IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10228925

A Reinforcement Learning Approach for Minimizing Job Completion Time in Clustered Federated Learning

Ruiting Zhou*†, Jieling Yu*, Ruobei Wang*, Bo Li‡, Jiacheng Jiang*, and Libing Wu*
*School of Cyber Science and Engineering, Wuhan University, China
†School of Computer Science Engineering, Southeast University, China
‡Department of Computer Science and Engineering, Hong Kong University of Science and Technology, HongKong
Email: {ruitingzhou, yjling, wrb.math.cs}@whu.edu.cn, bli@cs.ust.hk, {jachin.jiang, wu}@whu.edu.cn

***Abstract*—Federated Learning (FL) enables potentially a large number of clients to collaboratively train a global model with the coordination of a central cloud server without exposing client raw data. However, the FL model convergence performance, often measured by the job completion time, is hindered by two critical factors: non independent and identically distributed (non-IID) data across clients and the straggler effect. In this work, we propose a clustered FL framework, *MCFL*, to minimize the job completion time by mitigating the influence of non-IID data and the straggler effect while guaranteeing the FL model convergence performance. *MCFL* builds upon a two-stage operation: i) a clustering algorithm constructs clusters, each containing clients with similar computing and communications capabilities to combat the straggler effect within a cluster; ii) a deep reinforcement learning (DRL) algorithm based on soft actor-critic with discrete actions intelligently selects a subset of clients from each cluster to mitigate the impact of non-IID data, and derives the number of intra-cluster aggregation iterations for each cluster to reduce the straggler effect among clusters. Extensive testbed experiments are conducted under various configurations to verify the efficacy of *MCFL*. The results show that *MCFL* can reduce the job completion time by up to 70% compared with three state-of-the-art FL frameworks.**

I. Introduction

Federated learning (FL) has become a new distributed machine learning paradigm [1]. Under the coordination of a central cloud server, a large number of clients, typically mobile and Internet of Things (IoT) devices, use local raw data to collaboratively learn a global model in a privacy-preserving manner [2]. FL has been widely used in natural language processing, computer vision and speech recognition [3]. In each round of global training, a subset of clients train a global model locally and upload the local model to a cloud server for aggregation. Then, after the cloud server gets the updated global model, it distributes it to the clients for the next round of training until the model converges.

One of the fundamental problems in FL training is how to minimize the job completion time, or minimize the model convergence time. The job completion time of FL is mainly hindered by two critical factors: non independent and identically distributed (non-IID) data across all clients and the straggler effect. **First**, clients have different device usage patterns and data samples, leading to different data distributions located on devices. It has been shown that training the FL model on non-IID data will greatly reduce the accuracy of the FL model and increase the model convergence time [4], [5], [6]. **Second**, in practice, the resources possessed by clients are likely to be different, resulting in heterogeneity of computing power and communication capabilities. The device heterogeneity dictates that the FL job completion time is constrained by the slowest client under the synchronous framework, which is known as the straggler effect [7]. Under wireless and mobile environment with limited connectivity, the straggler effect becomes the main bottleneck in realizing FL.

To deal with non-IID data, some studies proposed to represent the non-IID degree of local datasets through the gradient or accuracy of the local model, such methods, however, cannot accurately model the non-IID degree [8]. Recent studies have sought to use the deep reinforcement learning (DRL) method to model the non-IID degree of clients and select suitable clients to participate in each training round [9], [10]. However, it is not sufficient for FL model acceleration by only considering the effect of non-IID alone [11]. To deal with the straggler effect, some studies proposed to adjust the workload with the assumption that the straggler is primarily caused by the mismatch of computing capability and a large amount of data. Thus, the straggler effect can be mitigated by dynamically reducing the workload, or shifting the workload to clients with larger computing power [12]. In addition, the training workload can be further reduced by model compression or pruning [13], [14]. Some researches believed that personalized FL can reduce the straggler effect, where each client trains a personalized local model according to its computation and communication capabilities [15], [16]. However, the realization of such methods could be complex as they either require the manipulation of client datasets or local models. Another approach is the asynchronous framework, which breaks the constraints of synchronization and does not need to wait for the straggler [13], [17]. But the asynchronous gradient aggregation

