. Beltrán ETM, Pérez MQ, Sánchez PMS, Bernal SL, Bovet G, Pérez MG, et al. Decentralized federated learning: fundamentals, state of the art, frameworks, trends, and challenges. IEEE Commun Surv Tutorials. 2023;25:2983–3013. doi: 10.1109/COMST.2023.3315746.

22. Barbarossa S, Sardellitti S, Lorenzo PD. Communicating while computing: distributed mobile cloud computing over 5G heterogeneous networks. IEEE Signal Process Mag. 2014 Nov;31:45–55. doi: 10.1109/MSP.2014.2334709.

23. Messaoud S, Bradai A, Ahmed OB, Quang PTA, Atri M, Hossain MS. Deep federated Q-learning-based network slicing for industrial IoT. IEEE Trans Ind Inform. 2021 Aug;17:5572–82. doi: 10.1109/TII.2020.3032165.

24. Hard A, Rao K, Mathews R, Ramaswamy S, Beaufays F, Augenstein S, et al. Federated learning for mobile keyboard prediction. arXiv preprint arXiv: 81103604. 2018 Nov.

25. Zhang T, Gao L, He C, Zhang M, Krishnamachari B, Avestimehr AS. Federated learning for the internet of things: applications, challenges, and opportunities. IEEE Internet of Things Magazine. 2022 Mar;5:24–9. doi: 10.1109/IOTM.004.2100182.

26. Li X, Huang K, Yang W, Wang S, Zhang Z. On the convergence of fedavg on non-iid data. arXiv preprint arXiv:190702189. 2019.

27. Lim WYB, Luong NC, Hoang DT, Jiao Y, Liang YC, Yang Q, et al. Federated learning in mobile edge networks: a comprehensive survey. IEEE Commun Surv Tutorials. 2020 Feb;22:2031–63. doi: 10.1109/COMST.9739.

28. Roy S, Chergui H, Sanabria-Russo L, Verikoukis C. A cloud native SLA-driven stochastic federated learning policy for 6G zero-touch network slicing. In: ICC 2022—IEEE International Conference on Communications; 2022; Seoul, Republic of Korea: IEEE. p. 4269–74.

29. Shi W, Zhou S, Niu Z, Jiang M, Geng L. Joint device scheduling and resource allocation for latency constrained wireless federated learning. IEEE Trans Wirel Commun. 2021 Jan;20:453–67. doi: 10.1109/TWC.7693.

30. Lai F, Zhu X, Madhyastha HV, Chowdhury M. Oort: efficient federated learning via guided participant selection. In: 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21); 2021; p. 19–35.

31. Reisizadeh A, Mokhtari A, Hassani H, Jadbabaie A, Pedarsani R. FedPAQ: a communication-efficient federated learning method with periodic averaging and quantization. In: International Conference on Artificial Intelligence and Statistics; 2020; PMLR. p. 2021–31.

32. Shlezinger N, Chen M, Eldar YC, Poor HV, Cui S. UVeQFed: universal vector quantization for federated learning. IEEE Trans Signal Process. 2021;69:500–14. doi: 10.1109/TSP.78.

33. Rothchild D, Panda A, Ullah E, Ivkin N, Stoica I, Braverman V, et al. FetchSGD: communication-efficient federated learning with sketching. In: International Conference on Machine Learning; 2020; PMLR. p. 8253–65.

34. Lian X, Zhang W, Zhang C, Liu J. Asynchronous decentralized parallel stochastic gradient descent. In: International Conference on Machine Learning; 2018; Stockholm, Sweden: PMLR. p 3043–52.

35. Sprague MR, Jalalirad A, Scavuzzo M, Capota C, Neun M, Do L, et al. Asynchronous federated learning for geospatial applications. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases; 2018; Dublin, Ireland: Springer. p. 21–8.

36. Xie C, Koyejo S, Gupta I. Asynchronous federated optimization. arXiv preprint arXiv:190 303934. 2013 Mar.

37. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, et al. Advances and open problems in federated learning. Found Trends Mach Learn. 2021;14:1–210. doi: 10.1561/2200000083.

