Resource-Efficient Federated Learning with Hierarchical Aggregation in Edge Computing

IEEE INFOCOM 2021, Paper # 1570663288

***Abstract*—Federated learning (FL) has emerged in edge computing to address limited bandwidth and privacy concerns of traditional cloud-based centralized training. However, the existing FL mechanisms may lead to long training time and consume a tremendous amount of communication resources. In this paper, we propose an efficient FL mechanism, which divides the edge nodes into $\mathcal{K}$ clusters by balanced clustering. The edge nodes in one cluster forward their local updates to cluster header for aggregation by synchronous method, called cluster aggregation, while all cluster headers perform the asynchronous method for global aggregation. This processing procedure is called hierarchical aggregation. Our analysis shows that the convergence bound depends on the number of clusters and the training epochs. We formally define the resource-efficient federated learning with hierarchical aggregation (RFL-HA) problem. We propose an efficient algorithm to determine the optimal cluster structure (*i.e.*, the optimal value of $\mathcal{K}$) with resource constraints and extend it to deal with the dynamic network conditions. Extensive simulation results obtained from our study for different models and datasets show that the proposed algorithms can reduce completion time by 34.8%-70% and the communication resource by 33.8%-56.5% while achieving a similar accuracy, compared with the well-known FL mechanisms.**

***Index Terms*—Federated Learning, Mobile Edge Computing, Resource-constraint, Cluster, Optimization.**

I. Introduction

In the era of big data, billions of Internet of Thing (IoT) devices and smartphones around the world produce a significant amount of data per second [1], [2]. Therefore, the traditional way of uploading those data to the remote cloud for processing will encounter many issues, including privacy leaks, network congestion, and high transmission delay [3]. Since data are generated at the network edge, mobile edge computing (MEC) is a natural alternative [4], [5], which uses the computing and storage resources of edge nodes to perform data processing close to the data generators. According to Cisco's survey, most IoT-created data will be stored, processed, and analyzed close to or at the network edge [6], [7]. With more data and advanced applications (*e.g.*, autonomous driving, virtual reality, and image classification), machine learning (ML) tasks will be a dominant workload in MEC [8], [9]. To alleviate the network bandwidth burden and avoid the privacy leakage, FL becomes an efficient solution to analyze and process the distributed data on edge nodes for those ML tasks [10]–[12].

Among previous FL frameworks, the dominant one is the parameter server (PS) based framework [13]. This framework comprises two components, the PS and workers, which form a two-layer architecture. Edge nodes usually act as workers and use their local data to cooperatively train models. After each node performs several local updates, the local model will be forwarded to the PS for processing, which is called *global aggregation*. Under this framework, the PS maintains globally shared models to solve large-scale ML problems, which can protect privacy [14], reduce network delay, and relieve network burden [15], [16].

However, the PS-based two-layer framework will also encounter many difficulties for the following reasons. 1) Since PS is always located on a remote cloud platform to provide reliable services [13], the communication between workers (or edge nodes) and the remote servers may be frequently unavailable, slow, and expensive (*e.g.*, frequent backhaul and long communication distance). 2) The resources (*e.g.*, communication and computation) are always limited in the MEC system [17], [18]. 3) The size of model parameters may reach tens or even hundreds of megabytes in many ML tasks [19]. 4) The number of workers (or edge nodes) often ranges from thousands to billions [20]–[22]. Therefore, communication cost may become the principal constraint of this framework. Even though many proposed solutions are put forward to improve the performance of FL, most of them will also cause network congestion and slow down the convergence of model training since they simply adopt the PS-based two-layer framework.

To alleviate the total communication cost in FL, a natural solution adopts the lossy compression, which reduces the amount of total forwarded data. Unfortunately, since it will not decrease the number of edge nodes that forward models to PS, they only offer a small amount of network traffic reduction, but sacrifice the accuracy of the trained model and incur high computation overhead [23], [24]. Hence, it is highly desirable for more effective and fundamental solutions, which can reduce the communication cost between edge nodes and PS without losing the model accuracy.

In MEC, many edge nodes cooperate to handle a general task by low-cost communication to exchange their current processing results [25], which motivates our design. In this paper, we aim to reduce the high-cost communication between edge nodes and PS, and strengthen cooperation among edge nodes. We propose a novel cluster-based FL mechanism. Under our mechanism, we divide all the edge nodes into $\mathcal{K}$ clusters by balanced clustering [26], [27]. Among the edge nodes in one cluster, a leader node (LN) is chosen as the cluster header to aggregate all local models in this cluster. Those edge nodes in the same cluster perform synchronous method for aggregation, called *cluster aggregation*, while

all LNs communicate with the PS for global aggregation in an asynchronous method. We call the whole processing procedure as *hierarchical aggregation*. Considering the low-cost intra-cluster communication between edge nodes, the proposed cluster-based FL result in a significant resource reduction in the runtime. Under hierarchical aggregation, the theoretical analyses show that the training performance mainly depends on the cluster topology and resource budgets in Section III. As a result, it is critical to determine how to organize these edge nodes into clusters, *i.e.*, the optimal value of $\mathcal{K}$, under resource constraints. The main contributions of this paper are as follows:

1) We design a novel cluster-based FL mechanism, which performs hierarchical aggregation. We analyze the convergence bound that incorporates the number of clusters and the training epochs.
2) Based on the analysis results, we formally define the resource-efficient federated learning with hierarchical aggregation (RFL-HA) problem. We propose an efficient algorithm to determine the optimal value of $\mathcal{K}$ with resource constraints. We also extend our algorithm to deal with a dynamic scenario in which the network situations may vary with time by re-clustering the edge nodes.
3) We conduct extensive simulations using various models and datasets. The simulation results show that the proposed algorithms reduce training time by 34.8%-70% and the communication resource by 33.8%-56.5% while achieving similar accuracy, compared with the well-known FL mechanisms.

The rest of this paper is organized as follows. We introduce some FL mechanisms and propose our cluster-based mechanism in Section II. In Section III, we present our algorithm for the RFL-HA problem and extend it to dynamic scenarios. Section IV evaluates the performance of our proposed algorithms. We conclude this paper in Section V.

II. Preliminaries

This section first reviews some related work about FL. Then, we describe our proposed mechanism and prove its convergence. Finally, we put forward the problem definition.

A. Federated Learning

Many well-known FL mechanisms have emerged since FL was first proposed in 2016 [28]. The authors in [8] propose a solution that dynamically adapts the frequency of global aggregation. Another mechanism, called FedAvg [18], combines local stochastic gradient descent on each worker with a server that performs model averaging. The mechanism proposed in [29] adopts a hierarchical structure and allows multiple edge servers to perform partial model aggregation, which is similar to ours. However, those mechanisms perform global aggregation in a synchronous method, and each training epoch only progresses as fast as the slowest edge nodes [30], called *straggler effect*. Due to node heterogeneity and data imbalance [10], the probability of the occurrence of straggler effect increases with more and more edge nodes. As a result, those solutions are not suitable for large-scale ML tasks, leading to poor scalability [11]. To conquer these disadvantages, the asynchronous FL [31]–[33] is proposed to perform global aggregation as soon as the PS collects one local update from an arbitrary worker. However, this mechanism usually requires more training epochs to achieve the similar accuracy as [8], [18], especially for unbalanced data distribution. In addition, all of the above works will consume huge communication resources due to PS-based two-layer communication architecture. The detailed comparison among these works is shown in Table I.