The corresponding author is Bo Li.
The research was supported in part by the NSFC Grants (62072344 and U20A20177), a RGC RIF grant under the contract R6021-20, and RGC GRF grants under the contracts 16209120 and 16200221.

often uses stale parameters, which may require more iterations to converge [7]. We will discuss this more in detail in Sec. II.

Different from existing literature, we propose to use clustering to address the problem of the straggler effect. With a clustering structure, there is no need to synchronize a large number of clients, but only consider the synchronization from a cluster perspective, which can greatly reduce the opreation complexity. We next introduce the training process in a clustered FL. As shown in Fig. 1, in each round of global training, selected clients in each cluster perform local training for a number of iterations and send the trained local models to the cluster head. The cluster head performs the intra-cluster aggregation to aggregate these local models to a cluster model and sends back to clients. After a number of intra-cluster aggregations, the cluster head sends the updated cluster model to a cloud server. The cloud server performs inter-cluster aggregation to obtain the global model and distributes the new model to the clusters.

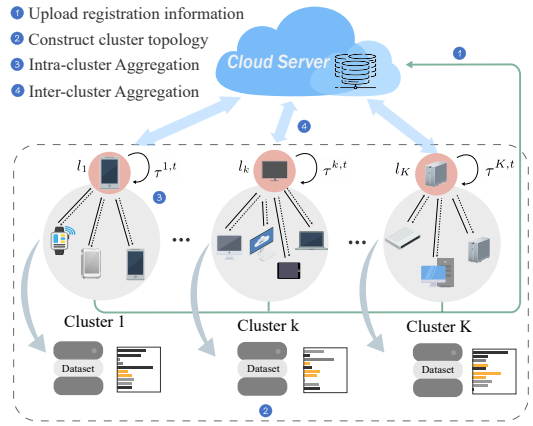


Fig. 1: The structure of clustered FL.

In our two-stage clustered FL framework, we first construct clusters, each containing clients with similar time of one round intra-cluster aggregation (*i.e.,* the computation time for a client to perform local training plus the communication time to send the local model to the cluster head) to eliminate the straggler among clients within a cluster. Although each cluster contains clients with similar time of one round intra-cluster aggregation, the straggler effect among clusters still exists. Different from the existing approaches, we use a DRL algorithm to take two actions: i) learn the optimal number of iterations for intra-cluster aggregation for each cluster, such that each cluster has similar time of one round inter-cluster aggregation (*i.e.,* the computation time for obtaining the updated cluster model plus the communication time to send the cluster model to the cloud server). As a result, the straggler effect among clusters can be mitigated; ii) select clients who contribute the most to the model accuracy by learning the non-IID attributes of clients. The impact of non-IID data can be reduced by the intelligent selection of clients. Our contribution is summarized as follows:

- *Two Stage Clustered FL Model.* To minimize the job completion time of FL, we divide the training process into two stages: i) Pre-clustering stage. We capture the device heterogeneity of clients, and model a maximum intra-cluster variance minimization problem to eliminate the stragglers within a cluster. ii) Training stage. By considering both non-IID data and the straggler effect among clusters, we model the job completion time minimization problem under the clustered FL framework. Our model can efficiently reduce the job completion time by optimizing the client selection and adjusting the number of iterations for intra-cluster aggregation.
- *MCFL Framework.* We propose a job completion time ***M***inimization ***C***lustered ***F***ederated ***L***earning framework, *MCFL*. To solve the NP-hard pre-clustering problem (PCP), *MCFL* introduces a weight index to relax the objective function of PCP and solve it by a MinMax K-means algorithm. After the clustering topology is fixed, *MCFL* leverages a DRL based on soft actor-critic with discrete actions (SACD) to take two discrete actions: select clients and determine the number of iterations for intra-cluster aggregation. The DRL algorithm can capture the functional relationship between the job completion time and the two actions under the privacy cost budget.
- *Excellent Empirical Performance.* The performance of *MCFL* is evaluated by extensive experiments based on real-world data. We observe some promising results: i) the job completion time spent by *MCFL* is only $1/5$ of that of the classical FL framework *FEDAVG* [1]; ii) *MCFL* saves up to 35% job completion time compared to *HACCS* [18] and 70% compared to *H-BASE* [19], under various settings; iii) *MCFL* converges 39% faster than *HACCS* and *H-BASE* to achieve the same target accuracy.