38. Lalitha A, Kilinc OC, Javidi T, Koushanfar F. Peer-to-peer federated learning on graphs. arXiv preprint arXiv:190111173. 2019 Jan.
39. Warnat-Herresthal S, Schultze H, Shastry KL, Manamohan S, Mukherjee S, Garg V, et al. Swarm learning for decentralized and confidential clinical machine learning. Nature. 2021 Jun;594:265–70. doi: 10.1038/s41586-021-03583-3.
40. Li C, Li G, Varshney PK. Decentralized federated learning via mutual knowledge transfer. IEEE Internet Things J. 2022 Jan;9:1136–47. doi: 10.1109/JIOT.2021.3078543.
41. Wang S, Yang S, Li H, Zhang X, Zhou C, Xu C, et al. PyramidFL: a fine-grained client selection framework for effiicient federated learning. In: 28th ACM Annual International Conference on Mobile Computing and Networking, MobiCom 2022; 2022; Sydney, NSW, Australia: Association for Computing Machinery. p. 542–55.
42. Horváth S, Sanjabi M, Xiao L, Richtárik P, Rabbat M. Fedshuffle: recipes for better use of local work in federated learning. arXiv preprint arXiv: 220413169. 2022.
43. Mishchenko K, Khaled A, Richtárik P. Random reshuffling: simple analysis with vast improvements. Adv Neural Inf Process Syst. 2020;33:17309–20.
44. Meta. Federated Learning Simulator (FLSim). Meta Res. 2022 Jun. Available from: https://github.com/facebookresearch/FLSim. [Accessed 2022].
45. Mishchenko K, Khaled A, Richtárik P. Proximal and federated random reshuffling. In: International Conference on Machine Learning; 2022; Baltimore, MD, USA: PMLR. p. 15718–49.

Appendix A. Proof of Lemma Shuffling Radius Bounds

Proof 1. First, we have to introduce the lemma appears in [43,45], which describes the variance of sampling multiple vectors from a finite set of vectors without replacement. Supposed that there are vectors $X_1, \ldots, X_n \in \mathbb{R}^d$. We define the average as $\overline{X}$ and the population variance as σ^2. $X_{\pi_1}, \ldots, X_{\pi_k}$ are sampled uniformly from $\{X_1, \ldots, X_n\}$ without replacement, and $\overline{X}_\pi$ is their average. The average and variance of sampling is

$$\mathbb{E}\left[\overline{X}_\pi\right] = \overline{X},$$
$$\mathbb{E}\left[\left\|\overline{X}_\pi - \overline{X}\right\|^2\right] = \frac{n-k}{k(n-1)}\sigma^2. \tag{11}$$

Since f_c is L_c-smooth and the definition of x_*^i, we can get the bound from Bregman divergence in Eq. (7).

$$\begin{aligned}\mathbb{E}\left[D_{f_{\pi_i}}(x_*^i, x_*)\right] \le \mathbb{E}\left[\frac{L_{\pi_i}}{2}\|x_*^i - x_*\|^2\right] &\le \frac{L_{\max}}{2}\mathbb{E}\left[\|x_*^i - x_*\|^2\right] \\ &= \frac{\eta^2 L_{\max}}{2}\mathbb{E}\left[\left\|\sum_{j\in[i]}\nabla f_{\pi_j}(x_*)\right\|^2\right] \\ &= \frac{\eta^2 L_{\max} i^2}{2}\mathbb{E}\left[\left\|\frac{1}{i}\sum_{j\in[i]}\nabla f_{\pi_j}(x_*)\right\|^2\right].\end{aligned} \tag{12}$$

Let $X_c \triangleq \nabla f_c(x_*)$, then we have $\overline{X}_\pi = \frac{1}{j}\sum_{j\in[i]} X_{\pi_j}$ and $\overline{X} = \nabla f(x_*)$, with Eq. (11) we get

$$\begin{aligned}\mathbb{E}\left[D_{f_{\pi_i}}(x_*^i, x_*)\right] &\le \frac{\eta^2 L_{\max} i^2}{2}\mathbb{E}\left[\left\|\overline{X}_\pi\right\|^2\right] \\ &= \frac{\eta^2 L_{\max} i^2}{2}\left(\|\nabla f(x_*)\|^2 + \frac{n-i}{i(n-i)}\sigma_*^2\right).\end{aligned} \tag{13}$$

With definition of shuffling radius and $i \le n$, we get

$$\sigma_{\text{rad}}^2 \le \frac{nL_{\max}}{2}\left(n\left\|\nabla f(x_*)\right\|^2 + \frac{1}{2}\sigma_*^2\right). \tag{14}$$

■

Appendix B. Proof of Theorem 5.1

Proof 2. Using μ-convexity and L-smoothness of f, we have:

$$\begin{aligned}&\frac{\mu}{2}\left\|x_{m,*}^k - x_{m,t}^k\right\|^2 \le D_{fm}\left(x_{m,*}^k, x_{m,t}^k\right) \\ &\mathbb{E}\left[\left\|\nabla f_m(x_{m,t}^k) - \nabla f_m(x_*)\right\|\right] \le 2LD_{fm}\left(x_{m,t}^k, x_*\right).\end{aligned}$$