FL mechanisms	Epoch duration	Scalability	Training efficiency	Resource consumption
[8], [18], [29]	Long	Poor	High	High
[31]–[33]	Short	Good	Low	High
Ours	Short	Good	High	Low

TABLE I: Comparison of different FL mechanisms

B. Gradient Descent-based FL

In FL, each edge node trains its own local model based on a collection of data samples. Let j denote a training sample, including feature x_j and label y_j. Given a model, *e.g.*, logistic regression (LR) [34], or convolutional neural network (CNN) [35], the loss function is denoted as $f_j(w, x_j, y_j)$, written as $f_j(w)$ for simplicity, where w is the model vector. The loss function on dateset D is

$$f(w) = \frac{1}{|D|} \sum_{j \in D} f_j(w) \tag{1}$$

where $|D|$ is the number of training samples in D. Then, the learning problem is to find the optimal model vector w that minimizes the loss function $f(w)$, expressed as

$$w^* = \arg\min f(w) \tag{2}$$

It is almost impossible to solve Eq. (2) directly, especially for deep learning models. Alternatively, each edge node will perform gradient-descent in each local update (*i.e.*, iteration) to gradually approach the optimal solution. For iteration t, the local update rule is described as follows:

$$w(t) = w(t-1) - \eta \nabla f(w(t)) \tag{3}$$

where $\eta > 0$ is the step size.

C. Cluster-based Federated Learning Mechanism

We introduce the cluster-based FL mechanism, as illustrated in Fig. 1. Assume that there are N edge nodes, and the edge nodes are divided into $\mathcal{K}$ clusters, with n_k edge nodes in each cluster k. The clustering operation will be triggered after each edge node i forwards its feature vector V_i to PS, where V_i is mainly extracted from the communication cost (*e.g.*, the model size, network bandwidth, and the communication distance between edge nodes) and the model training abilities (*e.g.*, the computing ability, the dataset size, and the remaining battery power). Specially, we consider a scenario in which there are several small areas (*e.g.*, companies, hospitals) separated by tens of kilometers or more, and each area has thousands of edge nodes. Apparently, we will completely separate the edge nodes in each area according to their communication cost. After that, those nodes in each area will be further divided into several small clusters.

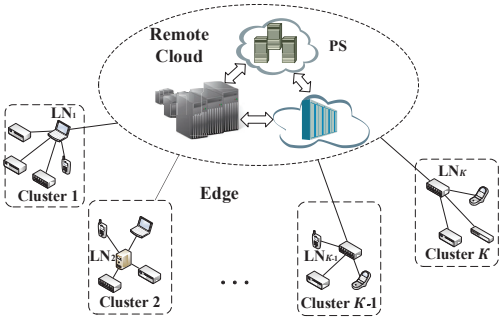

Fig. 1: The architecture of our proposed mechanism

To avoid the occurrence of empty clusters (*e.g.*, $n_k = 0$), and balance the weight of each cluster, we adopt the well-known balanced clustering method (*e.g.*, balanced K-means [26], FSCL [27]), which minimizes the mean square error and balances the cluster size. In other words, they reduce the difference in the number of edge nodes among all clusters to make $n_k \to \lfloor \frac{N}{\mathcal{K}} \rfloor + \beta_k$, where $\beta_k \in \{0, 1\}$. The whole clustering process is performed by PS. Then, PS simply chooses the leader node LN_k of cluster k with powerful computing and communication capabilities based on feature vector, and report the clustering result to all edge nodes.

After clustering, edge node i in cluster k starts using its dataset D_k^i to train the local model w_k^i. The loss function is

$$F_k^i(w_k^i) = \frac{1}{|D_k^i|} \sum_{j \in D_k^i} f_j(w_k^i) \tag{4}$$

Assume that there are totally T training epochs, with one global aggregation in each epoch. Let H denote the number of local updates (*i.e.*, iterations) that each edge node performs between its two consecutive cluster aggregation. After H iterations, edge nodes in cluster k will send the updated local model to LN_k. Once collecting local models from all edge nodes in cluster k, LN_k will perform the cluster aggregation. The new model after cluster aggregation is defined as

$$w_k = \frac{\sum_{i=1}^{n_k} |D_k^i| w_k^i}{\sum_{i=1}^{n_k} |D_k^i|} \tag{5}$$

This model will be forwarded to PS for calculating the global model by staleness-aware global update approach mentioned in the next subsection. Then, the global loss function $F(w^t)$ after $t \in \{1, 2, .., T\}$ epochs is

$$F(w^t) = \frac{\sum_{k=1}^{\mathcal{K}} \sum_{i=1}^{n_k} |D_k^i| F_k^i(w^t)}{\sum_{k=1}^{\mathcal{K}} \sum_{i=1}^{n_k} |D_k^i|} \tag{6}$$

The global loss function $F(w^t)$ cannot be directly computed without sharing global model to all edge nodes by PS.

We should note that our proposed FL mechanism is the generalization of the previous FL solutions. For example, if $\mathcal{K}$ is 1, it becomes the synchronous FL [36]. If $\mathcal{K}$ is N, it is exactly the asynchronous FL [31]. In this paper, we will study the impact of parameter $\mathcal{K}$ on the training performance.

D. Staleness-aware Global Update

Since edge nodes in each cluster perform the synchronous FL method, their staleness, denoted as τ, is the same. For cluster k, its staleness is defined as the number of experienced epochs since its last global update. PS updates the global model with staleness treatment, that is, the weight of each newly received model from an arbitrary cluster will be determined by τ,

$$w^t = (1 - \alpha_\tau^t) w^{t-1} + \alpha_\tau^t w_k \tag{7}$$

where α_τ^t is the weight of w_k at epoch t with staleness τ. Then, we adopt a function [31] to determine the value of α_τ^t, that is,

$$\alpha_\tau^t = \begin{cases} \alpha, & \tau \leq a \\ \alpha \cdot \tau^{-b}, & \tau > a \end{cases} \tag{8}$$

where $a > 0$, $b \geq 0$, and $\alpha \in (0, 1)$ is an initial model weight. This function implies that when $\tau > a$, the weight of model drops rapidly as the staleness becomes higher. In fact, when we divide edge nodes into different numbers of clusters, the model weight of each cluster will drop with the increasing number of clusters. We initialize weight α as $\alpha = \phi(\mathcal{K}) = 1 - \frac{\mathcal{K}-1}{N}$, with $\mathcal{K} \in \{1, .., N\}$, and we also set a lower bound for α based on the analysis in Section III. After substituting α in Eq. (8), we obtain the expression of α_τ^t as follows

$$\alpha_\tau^t = \begin{cases} 1 - \frac{\mathcal{K}-1}{N}, & \tau \leq a \\ (1 - \frac{\mathcal{K}-1}{N}) \tau^{-b}, & \tau > a \end{cases} \tag{9}$$

By Eq. (9), we have $\alpha_1^t = 1$ if $\mathcal{K} = 1$. As a result, $w^t = \frac{\sum_{i=1}^{N} |D_i| w_i}{\sum_{i=1}^{N} |D_i|}$, which is same as that in the synchronous FL [8].

E. Convergence Analysis

To analyze the convergence of our proposed mechanism, we assume the loss function is strongly-convex and smooth [31].

Assumption 1: Assume that the loss function f satisfies the following conditions:

1) f is μ-strongly convex, where $\mu \geq 0$, *i.e.*,
$$f(y) - f(x) \geq \nabla f^{\mathbb{T}}(x)(y - x) + \frac{\mu}{2} \|y - x\|^2, \forall x, y$$
2) f is β-smoothness, where $\beta > 0$, *i.e.*,
$$f(y) - f(x) \leq \nabla f^{\mathbb{T}}(x)(y - x) + \frac{\beta}{2} \|y - x\|^2, \forall x, y$$
3) There exists at least one solution x^* for global optimization that can minimize the loss function, *i.e.*,
$$x^* = \inf_x f(x) \text{ and } \nabla f(x^*) = 0, \exists x^* \in R^d$$

The above assumptions can be satisfied for many models with convex loss function, *e.g.*, linear regression [37], LR [34] and SVM [38]. The loss function $f(w, x_j, y_j)$ of those models are listed below.