The remainder of this paper is organized as follows. The related work is reviewed in Sec. II. Sec. III presents the system model and formulates the optimization problem. In Sec. IV, the clustered FL framework, *MCFL* is proposed. We conduct extensive experiments in Sec. V. Finally, the conclusion is provided in Sec. VI.

II. Related Work

A. Federated Learning

Non-IID in FL. Yu *et al.* [4] and Zhao *et al.* [20] demonstrate that the performance of FL is significantly reduced by the imbalanced and non-IID data. [21], [22] suggest excluding non-IID devices. Deng *et al.* [8] propose a quality-aware client selection framework, which can evaluate the learning quality of clients and select them with quality-awareness for a given FL task within a limited budget. Li *et al.* [9] privately select high-quality devices and data samples from the perspective of data quality. Zhang *et al.* [10] utilize weight divergence to select the clients with lower degrees of non-IID data. Wang *et al.* [23] intelligently choose client devices to participate in each round of FL to counterbalance the bias introduced by non-IID data and speed up the convergence. None of the above studies jointly consider the impact of non-IID and the straggler effect to minimize the job completion time. We leverage the

DRL technique to capture the relationship between the job completion time and the coupled two impacts.

Stragglers in FL. Believe that the stragglers are the result of an imbalance between the workload and the processing capability, [24], [25] accelerate the training process of stragglers by adjusting the workload (*i.e.*, the amount of processing data) of each device. Additionally, Ji *et al.* [12] minimize the training time of FL by offloading clients' workloads to idle edge servers. However, the moving of training data may bring privacy leakage. [14], [15] speed up the training of stragglers by adjusting their models, including compressing models and placing heterogeneous models on different devices. Besides, the asynchronous manner [7], [13], [17], [26] is proposed to break the limitation of the synchronous scheme without waiting for the stragglers. However, the asynchronous approach comes with staleness, which may require more iterations to converge. Different from the above studies, we focus on reducing the straggler effect by determining the number of iterations for intra-cluster aggregation for each cluster. Our method retains the benefits of synchronization without complex adjustment, *e.g.*, model or training data adjustment.

B. Clustered Federated Learning

Briggs *et al.* [27] propose a hierarchical clustering structure to promote the performance of FL. Liu *et al.* [19] propose a client-edge-cloud hierarchical FL system to achieve communication-computation trade-offs. Wang *et al.* [28] study the clustered FL problem with time-varying data, aiming to minimize the job completion time while guaranteeing the convergence of models. These papers adopt the clustering structure to handle non-IID data, hoping to achieve approximate IID data within a cluster. Wang *et al.* [29] propose an efficient hierarchical FL to reduce the training time. Zhang *et al.* [30] implement three-layer collaborative FL to make the best client selection. These geographically based clustering frameworks aim to solve the problem of excessive communication cost. However, those papers ignore the straggler effect, and train models in a synchronous manner. Few studies [7], [31] apply hierarchical architecture to tackle this problem, but the non-IID data of clients is ignored. Considering both non-IID data and the straggler effect, we adopt clustered FL to minimize the job completion time. We cluster the clients from the perspective of time similarity to significantly reduce the impact of straggler within a cluster and among clusters. We further select clients with high quality data to speed up the convergence speed under the privacy cost budget.