With the inequality, we have:

$$\begin{aligned}\mathbb{E}\left[\left\|x_{m,t}^{k,j+1} - x_{m,*}^{k,j+1}\right\|^2\right] \le &(1-\eta\mu)\mathbb{E}\left[\left\|x_{m,t}^{k,j} - x_{m,*}^{k,j}\right\|^2\right] + 2\eta\mathbb{E}\left[D_{fm}\left(x_{m,t}^{k,j}, x_*\right)\right] \\ &- 2\eta(1-\eta L)\mathbb{E}\left[D_{fm}\left(x_{m,t}^{k,j}, x_*\right)\right].\end{aligned} \tag{15}$$

Using the definition of σ_{rad}^2 and $\eta \le \frac{1}{L}$, we get following bound:

$$\mathbb{E}\left[\left\|x_{m,t}^{k,j+1} - x_{m,*}^{k,j+1}\right\|^2\right] \le (1-\eta\mu)\mathbb{E}\left[\left\|x_{m,t}^{k,j} - x_{m,*}^{k,j}\right\|^2\right] + 2\eta^3\sigma_{\text{rad}}^2. \tag{16}$$

Unrolling the recursion, we have:

$$\mathbb{E}\left[\left\|x_{m,t}^{k,n} - x_{m,*}^{k,n}\right\|^2\right] \le (1-\eta\mu)^N\mathbb{E}\left[\left\|x_t^k - x_*^k\right\|\right] + 2\eta^3\sigma_{\text{rad}}^2\sum_{j=0}^{N-1}(1-\eta\mu)^j. \tag{17}$$

With the K recursion of epochs and clients selection involved:

$$\begin{aligned}\mathbb{E}\left[\left\|x_t^{k+1}-x_*^{k+1}\right\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{C}\sum_{m\in C_t}x_{m,t}^k-\frac{1}{C}\sum_{m\in C_t}x_{m,*}^k\right\|^2\right]\\ &\le \frac{1}{C}\sum_{m\in C_t}\mathbb{E}\left[\left\|x_{m,t}^k-x_{m,*}^k\right\|^2\right]\\ &\le \frac{1}{C}\sum_{m\in C_t}\left((1-\eta\mu)^N\mathbb{E}\left[\left\|x_t^k-x_*^k\right\|^2\right]+2\mu^3\sigma_{\text{rad}}^2\sum_{j=0}^{N-1}(1-\eta\mu)^j\right)\\ &= (1-\eta\mu)^N\mathbb{E}\left[\left\|x_t^k-x_*^k\right\|^2\right]+2\mu^3\sigma_{\text{rad}}^2\sum_{j=0}^{N-1}(1-\eta\mu)^j.\end{aligned} \tag{18}$$

Since $x_{t+1}=x_t^K$ and $x_*=x_*^K$, the recursion index r can be unrolled:

$$\begin{aligned}\mathbb{E}\left[\|x_{t+1}-x_*\|^2\right] &= \mathbb{E}\left[\left\|x_t^K-x_*^K\right\|^2\right]\\ &\le (1-\eta\mu)^{NK}\mathbb{E}\left[\|x_t-x_*\|^2\right]+2\mu^3\sigma_{\text{rad}}^2\sum_{j=0}^{N-1}(1-\eta\mu)^j\sum_{k=0}^{K-1}(1-\eta\mu)^{kN}.\end{aligned} \tag{19}$$

Unroll the recursion for index t and apply the property described before:

$$\begin{aligned}\mathbb{E}\left[\|x_T-x_*\|^2\right] &\le (1-\eta\mu)^{NKT}\|x_0-x_*\|^2+2\mu^3\sigma_{\text{rad}}^2\sum_{j=0}^{N-1}(1-\eta\mu)^j\sum_{k=0}^{K-1}(1-\eta\mu)^{kN}\sum_{t=0}^{T-1}(1-\eta\mu)^{tKN}\\ &\le (1-\eta\mu)^{NKT}\|x_0-x_*\|^2+2\mu^3\sigma_{\text{rad}}^2\sum_{j=0}^{NKT-1}(1-\eta\mu)^j\\ &\le (1-\eta\mu)^{NKT}\|x_0-x_*\|^2+\frac{2\eta^2}{\mu}\sigma_{\text{rad}}^2.\end{aligned} \tag{20}$$

■