- Liner regression: $\frac{1}{2} \|y_j - w^{\mathbb{T}} x_j\|^2, y_j \in R$
- LR: $-\log(1 + \exp(-y_j w^{\mathbb{T}} x_j)), y_j \in \{0, 1\}$
- SVM: $\frac{\lambda}{2} \|w\|^2 + \frac{1}{2} \max\{0; 1 - y_j w^{\mathbb{T}} x_j\}, y_j \in \{-1, 1\}$

According to the above assumptions, we prove the convergence of our two-layer FL mechanism in two steps. We analyze the convergence bound after H local updates. Based on that, we will derive the bound after T epochs.

Definition 1: Assume that the global loss function F is β-smooth and μ-strongly convex. $\forall w \in R^d$ and $\forall j \in D_k^i$,

where $k \in \{1, ..., \mathcal{K}\}$ and $i \in \{1, ..., n_k\}$, we define an upper bound Q_1 of $\|\nabla f(w; j) - \nabla F(w)\|^2$, *i.e.*,

$$\mathbb{E}\|\nabla f(w; j) - \nabla F(w)\|^2 \leq Q_1$$

We also define Q_2 as the upper bound of $\|\nabla f(w; j)\|^2$, *i.e.*,

$$\mathbb{E}\|\nabla f(w; j)\|^2 \leq Q_2$$

Theorem 1: When the following conditions are satisfied: 1) $\eta < \frac{1}{\beta}$; and 2) $F(w^0) - F(w^*) > \frac{Q_1+Q_2}{2\eta\mu^2}$, the convergence bound of the global loss function F after T epochs is,

$$\begin{aligned}
&\mathbb{E}[F(w^T) - F(w^*)] \\
&\leq [\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H]^T (F(w^0) - F(w^*)) \\
&+ \frac{(Q_1+Q_2)(1 - [\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H]^T)}{2\eta\mu^2}
\end{aligned} \tag{10}$$

where w^0 is the initial model parameter, and w^* denotes the optimal model which minimizes the global loss function F.

Proof: Since some proof procedure can be found in the previous work [31], we only show the differences from [31]. After edge node i in an arbitrary cluster k performs H local updates by the model $w^{t-\tau}$, the convergence bound is

$$\begin{aligned}
&\mathbb{E}[F(w^{t-\tau,H}) - F(w^*)] \\
&\leq (1-\eta\mu)^H [F(w^{t-\tau,0}) - F(w^*)] + \frac{H\eta Q_1}{2}
\end{aligned} \tag{11}$$

where $w^{t-\tau,H}$ is derived from the global model $w^{t-\tau}$ by H iterations. Apparently, $w^{t-\tau,0}$ and $w^{t-\tau}$ are equal. Then, the asynchronous FL mechanism in [31] will perform one global aggregation immediately. But for our FL mechanism, only after n_k edge nodes in cluster k have performed H iterations and the LN_k aggregates their local models based on Eq. (5), the PS will update the global model by Eq. (7). Thus, the convergence bound of our FL mechanism after t epochs is:

$$\begin{aligned}
&\mathbb{E}[F(w^t) - F(w^*)] \\
&\leq (1-\alpha_\tau^t) F(w^{t-1}) + \alpha_\tau^t \mathbb{E}[F(w_k)] - F(w^*) \\
&\leq (1-\alpha_\tau^t) F(w^{t-1}) + \alpha_\tau^t \mathbb{E}[F(\frac{\sum_{i=1}^{n_k} |D_k^i| w_k^i}{\sum_{i=1}^{n_k} |D_k^i|})] - F(w^*) \\
&\leq (1-\alpha_\tau^t) F(w^{t-1}) + \alpha_\tau^t \sum_{i=1}^{n_k} \frac{D_k^i}{\sum_{i=1}^{n_k} |D_k^i|} F(w_k^i) - F(w^*) \\
&\leq (1-\alpha_\tau^t)[F(w^{t-1}) - F(w^*)] + \alpha_\tau^t \mathbb{E}[F(w_k^i) - F(w^*)]
\end{aligned} \tag{12}$$

Since the edge nodes in cluster k adopt $w^{t-\tau}$ to perform local updates, therefore, the weight of the new model w_k is α_τ^t, where $\alpha_\tau^t \leq \alpha_1^t$ according to Eq. (9). Based on Eq. (11) and Eq. (12), we can derive the convergence bound after T epochs as shown Theorem 1. ■

F. Problem Formulation

In this section, we give the definition of resource-efficient federated learning with hierarchical aggregation (RFL-HA) problem. For a specific ML task, we will determine the optimal number of clusters (*i.e.*, $\mathcal{K}$) for training under resource constraints. To train models among distributed edge nodes by FL, it is inevitable to consume resources (*e.g.*, time, computing, and communication resources). Formally, we consider M different types of resources, and each resource $m \in \{1, 2, ..., M\}$ has a budget R_m. We assume that it consumes c_m of resource m for performing H local updates at each edge node. Meanwhile, let b_m denote the consumption of resource m for each global aggregation of an edge node. Thus, the total resource consumption of T epochs is $T \cdot n_k \cdot p_k^t \cdot (c_m + b_m)$, where p_k^t is a binary variable to indicate whether cluster k is involved in global aggregation at epoch t or not. Since each cluster runs asynchronously, p_k^t is determined in real-time during training. We formulate the RFL-HA problem as follows:

$$\begin{aligned}
&\min_{T \in \{1,2,3,...\}} F(w^T) \\
&s.t. \begin{cases}
\sum_{t=1}^{T} n_k p_k^t (c_m + b_m) \leq R_m, & \forall m \\
\sum_{k=1}^{\mathcal{K}} p_k^t = 1, & \forall t \\
n_k = \lfloor \frac{N}{\mathcal{K}} \rfloor + \beta_k, & \forall k \\
p_k^t \in \{0,1\}, \beta_k \in \{0,1\} & \forall t, k
\end{cases}
\end{aligned} \tag{13}$$

The first set of inequalities indicates that the resource consumption on each type during T epochs should not exceed its budget. The second formula represents that all clusters asynchronously perform the global update, and the PS updates the global model when it receives the parameters from an arbitrary LN. In the third formula, we consider the situation that balanced clustering achieves the optimal clustering result. Since we target to minimize the loss function $F(w^T)$, it is impossible to solve Eq. (13) by finding an exact expression among $\mathcal{K}, T$ and $F(w^T)$ [8]. But using the convergence bound of the loss function, we can solve RFL-HA approximately to obtain the optimal value of $\mathcal{K}$. Then T can be derived from $\mathcal{K}$ accordingly.