III. System Model

A. System Overview

Clustered FL Scenario. As shown in Fig. 1, we consider a FL platform with I heterogeneous clients and a cloud server. When clients register to the FL platform, each of them uploads its hardware information[1]. Our goal is to minimize the job completion time while achieving the accuracy requirement ε, where ε is the desired difference between the model loss and the minimum loss. We adopt the clustered FL architecture. At t-th global round, the selected client i in cluster k performs local training for Δ_i iterations and sends the trained local model to the cluster head l_k. We assume that Δ_i is predetermined and fixed during the entire training process. Next, the cluster head performs intra-cluster aggregation to aggregate these local models to obtain the cluster model. After $\tau^{k,t}$ iterations of intra-cluster aggregation, the cluster head sends the cluster model to the cloud server. The cloud server performs inter-cluster aggregation to obtain the global model and shares the new model with the clusters afterwards. The training process terminates when the model accuracy meets the desired accuracy requirement ε. The number of global rounds during this process is defined as T. A cluster can only accommodate n_k clients per global round due to the scarcity of communication resources. Based on the training process of clustered FL, we build a two-stage operation: i) **Pre-clustering Stage**. To effectively eliminate the straggler among clients in the cluster, the cloud server first constructs K clusters, each containing clients with similar time of one round intra-cluster aggregation[2], *i.e.,* the computation time for a client to perform local training plus the communication time to send the local model to the cluster head. ii) **Training Stage**. To reduce the straggler effect among clusters, the number of iterations for intra-cluster aggregation in cluster k, $\tau^{k,t}$, will be adaptively determined. At each global round t, to overcome the impact of non-IID data, the cloud server selects a subset of clients for each cluster, under the consideration of the privacy cost budget C^{max}. Let $\mathcal{X}$ denote the integer set $\{1, 2, ..., X\}$.

Privacy in CFL. Since the exposed gradients during training also contain data owners' sensitive information [32], we apply differential privacy (DP) mechanism to further protect client's private information. During the transmission of each update, adaptive noises will be applied to disturb stochastic gradients.

B. Pre-clustering Stage

Decision Variables. This stage aims to reduce the differences in completion time for one round intra-cluster aggregation among clients. The cloud server determines the cluster topology by making the following decisions: $x_i^k \in [0, 1]$, which denotes whether client i belongs to cluster k or not.

Next, we model the time of client i for one round intra-aggregation, consisting of the following two components.

Computation Time for Local Training. We assume that all clients use the same data size for one round of local training, which is D. For client i, let f_i represent the computing capability (*i.e.*, the CPU frequency) and U_i denote CPU cycles to train one data sample. Therefore, the computational time for client i is $t_i^{cmp} = \frac{D\Delta_i U_i}{f_i}$.

Communication Time for Sending Local Updates. After local training, client i uploads its local updates to the cluster

[1]The hardware information can be obtained from the Internet. It is used to compute the time of one round intra-cluster aggregation.

[2]The time of one round intra-cluster aggregation can also be obtained by pre-training.

head l_k. The achievable transmission rate of client i is denoted as $r_i = B_i \log_2(1 + \frac{g_i p_i}{N_0})$, where N_0 denotes background noise power, p_i indicates the transmit power, B_i is the transmit bandwidth and g_i is the channel gain [19]. We assume that the data size that each client requires to upload is a constant, M, then the communication time of client i is $t_i^{com} = \frac{M}{r_i}$.