III. Cluster-based Algorithm for FL

A. Approximate Solution for RFL-HA

Based on the above analyses, we first study how to solve the RFL-HA problem in a static scenario with stable conditions of both networks (*e.g.*, stable bandwidth) and edge nodes (*e.g.*, sufficient power and stable size of datasets). Given a loss function, its minimum value $F(w^*)$ is a constant. We rewrite the objective function $F(w^T)$ in Eq. (13) as $F(w^T) - F(w^*)$. Then we replace $F(w^T) - F(w^*)$ by the approximate value which is derived by convergence analysis in Eq. (10). We also substitute the first set of inequalities in Eq. (13) by $T \cdot \lceil \frac{N}{\mathcal{K}} \rceil \cdot (c_m + b_m) \leq R_m$ since $\beta_k \in \{0, 1\}$. As a result, we reformulate the RFL-HA problem as:

$$\begin{aligned}
&\min \gamma(F(w^0) - F(w^*)) + \tfrac{(Q_1+Q_2)(1-\gamma)}{2\eta\mu^2} \\
&s.t. \begin{cases}
T\lceil \frac{N}{\mathcal{K}} \rceil (c_m + b_m) \leq R_m, & \forall m \\
\sum_{k=1}^{\mathcal{K}} p_k^t = 1, & \forall t \\
p_k^t \in \{0,1\}, & \forall t, k \\
\mathcal{K} \in \{1, 2, ..., N\}
\end{cases}
\end{aligned} \tag{14}$$

where $\gamma = [\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H]^T$, which is related to T and K. The objective function in Eq. (14) decreases as T increases based on Theorem 1. Thus, the optimal value of T is $\lfloor \min_m \frac{R_m}{\lceil \frac{N}{\mathcal{K}} \rceil (c_m + b_m)} \rfloor$. We omit the rounding operation for simplicity. Then, we substitute T into the objective function in Eq. (14), yielding

$$L(\mathcal{K}) = [\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H] \frac{R_m \mathcal{K}}{N(c_m+b_m)} [F(w^0) - F(w^*)] + \frac{(Q_1+Q_2)[1 - (\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H) \frac{R_m \mathcal{K}}{N(c_m+b_m)}]}{2\eta\mu^2} \quad (15)$$

After that, the optimal value of $\mathcal{K}$ can be determined as

$$\mathcal{K}^* = \arg \min_{\mathcal{K} \in \{1,2,...,N\}} L(\mathcal{K}) \quad (16)$$

Theorem 2: We set $R_{min} = \min_m R_m$. When $R_{min} \to \infty$, we have $F(w^T) - F(w^*) \leq \frac{Q_1+Q_2}{2\eta\mu^2}$.

Proof: Because $R_{min} \to \infty$, that is, $R_m \to \infty, \forall m$, we have $T = \lfloor \min_m \frac{R_m \mathcal{K}}{N(c_m+b_m)} \rfloor \to \infty$. We also have $\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H < 1$ based on Eq. (9). Thus, $[\frac{\mathcal{K}-1}{N} + \alpha(1-\eta\mu)^H]^T \to 0$ and $\gamma \to 0$. Then it follows $\gamma(F(w^0) - F(w^*)) \to 0$ and $(1-\gamma) \to 1$. As a result, $F(w^T) - F(w^*) \leq \frac{Q_1+Q_2}{2\eta\mu^2}$. This result denotes that the model will eventually converge regardless of the value of $\mathcal{K}$ in the condition without resource constraint. ■

However, the resource constraints are unavoidable in MEC. Therefore, we demand to study the impact of $\mathcal{K}$ on Eq. (15) under limited resources, which is important for improving the training performance of our FL mechanism. We set

$$g(\mathcal{K}) = [1 - \frac{N+1-\mathcal{K}}{N}(1-(1-\eta\mu)^H)] \frac{R_m \mathcal{K}}{N(c_m+b_m)} \quad (17)$$

Then, we focus on the monotonicity of this function instead of $L(\mathcal{K})$ in Eq. (15). For simplicity, we define

$$A = \frac{1}{N}(1-(1-\eta\mu)^H), B = \frac{R_m}{N(c_m+b_m)} \quad (18)$$

We rewrite Eq. (17) as $g(\mathcal{K}) = (1 + \mathcal{K}A - (N+1)A)^{\mathcal{K}B}$. Taking the derivative, we get

$$\frac{\partial g(\mathcal{K})}{\partial \mathcal{K}} = (1 + \mathcal{K}A - (N+1)A)^{\mathcal{K}B} [B \ln(1 + \mathcal{K}A - (N+1)A) + \frac{\mathcal{K}BA}{1 + \mathcal{K}A - (N+1)A}] \quad (19)$$

The second derivative result is

$$\frac{\partial^2 g(\mathcal{K})}{\partial^2 \mathcal{K}} = (1 + \mathcal{K}A - (N+1)A)^{\mathcal{K}B} [B \ln(1 + \mathcal{K}A - (N+1)A) + \frac{\mathcal{K}BA}{1 + \mathcal{K}A - (N+1)A}]^2 + \frac{2BA[A(\mathcal{K}-1) + (1-NA)]}{[1 + \mathcal{K}A - (N+1)A]^2} \quad (20)$$

According to Eq .(18), we have $A \in (0, \frac{1}{N})$. It follows $A(\mathcal{K}-1)+(1-NA) > 0$, and $\frac{\partial^2 g(\mathcal{K})}{\partial \mathcal{K}} > 0$. So $\frac{\partial g(\mathcal{K})}{\partial \mathcal{K}}$ is monotonically increasing with $\mathcal{K}$. We define

$$H(\mathcal{K}, A) = B \ln(1 + \mathcal{K}A - (N+1)A) + \frac{\mathcal{K}BA}{1 + \mathcal{K}A - (N+1)A} \quad (21)$$

The partial derivative of function $H(\mathcal{K}, A)$ on A is

$$\frac{\partial H(\mathcal{K}, A)}{\partial A} = \frac{B[A(\mathcal{K}-N-1)^2 + 2\mathcal{K} - N - 1]}{[1 + (\mathcal{K}-N-1)A]^2} \quad (22)$$

We note that $H(\mathcal{K}, 0) = 0$ if $A = 0$.

Algorithm 1 Cluster-based Federated Learning (CFL)

```
Initialize the cluster structure;
Initialize w^0, t = 0 and r_m = 0, ∀m;
repeat
    Global Aggregation at the Parameter Server
    Receive update from LN_k, and set t ← t + 1;
    Compute w^t according to Eq. (7);
    Estimate b_m^i, using b̂_m^i, ∀m, i;
    update r_m ← r_m + Σ_{i=1}^{n_k} c_m^i + Σ_{i=1}^{n_k} b_m^i,
    Send w^t back to LN_k;
    Cluster Aggregation at the Leader Node k
    Receive local updates from all edge nodes in cluster;
    Compute w_k by Eq. (5), and estimate b̂_m^i, ∀m, i;
    Send w_k, Σ_{i=1}^{n_k} c_m^i, and b̂_m^i, ∀m, i, to PS;
    Receive w^t from PS and return it back to edge nodes;
    Procedure at Edge Node i in Cluster k
    Receive w^t from LN_k;
    Perform H local updates;
    Send resource consumption c_m^i and w_k^i to LN_k;
until r_m ≥ R_m, ∃m;
return the final model parameter w^t;
```

Theorem 3: If $R_m < \mathbb{R}, \forall m$, where $\mathbb{R}$ a finite real number, we have $\mathcal{K}^* \in \{1, ..., \lfloor \frac{N+1}{2} \rfloor\}$.

Proof: If $\frac{N+1}{2} < \mathcal{K} \leq N$, we obtain $\frac{\partial H(\mathcal{K}, A)}{\partial A} > 0$. For $H(\mathcal{K}, 0) = 0$, we have $H(\mathcal{K}, A) > 0$ if $A > 0$. Thus, we get $\frac{\partial g(\mathcal{K})}{\partial \mathcal{K}} > 0$, making the loss function upper bound increase with $\mathcal{K}$.

On the contrary, assume that $1 \leq \mathcal{K} \leq \frac{N+1}{2}$. We consider the following two propositions with different conditions.

1) $\frac{N-1}{N^2} < A < \frac{1}{N}$. We have $\frac{\partial H(\mathcal{K}, A)}{\partial A} > 0, \forall \mathcal{K}$, and $\frac{\partial g(\mathcal{K})}{\partial \mathcal{K}} > 0$. Thus, the minimum value of function $g(\mathcal{K})$ is $g(1)$, which means $\mathcal{K}^* = 1$. This case becomes the synchronous FL.

2) $0 < A \leq \frac{N-1}{N^2}$. If $\mathcal{K} = 1$, we have $\frac{\partial H(1,A)}{\partial A} < 0$. Since $H(1, 0) = 0$, we derive $H(1, A) < 0$. Meanwhile, if $K = \frac{N+1}{2}$, we have $H(\frac{N+1}{2}, A) > 0$ for $\frac{\partial H(\frac{N+1}{2}, A)}{\partial A} > 0$, and $H(\frac{N+1}{2}, 0) = 0$. Due to the continuity of the function $H(\mathcal{K}, A)$, there exist $\mathcal{K}^* \in (1, \frac{N+1}{2})$, making $H(\mathcal{K}^*, A) = 0$. As a result, $g(\mathcal{K})$ will takes a minimum value at $\mathcal{K}^*$ where $\frac{\partial g(\mathcal{K}^*)}{\partial \mathcal{K}^*} = 0$. ■

Since the solution of $\frac{\partial g(\mathcal{K})}{\partial \mathcal{K}} = 0$ is difficult to be obtained directly and $\mathcal{K}^*$ is a positive integer with less than $\frac{N+1}{2}$, we can explore a proper value of $\mathcal{K}^*$ within a finite range that minimizes function $g(\mathcal{K})$. To calculate the value of $g(\mathcal{K})$, we should estimate some parameters (*e.g.*, c_m, b_m, μ and α) during training in practice, *e.g.*, we calculate μ based on Assumption 1. Since Eq. (17) always increases when $\mathcal{K}$ become larger if $\mathcal{K} > \frac{N+1}{2}$, We take the lower bound of α as 0.5 is benefit for the model convergence. After that, we can search $\mathcal{K}^*$ by Eq. (17) with a time complexity of $\mathcal{O}(\log \frac{N+1}{2})$.

B. Cluster-based Federated Learning

In this section, we present the cluster-based federated learning (CFL) algorithm, as described in Alg. 1. During initialization, the PS demand to perform global aggregation

for several epochs among all edge nodes for collecting V_i from edge node i and estimating the parameters (*e.g.*, c_m, b_m and μ). Then, PS can search $\mathcal{K}^*$, perform balanced clustering and select LNs. After that, we focus on the detailed operations on three components, PS, LNs, and edge nodes, respectively. The PS mainly implements two functions (Line 5-9). 1) After receiving local models from each LN, PS updates the global model using Eq. (7) and returns the updated model. 2) The PS also estimates the resource consumption and monitors whether the resource constraints are satisfied or not. Specifically, the PS will update the resource counter r_m to record total resource consumption, and return the final model when r_m exceeds the constraint. Each LN is responsible for forwarding the result of cluster aggregation to PS (Line 13) and returning the updated model to edge nodes in the cluster (Line 14). Edge nodes mainly play the role of local model trainers. After receiving the global model, edge nodes perform H iterations and forward the results to the LN in the cluster. The parameters c_m^i and b_m^i is utilized to denote the different resource consumption of each edge node i in practical. Each edge node repeats training until the algorithm stops (Line 16-18). In fact, LNs are also the members of edge nodes, but we treat LNs as an independent component in this algorithm.

C. Extension to Dynamic Scenarios

The CFL algorithm assumes that the network condition is stable and each edge node will not suffer from network interruption and training capability degradation due to unexpected accidents, *e.g.*, battery exhausted and computing resource occupation. However, it is not the case in practice [39]. Based on these concerns, we study a dynamic cluster-based FL mechanism for practical situations in this section. To this end, we propose two approaches. One is fixed re-clustering, and the other is adaptive re-clustering, which can be invoked as a supplement to the CFL algorithm.

Fixed Re-clustering. Fixed re-clustering means clustering all edge nodes after every T_b epochs, where T_b is a pre-set constant. Obviously, PS keeps a counter t_b to denote the number of experienced epochs since the last re-clustering. If $t_b \geq T_b$, re-clustering will be triggered. However, the re-clustering operation will suspend the training process. For example, edge node i will stop training, transfer the new feature vectors V_i to PS, and wait for the re-clustering result. Therefore, it is reasonable to trigger re-clustering only after all clusters have undergone several updates, that is,

$$T_b = \varrho \cdot \mathcal{K}, \varrho \in N^* \tag{23}$$

To determine the value of ϱ, we consider the specific value of $\mathcal{K}$. A small value of $\mathcal{K}$ indicates that there are more edge nodes in one cluster. Thus, the FL mechanism is more susceptible to the stragglers effect, and we adopt a small ϱ. But for $\mathcal{K} = 1$ and $\mathcal{K} = N$, we set $\varrho \to \infty$, because both approaches can not implement re-clustering. However, the way to obtain the optimal value of ϱ demands future study.

Adaptive Re-clustering. Fixed re-clustering is only triggered after every T_b epochs. Thus, it can not fully adapt to the real-time situation of severe deterioration of network status. To be more flexible, we propose another approach, called adaptive re-clustering. PS maintains an array $\mathcal{S} = \{s_1; s_2; ...; s_{\mathcal{K}}\}$ to denote the conditions of all clusters, while $s_k = 0, \forall k \in \{1, 2, ..., \mathcal{K}\}$, indicates that no straggler appears in cluster k. As soon as LN_k discovers the occurrence of straggler (*e.g.*, the time for cluster aggregation is greatly increased), it sets $s_k = 1$ and forward this parameter to PS during the next global aggregation. The PS will modify the corresponding value s_k to 1 in array $\mathcal{S}$. When a certain number, *e.g.*, $\widetilde{\mathcal{K}} \in \{1, .., \mathcal{K}\}$, of clusters report the presence of stragglers, $i.e.$,

$$\sum_{k=1}^{\mathcal{K}} s_k = \widetilde{\mathcal{K}} \tag{24}$$

re-clustering will be triggered. After adaptive re-clustering, we reset $t = 0$ and $\mathcal{S} = \{0; ...; 0\}$ for the next re-clustering. Even though LN dies in adaptive re-clustering and cannot perform cluster aggregation and forward parameters to PS, fixed re-clustering can be triggered normally to discard the dead nodes and select new LNs for future training. Therefore, the combination of two approaches can deal with the slow or sudden degradation of training conditions, *e.g.*, straggler effect and nodes failure, and maintain efficient training until the ML task is completed.

We introduce the dynamic cluster-based FL algorithm (DCFL) in Alg. 2 and omit the process of model training which is similar to that in Alg. 1. In addition to maintaining the global model and updating resource constraints, PS

Algorithm 2 Dynamic Cluster-based Federated learning (DCFL)

1: Initialize the cluster structure;
2: Initialize $T_b, \widetilde{\mathcal{K}}, t_b = 0, \mathcal{S} = \{0; ...; 0\}$;
3: **repeat**
4: **Global Aggregation at the Parameter Server**
5: Receive update from LN_k, and set $t_b \leftarrow t_b + 1$;
6: Update global model, $\mathcal{S}$, and resource budgets;
7: **if** $\sum_{k=1}^{\mathcal{K}} s_k \geq \widetilde{K}$ or $t_b \geq T_b$ **then**
8: Broadcast CLU flag to edge nodes;
9: Receive V_i from edge nodes, and perform re-clustering;
10: Send clustering result and global model to edge nodes;
11: Set $t_b \leftarrow 0, \mathcal{S} \leftarrow \{0; ...; 0\}$;
12: **else**
13: Send updated global model back to LN_k;
14: **Cluster Aggregation at the Leader Node** k
15: **if** Receive CLU flag **then**
16: Wait for clustering result;
17: **else**
18: Wait for local updates from edge nodes in cluster k;
19: Perform cluster aggregation, and update s_k;
20: Receive global model from PS and send to edge nodes;
21: **Procedure at Edge Node** i **in Cluster** k
22: **if** Receive CLU flag **then**
23: Send V_i to PS;
24: Wait for clustering result;
25: **else**
26: Receive model from LN_k or PS;
27: Perform local updates and send updated model to LN_k;
28: **until** Resource constraints are exceeded;
29: return the final global model;

should manage re-clustering in DCFL. Whenever PS collects updates from a cluster, it checks whether the condition of re-clustering is satisfied or not. Re-clustering only occurs when the conditions for fixed re-clustering ($t_b \geq T_b$) or adaptive re-clustering ($\sum_{k=1}^{\mathcal{K}} s_k \geq \widetilde{\mathcal{K}}$) are met (Line 7). Then PS will perform re-clustering and select the new LN for each cluster (Lines 8-10) depending on the new feature vector V_i of each edge node. For LNs and edge nodes, they will stop performing cluster aggregation or local updates when they receive the CLU flag. Edge nodes will forward the new feature vectors to the PS (Line 23). After receiving the re-clustering results, those edge nodes will initiate their procedure for model training.