Problem Formulation. The goal of clustering is to minimize the maximum intra-cluster variance of the time for one round intra-cluster aggregation [33]. We define the problem as the Pre-clustered Problem (PCP). The time of one round intra-cluster aggregation for client i is $t_i = t_i^{cmp} + t_i^{com}$. Let $\mathbf{m}_k = \sum_{i=1}^{I} x_i^k t_i / \sum_{i=1}^{I} x_i^k$ and $\mathcal{V}_k = \sum_{i=1}^{I} x_i^k \|t_i - \mathbf{m}_k\|^2$ denote the mean and the variance of the time for one round intra-cluster aggregation in cluster k. Therefore, PCP is formulated as:

$$\text{minimize} \quad \max_{i \le k \le K} \mathcal{V}_k \tag{1}$$

$$\sum_{k \in \mathcal{K}} x_i^k = 1, \quad \forall k \in \mathcal{K}, \tag{1a}$$

$$x_i^k \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K}. \tag{1b}$$

Constraint (1a) specifies that every client can only be assigned to one cluster.

Challenges. The problem of minimizing the sum of intra-cluster variances (*i.e.*, the K-means problem) is NP-hard [34], which is equivalent to the 3-SAT problem. Thus K-means reduces to problem (1) and problem (1) is NP-hard.

C. Training Stage

Decision Variables. The cloud server needs to make the following decisions to minimize the job completion time: i) $y_i^t \in [0, 1]$: whether or not select clients i at global round t; ii) $\tau^{k,t} \in \{1, 2, \ldots, \tau_{max}\}$, the number of iterations for intra-cluster aggregation of cluster k at global round t.

Job Completion Time. Besides the time of intra-cluster aggregation, the communication time of the cluster head l_k for inter-cluster aggregation is $t_k^{com} = \frac{M}{r_k}$, where r_k is the transmit rate between the cluster head l_k and the cloud server. The calculation of r_k is the same as that of r_i. Due to the sufficient transmit power and downlink bandwidth of the cloud server, we ignore the time of downlink communication. Let S_k be the clients set of cluster k. Thus, the job completion time is expressed as:

$$T_c = \sum_{t=1}^{T} T_c^t = \sum_{t=1}^{T} \max_{k \in \mathcal{K}} (\max_{i \in S_k} (\tau^{k,t} \cdot t_i \cdot y_i^t) + t_k^{com}). \tag{2}$$

Privacy Budget. We first describe the model training process with DP. During the transmission of each update, the Gaussian noises $\boldsymbol{b}_n \sim \mathbb{N}(0, \sigma_i^2 \boldsymbol{I}_d)$ (with $\boldsymbol{I}_d$ denoting the d-dimensional identity matrix) are injected into stochastic gradient $g(\mathcal{B}_n; \theta^t)$ of each client. The perturbed stochastic gradient can be denoted as $\tilde{g}(\mathcal{B}_n; \theta^t) = g(\mathcal{B}_n; \theta^t) + \boldsymbol{b}_n$. Next, considering the limited privacy budget ρ_i, each client adopts an appropriate perturbation noise accordingly. Specifically, if a client can tolerate more loss of privacy (*i.e.,* it has a larger ρ_i), it will performs a weaker gradient perturbation (smaller σ_i^2). Next, the explicit relationship between σ_i^2 and ρ_i is presented in Lemma 1.

Lemma 1. *In FL with DP, if client i with a privacy budget ρ_i adopts the Gaussian mechanism for gradient perturbation, the noise variance is $\sigma_i^2 = \frac{2\mathcal{L}^2}{\rho_i B_m}$, where $\mathcal{L}$ is the Lipschitz constant and B_m is the mini-batch size [35].*

Privacy Cost[3]**.** Even with the protection from DP, clients still suffer from a certain degree of privacy leakage, leading to a privacy cost. One client's privacy cost is determined by two factors [35]: privacy budget ρ_i, and privacy preference ζ_i. The privacy preference indicates how sensitive the client is to privacy leakage. Specifically, the privacy cost at round t, C_p^t, increases linearly with the number of iterations, which is modeled as:

$$C_p^t = \sum_{k \in \mathcal{K}} \tau^{k,t} \sum_{i \in S_k} \Delta_i \zeta_i \rho_i y_i^t \tag{3}$$

Important notations are listed in Table I.

TABLE I: Notations and Descriptions

Inputs	**Description**
$\mathcal{I}$	the set of candidate clients in FL
$\mathcal{K}$	the set of clusters
l_k	the cluster head of cluster k
S_k	the clients belongs to cluster k
ρ_i	the privacy budget of client i t
C_p^t	the privacy cost at global round t
t_i^{cmp}	computation time of client i for one intra-cluster aggregation
t_i^{com}	communication time of client i for one intra-cluster aggregation
t_k^{com}	communication time of cluster head l_k for one inter-cluster aggregation
n_k	Maximum selected number of clients per cluster per global round
Outputs	**Description**
x_i^k	whether client i belongs to cluster k
y_i^t	whether or not select client i at global round t
$\tau^{k,t}$	# of intra-cluster iterations for clients in cluster k at round t

Problem Formulation. Under the cost concern, the job completion time minimization problem is formulated as follows:

$$\text{minimize} \quad T_c \tag{4}$$

$$F(\omega^T) - F(\omega^*) \le \varepsilon, \tag{4a}$$

$$\sum_{t=1}^{T} C_p^t \le C^{max}, \tag{4b}$$

$$\sum_{i \in S_k} y_i^t \le n_k, \quad \forall k \in \mathcal{K}, \forall t \in T, \tag{4c}$$

$$y_i^t \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \forall t \in T, \tag{4d}$$

$$\tau^{k,t} \in \{1, 2, ..., \tau_{max}\}, \quad \forall k \in \mathcal{K}, \forall t \in T. \tag{4e}$$

Constraint (4a) guarantees the accuracy requirement ε, where $F(\omega^T)$ indicates the loss of the global model after T iterations, and $F(\omega^*)$ denotes the minimum loss. Constraint (4b) ensures that the overall privacy cost after global round T will not exceed the cost budget. Constraint (4c) ensures that no more than n_k clients are selected for each global round.

Challenges. i) Online. Due to the long-term privacy cost budget, it is difficult to make decisions without the future information. ii) Non-IID. It is hard to select clients to guarantee

[3]Because the computation/communication cost is proportional to the computation/communication time, which has been considered in the job completion time, here we only consider the privacy cost.

the model accuracy without any information on the client's data. iii) Inexplicit relationship. The optimization objective cannot be measured by any explicit functional relationship with decision variables.

IV. The Design of *MCFL*

A. Design Overview

To minimize the job completion time, we propose a two-stage clustered FL framework, *MCFL*:

i. Pre-clustering Stage. *MCFL* applies a MinMax K-means algorithm to construct the cluster topology based on the time of each client for one round intra-aggregation. *MCFL* first uses a weighted objective to approximate the objective of problem (1). Then, a MinMax K-means algorithm is adopted to determine the cluster topology.

ii. Trainning Stage. After the clustering, *MCFL* leverages a DRL approach based on SACD to solve problem (4). We first transform the original problem (4) into a Markov decision process (MDP). In the clustered FL scenario, the cloud server is considered as the agent. The agent can dynamically obtain the state of the environment s^t at each FL global round t. Given the state s^t, the agent takes the action a^t based on the policy π. The action decides the client selection and the number of iterations for intra-cluster aggregation in each cluster k. Here, the policy π is parameterized with θ (*e.g.*, the weights of the neural network for the agent). After the agent carries out the actions, the environment updates the next state s^{t+1} and turns out a reward r^t. Then the tuple $\{s^t, a^t, r^t, s^{t+1}\}$ is stored to the replay buffer $\mathcal{O}_b$, which is sampled by the agent for its actor and critic networks update. The agent adjusts the parameters of the actor and critic networks until convergence. After finishing the process of DRL network training, the trained DRL model can be used to decide the client selection and the number of iterations for intra-cluster aggregation.