IV. Performance Evaluation

A. Simulation Settings

Benchmarks. We compare our proposed algorithms with two benchmarks for performance evaluation. The first benchmark is FedAsync [31], which is an asynchronous FL algorithm with staleness treatment. The global update of FedAsync is performed as soon as one model from an arbitrary edge node is received by PS. We also choose an improved synchronous FL algorithm FedAvg [18] as the baseline. It randomly selects a fixed number of edge nodes in each epoch and aggregates the local models from these edge nodes.

Models and Datasets. The experiments are conducted over three different models (*e.g.*, SVM [38], LR [34] and CNN [35]) and two real datasets (*e.g.*, MNIST [40] and CIFAR-10 [41]). SVM and LR are trained over MNIST, which is composed of 60,000 handwritten digits for training and 10,000 for testing. They divide the digits into odd and even categories. CNN[1] is trained over both MNIST and CIFAR-10. CIFAR-10 includes 50,000 images for training and 10,000 for testing, and has ten different types of objects. We will perform stochastic gradient descent to process the mini-batch samples for training two datasets. We adopt the same mini-batch size for each edge node (*e.g.*, 60 for MNIST and 50 for CIFAR-10) [31].

Performance Metrics. We mainly adopt three common metrics for performance evaluation. 1) *Loss function* measures the difference between the predicted values and the actual values. 2) *Classification accuracy* is the proportion of correctly classified samples to all samples in the dataset. Both metrics can be derived from the model after global aggregation. 3) *Completion time* denotes the time spent until training terminates, which is used to evaluate the model training speed.

Resources. In the experiments, we compare the training performance of different algorithms under the resource (*e.g.*, time and communication) constraints. We consider one resource type in each experiment for simplicity. To quantify the communication cost, we adopt the cost of each model exchange between LN and PS as one unit [18], [31]. We also set the communication cost among edge node and LN as 0.1 unit for the following reasons. 1) The communication latency between LN and PS is more than ten times of intra-cluster communication [42], [43]. 2) Long communication distance and frequent backhaul will often lead to network congestion [42], [44], [45].

Data Distribution. To evaluate the impact of data imbalance on the simulation performance, we conduct our evaluations in a simulated environment with 100 edge nodes and distribute the dataset (with its size D) into separate nodes in three cases. In case 1, data samples are assigned to each edge node uniformly (*i.e.*, $\frac{D}{100}$ for each node). In case 2 and case 3, the dataset is distributed to each edge node according to the Gaussian distribution with the same expectation (*e.g.*, $\frac{D}{100}$) but different standard deviation σ (*e.g.*, 100 for case 2 and 300 for case 3), which denote the data size of each edge node mainly ranging from $\frac{D}{100} - 2\sigma$ to $\frac{D}{100} + 2\sigma$.

Simulation Parameters. In all simulations, we by default set the learning rate η as 0.01, and the number of local updates in each epoch as $H = 10$ [8]. We adopt $a = 5$ and $b = 1$ to deal with staleness. The only feature we consider for performing clustering is the dataset size of edge nodes. We randomly select an LN in each cluster. To evaluate the performance of our algorithms when Assumption 1 is not satisfied (*e.g.*, non-convex or non-smooth), we conduct experiments on several values of $\mathcal{K}$ (*e.g.*, 5, 10, 20, 25 and 50) to observe the impact on the performance. Then, CFL(10) denotes the algorithm with $\mathcal{K} = 10$. To analyze the impact of data imbalance, we test the completion time of different algorithms while achieving the same training accuracy (*e.g.*, 80% for SVM and 50% for CNN over MNIST) in cases 1-3, and other experiments are only conducted in case 1. For DCFL, we assume a certain fraction (*e.g.*, 0.3) of edge nodes will die at any time during training [46]. We set $\varrho = 10$ and $\widetilde{\mathcal{K}} = 1$ for re-clustering. For the benchmark, we set two subset sizes z (*e.g.*, 10 and 100) for FedAvg. We adopt the average results of 5 independent experiments to avoid accidents.

B. Simulation Results

Convergence Performance. Our first set of simulations compares CFL to baselines with different numbers of epochs, ranging from 0 to 1,000. The results are shown in Figs. 2-4. We observe that CFL(10) always achieves the lowest loss and the highest accuracy compared with both FedAvg ($z = 10$) and FedAsync, and its superiority is better reflected on CNN than on SVM. Meanwhile, the performance of FedAvg ($z = 10$) is similar to that of CFL(20), which is better than the performance of CFL(50). For example, after training 1,000 epochs on CIFAR-10 using CNN, the accuracy of CFL(10) and FedAvg is 56% and 39%, respectively. Besides, CFL requires about 300 epochs to achieve the same accuracy of FedAsync after 1,000 epochs with CNN over MNIST, which reduces the number of training epochs by 70%.

[1]The detailed CNN network architectures for MNIST (CIFAR-10): $5 \times 5 \times 32(64)$ Convolutional $\rightarrow$ Local Response Normalization $\rightarrow$ 2×2 MaxPool $\rightarrow$ $5 \times 5 \times 64(128)$ Convolutional $\rightarrow$ Local Response Normalization $\rightarrow$ 2×2 MaxPool $\rightarrow$ $1600(3200) \times 512$ Fully connected $\rightarrow$ $512 \times 10(256)$ Fully connected ($\rightarrow$ 256×10 Fully connected) $\rightarrow$ Softmax.

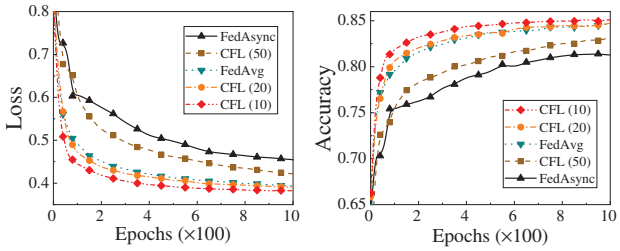

Fig. 2: Loss and Accuracy vs. No. of Epochs with SVM over MNIST.

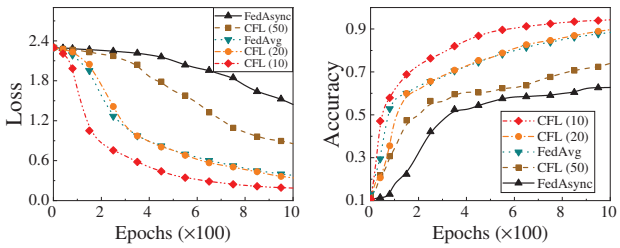

Fig. 3: Loss and Accuracy vs. No. of Epochs with CNN over MNIST.