B. Solving Pre-clustering Problem

Problem Reformulation. Inspiring by the MinMax K-means algorithm proposed in [33], we propose a clustering algorithm, *CA*, to tackle problem (1). Similar to [33], we first reformulate the objective function of problem (1) to the sum of the weighted variances, $\Omega_w = \sum_{k\in\mathcal{K}} \varphi_k^p \mathcal{V}_k$, where φ_k is a weight index (which will be defined later) and its value increases with the growth of the variance $\mathcal{V}_k$. The exponent p is a constant that takes the value in the range $[0, 1)$. In this way, $\sum_{k\in\mathcal{K}} \varphi_k^p \mathcal{V}_k$ mimics the behavior of the maximum variance criterion $\max_{k\in\mathcal{K}} \mathcal{V}_k$. The reformulated problem is:

$$\text{minimize} \quad \sum_{k\in\mathcal{K}} \varphi_k^p \mathcal{V}_k \tag{5}$$

$$(1a) - (1b),$$

$$\sum_{k\in\mathcal{K}} \varphi_k = 1, \quad \forall k \in \mathcal{K}, \tag{5c}$$

$$\varphi_k > 0, \quad 0 \le p < 1, \quad \forall k \in \mathcal{K}. \tag{5d}$$

Clustering Algorithm Design. We apply an iterative approach. At R-th iteration, given the weight index $\varphi_k^{(R-1)}$ and the mean $\mathbf{m}_k^{(R-1)}$ (which can be interpreted as the center of cluster k) from the $(R-1)$-th iterations, the value of x_i^k is updated. The client i is independently assigned to the closest cluster, *i.e.,* assigned to the cluster with the smallest weighted distance between the client i and the center $\mathbf{m}_k^{(R-1)}$. Then the assignment of the client i at R-th iteration is:

$$x_i^{k^{(R)}} = \begin{cases} 1, & k = \arg\min_{k'\in\mathcal{K}} (\varphi_{k'}^{(R-1)})^p \|t_i - \mathbf{m}_{k'}^{(R-1)}\|^2, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

After x_i^k is updated, to re-evaluate the cluster center $\{\mathbf{m}_k\}_{k\in\mathcal{K}}$, the equation of the derivative $\frac{\partial \Omega_w}{\partial \mathbf{m}_k} = 0$ is calculated. From the derivative, we can get the center of cluster k at iteration R:

$$\mathbf{m}_k^{(R)} = \frac{\sum_{i=1}^I x_i^k t_i}{\sum_{i=1}^I x_i^k}. \tag{7}$$

Next, given the clients' assignment $\{x_i^k\}_{k\in\mathcal{K}}$ and the center $\mathbf{m}_k$ of cluster k, the weight index $\varphi_k^{(R)}$ of cluster k at R-th iteration can be updated according to the equation of the derivative $\frac{\partial \Omega_w}{\partial \varphi_k} = 0$:

$$\varphi_k^{(R)} = \frac{\mathcal{V}_k^{\frac{1}{1-p}}}{\sum_{k'=1}^K \mathcal{V}_{k'}^{\frac{1}{1-p}}}. \tag{8}$$

Because $0 < p < 1$, the weight index φ_k becomes larger when the cluster variance $\mathcal{V}_k$ increases. To enhance the stability of the MinMax K-means algorithm, a memory effect is added to the weight index:

$$\varphi_k^{(R)} = \beta \varphi_k^{(R-1)} + (1-\beta) \frac{\mathcal{V}_k^{\frac{1}{1-p}}}{\sum_{k'=1}^K \mathcal{V}_{k'}^{\frac{1}{1-p}}}, 0 < \beta < 1, \tag{9}$$

where β is a constant and controls the influence of the previous weight index on the current update.