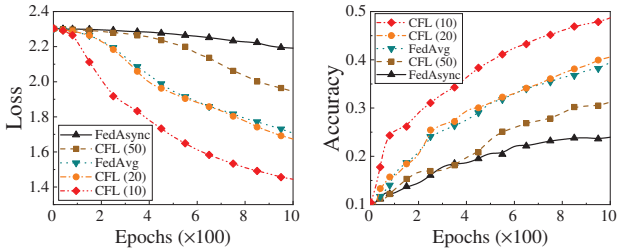

Fig. 4: Loss and Accuracy vs. No. of Epochs with CNN over CIFAR-10.

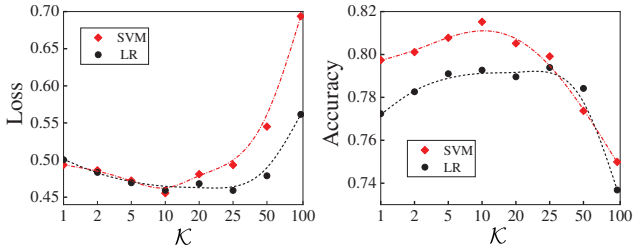

Fig. 5: Loss and Accuracy vs. $\mathcal{K}$ with Time Budget of 600s.

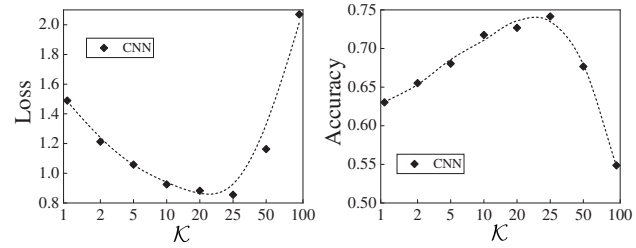

Fig. 6: Loss and Accuracy vs. $\mathcal{K}$ with Time Budget of 1,800s.

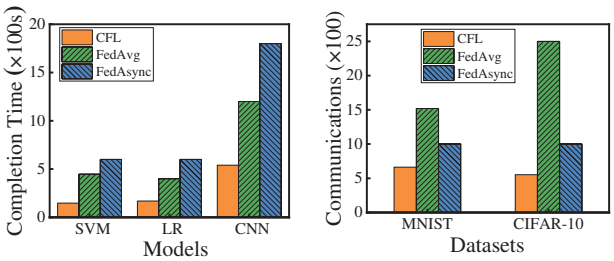

Fig. 7: The Comparison of Resource Consumption. *Left plot:* Time; *Right plot:* Communication.

Resource Constraints. In the second set of experiments, we compare the performance of different algorithms in the scenario with resource constraints. Similar to the settings in [8], we adopt the time budget (*e.g.*, 600s for SVM/LR and 1,800s for CNN over MNIST) for different values of $\mathcal{K}$. The communication budgets vary from 100 to 1,000 for CNN over MNIST and CIFAR-10. The results are shown in Figs. 5-9. Figs. 5-6 show that CFL achieves higher accuracy than both FedAvg ($z = 100$) and FedAsync under the time budget. For different models and datasets, there are different optimal values of $\mathcal{K}$ between 10 and 25. For instance, CFL achieves an accuracy of 74.2% with CNN over MNIST when it takes the optimal value of $\mathcal{K}$, which is about 15% and 26% higher than FedAvg and FedAsync, respectively. In the left plot of Fig. 7, we compare the completion time of three algorithms when achieving the final accuracy of FedAsync in Figs. 5-6. The result shows that CFL reduces the time by about 34.8%-70%. By the right plot of Fig. 7, CFL(50) reduces the communication cost by about 33.8% and 56.5% when achieving the accuracy of FedAsync after 1,000 communication budgets, compared with FedAvg and FedAsync, respectively. Figs. 8-9 shows the specific training process. The reason for the significant performance improvement of CFL under the communication budgets is that CFL performs cluster aggregation and reduces the communication cost between edge nodes and PS.

Data Imbalance. The completion time of different algorithms (*e.g.*, CFL, FedAvg, and FedAsync) using SVM or CNN to train MNIST is shown in Fig. 10. As the degree of data imbalance increases, the training time of FedAvg ($z = 100$) and FedAsync also rapidly increases. However, the training time for CFL remains stable in cases 1-3, especially for CNN over MNIST. We speculate that this may be attributed to the positive effect of clustering, which avoids the adverse effects of single edge nodes (FedAsync) and prevents the situation for waiting for all edge nodes in each epoch (FedAvg). In comparison, CFL demands less training time than both FedAvg and FedAsync. For example, by the left plot of Fig. 10, the training time of CFL is about 1,036s in case 1, which is 56.8% and 67.2% less than that of FedAvg and FedAsync, respectively.

Dynamic Scenarios. Assuming that a certain fraction (0.3) of edge nodes will encounter failure during training [46], we conduct this simulation to compare the performance of CFL, DCFL, and baselines. We adopt $\mathcal{K} = 10$ for CFL and DCFL. The results for CNN over MNIST and CIFAR-10 are shown in Figs. 11-12. Two aspects attract our attention. 1) The accuracy of FedAvg and CFL stops increasing at around 100 and 700 epochs, respectively. That is because the training process of FedAvg will terminate when a node

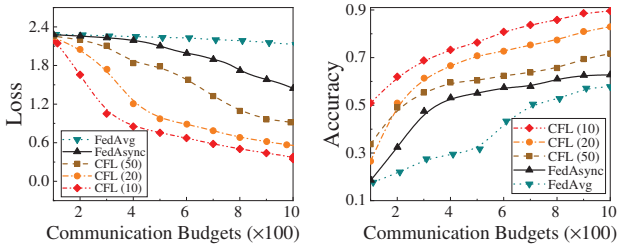

Fig. 8: Loss and Accuracy vs. Communication Budgets with CNN over MNIST.

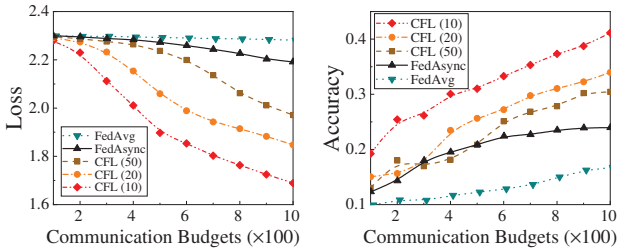

Fig. 9: Loss and Accuracy vs. Communication Budgets with CNN over CIFAR-10.

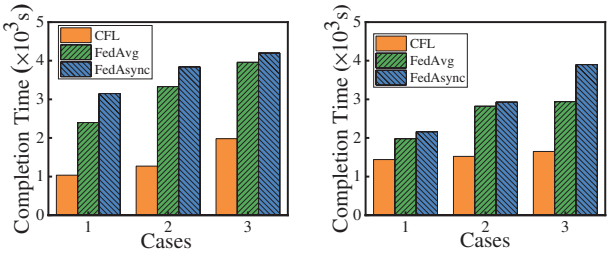

Fig. 10: Completion Time vs. Data Distribution Cases. *Left plot:* SVM; *Right plot:* CNN.

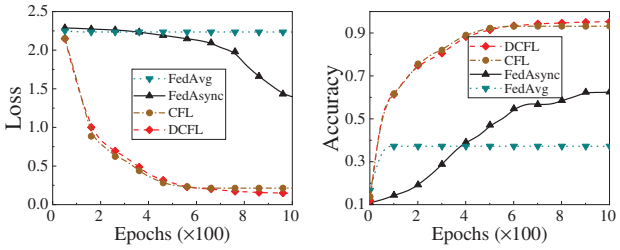

Fig. 11: Loss and Accuracy vs. No. of Epochs with CNN over MNIST in Dynamic Scenarios.

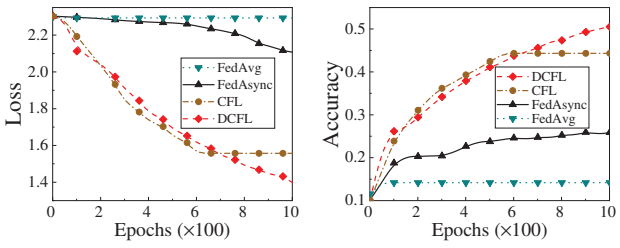

Fig. 12: Loss and Accuracy vs. No. of Epochs with CNN over CIFAR-10 in Dynamic Scenarios.

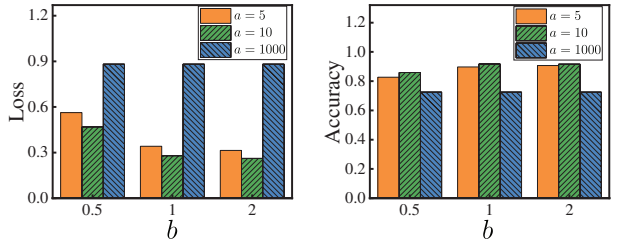

Fig. 13: Impact of Parameters a and b with CNN over MNIST. *Left plot:* Loss; *Right plot:* Accuracy.

dies. As a result, FedAvg achieves the lowest accuracy (14%) compared with other algorithms. For CFL, since edge nodes in each cluster run synchronously, the training will stop at the time when one node dies in each cluster. 2) Before CFL terminates, we notice that it converges faster than DCFL. But DCFL finally achieves higher accuracy than CFL. The reason is that CFL has fewer clusters for global updates since some clusters stop training due to the dead nodes, which results in a lower average staleness than DCFL. After that, DCFL will keep training and can achieve better performance than CFL. For example, the final accuracy of DCFL is 51% with CNN over CIFAR-10, which is 16% higher than that of CFL.

Sensitivity of a and b. We finally analyze the influence of parameters a and b on convergence performance using CFL with $\mathcal{K} = 20$. We adopt three values of a (e.g., 5, 10 and 20) and b (e.g., 0.5, 1 and 2) to train CNN over MNIST for 1000 epochs. Since there are 1000 epochs totally, $a = 1000$ indicates that we have not taken measures to deal with staleness. The results are shown in Fig. 13. We observe that different values of a and b have a minor impact on the accuracy of CFL except when $a = 1000$. For example, when $a = 10, b = 1$, CFL achieves the accuracy of 91.8%, whereas if $a = 1000$, the accuracy would be 26% lower.

To summarize, our proposed algorithms can substantially outperform two benchmarks in the following aspects. Firstly, from Figs. 2-4, CFL always achieve better convergence than benchmarks on different models during training. Secondly, Figs. 5-9 shows that CFL reduce the resource consumption by 33.8%-70% while achieving a similar accuracy, compared with baselines. We also observe that CFL effectively deals with data imbalance based on Fig. 10. Meanwhile, from Figs. 11-12, we realize that DCFL handles the node failure and maintains the training process well. Finally, Fig. 13 shows that CFL with staleness treatment improves the classification accuracy by 26% compared with the algorithm without dealing with staleness.

V. Conclusion

In this paper, we have designed a cluster-based federated learning mechanism with hierarchical aggregation. We have proposed an efficient algorithm to determine the optimal number of clusters with resource constraints and perform training in edge computing. We have further extended our algorithm to deal with the network dynamics in practice. The experimental results have indicated that the proposed mechanism can obtain excellent performance under resource constraints compared with baselines. We believe that our proposed mechanism will provide a valuable solution for federated learning.

REFERENCES

[1] K. L. Lueth, "State of the iot 2018: Number of iot devices now at 7b-market accelerating," *IOT Analytics*, 2018.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] B. M. Gaff, H. E. Sussman, and J. Geetter, "Privacy and big data," *Artificial Intelligence*, vol. 47, no. 6, pp. 7–9, 2014.

[4] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[5] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, pp. 1–1, 2020.

[6] C. V. Networking, "Cisco global cloud index: Forecast and methodology, 2015-2020. white paper," *Cisco Public, San Jose*, 2016.

[7] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO White Paper*, vol. 1, pp. 1–11, 2011.

[8] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 63–71.

[9] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[10] D. Verma, S. J. Julier, and G. Cirincione, "Federated ai for building ai solutions across multiple agencies." *arXiv: Computers and Society*, 2018.

[11] A. Hard, C. Kiddon, D. Ramage, F. Beaufays, H. Eichner, K. Rao, R. Mathews, and S. Augenstein, "Federated learning for mobile keyboard prediction." *arXiv: Computation and Language*, 2018.

[12] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records." *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018.

[13] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, B. Mcmahan *et al.*, "Towards federated learning at scale: System design," *arXiv: Learning*, 2019.

[14] M. Ammaduddin, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system." *arXiv: Information Retrieval*, 2019.

[15] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *arXiv: Artificial Intelligence*, 2019.

[16] S. Silva, B. A. Gutman, E. Romero, P. M. Thompson, A. Altmann, and M. Lorenzi, "Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data," *arXiv: Machine Learning*, 2018.

[17] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," pp. 19–27, 2014.

[18] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," pp. 1273–1282, 2017.

[19] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[20] 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," *arXiv: Learning*, 2019.

[21] X. Zhang, F. Li, Z. Zhang, Q. Li, C. Wang, and J. Wu, "Enabling execution assurance of federated learning at untrusted participants," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1877–1886.

[22] Y. Zhan and J. Zhang, "An incentive mechanism design for efficient edge learning by deep reinforcement learning approach," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2489–2498.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," pp. 265–283, 2016.

[24] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv: Computer Vision and Pattern Recognition*, 2017.

[25] G. Castellano, F. Esposito, and F. Risso, "A distributed orchestration algorithm for edge computing resources with guarantees," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2548–2556.

[26] X. Chang, F. Nie, Z. Ma, and Y. Yang, "Balanced k-means and min-cut clustering," *arXiv: Learning*, 2014.

[27] A. Banerjee and J. Ghosh, "Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 702–719, 2004.

[28] J. Konecný, H. B. Mcmahan, D. Ramage, and P. Richtarik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv: Learning*, 2016.

[29] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning." *arXiv: Networking and Internet Architecture*, 2019.

[30] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey." *arXiv: Networking and Internet Architecture*, 2019.

[31] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization." *arXiv: Distributed, Parallel, and Cluster Computing*, 2019.

[32] M. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," pp. 21–28, 2018.

[33] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices." *arXiv: Distributed, Parallel, and Cluster Computing*, 2019.

[34] S. L. Gortmaker, D. W. Hosmer, and S. Lemeshow, "Applied logistic regression." *Contemporary Sociology*, vol. 23, no. 1, p. 159, 1994.

[35] S. Shalev-Shwart and S. Ben-David, "Understanding machine learning: From theory to algorithms." *Cambridge, U.K.: Cambridge Unix. Press*, 2015.

[36] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv: Machine Learning*, 2019.

[37] S. Wassertheil and J. Cohen, "Statistical power analysis for the behavioral sciences," *Biometrics*, vol. 26, no. 3, p. 588, 1970.

[38] T. Joachims, "Making large-scale svm learning practical," *Technical reports*, 1998.

[39] Y. Tu, Y. Ruan, S. Wang, S. Wagle, C. G. Brinton, and C. Joe-Wang, "Network-aware optimization of distributed learning for fog computing," *arXiv preprint arXiv:2004.08488*, 2020.

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[41] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[42] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[43] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," pp. 49–62, 2010.

[44] J. Zhang, W. Xie, F. Yang, and Q. Bi, "Mobile edge computing and field trial results for 5g low latency scenario," *China Communications*, vol. 13, no. 2, pp. 174–182, 2016.

[45] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[46] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.