Algorithm 1 Clustering Algorithm (CA)

Input: register information of clients, K, p, β, ϵ, R_{max};
Output: Cluster assignments $\{x_i^k\}_{i\in\mathcal{I},k\in\mathcal{K}}$;

1: Initialize centers $\{\boldsymbol{m}_k^{(0)}\}_{k\in\mathcal{K}}$, $R = 0$, $\{\varphi_k^{(0)}\}_{k\in\mathcal{K}} = 1/M$;
2: calculate $\{t_i\}_{i\in\mathcal{I}}$ according to the register information of clients;
3: **repeat**
4: $R = R + 1$
5: **for** $i = 1$ to I **do**
6: **for** $k = 1$ to K **do**
7: Given the weights $\varphi_k^{(R-1)}$ and centers $\mathbf{m}_k^{(R-1)}$, update assignments $\{x_i^{k^{(R)}}\}_{i\in\mathcal{I}}$ according to Eq.(6);
8: **end for**
9: **end for**
10: **for** $k = 1$ to K **do**
11: update the centers $\{\mathbf{m}_k^{(R)}\}_{k\in\mathcal{K}}$ and weights $\{\varphi_k^{(R)}\}_{k\in\mathcal{K}}$ according to Eq. (9) and Eq.(7) respectively;
12: **end for**
13: calculate the $\Omega_w^{(R)}$;
14: **until** $\Omega_w^{(R)} - \Omega_w^{(R-1)} \le \epsilon$ or $R > R_{max}$
15: Select a stable client in each cluster as the cluster head;
16: **return** $\{x_i^k\}_{i\in\mathcal{I},k\in\mathcal{K}}$

Algorithm Details. We propose a clustering algorithm *CA* in Alg. 1 to solve problem (1). Firstly, the center $\{\boldsymbol{m}_k^{(0)}\}_{k\in\mathcal{K}}$ and the weight index $\{\varphi_k^{(0)}\}_{k\in\mathcal{K}}$ are initialized (line 1). Then

the time of one round intra-cluster aggregation for each client $\{t_i\}_{I\in\mathcal{I}}$ is calculated (line 2). Secondly, $\{x_i^{k^{(R)}}\}_{i\in\mathcal{I}}$ is updated according to $\{\boldsymbol{m}_k^{(R-1)}\}_{k\in\mathcal{K}}$ and $\{\varphi_k^{(R-1)}\}_{k\in\mathcal{K}}$ generated at the last iteration (lines 5-9). Thirdly, $\{\mathbf{m}_k^{(R)}\}_{k\in\mathcal{K}}$ and $\{\varphi_k^{(R)}\}_{k\in\mathcal{K}}$ are re-evaluated based on the updated clients' assignments $\{x_i^{k^{(R)}}\}_{i\in\mathcal{I}}$ (lines 10-12). Finally, the current $\Omega_w^{(R)}$ is calculated to evaluate the current cluster topology (line 13). Here ϵ is a tiny constant that serves as the termination criterion. When the difference of the weighted variance between two continuous iterations is less than ϵ or the iterations R is larger than the maximum iterations R_{max}, the updating process stops. Finally, the cluster heads are selected (line 15).

C. Deep Reinforcement Algorithm Design

Problem Transformation. To minimize the job completion time, problem (4) is transformed into a MDP, which is described by the tuple $\{\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma\}$. Here $\mathcal{S}$ denotes a set of states and $\mathcal{A}$ denotes a set of actions. The state transition function $\mathcal{P}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow[0,1]$ is used to compute the state transition probability $p(s^{t+1}|s^t,a^t)$, given current action a^t and state s^t. The reward r^t at global round t is computed by the reward function $\mathcal{R}:\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$ and future rewards are discounted by the factor $\gamma\in[0,1]$. The details of the state, action, and reward is presented as follows:

1) State $\mathcal{S}$: At t-th global round, the state of the clustered FL environment consists of: the local loss of each client $F(\omega_i^t)$, the current training time of each cluster $t_k^t=\max_{i\in S_k}(\tau^{k,t}\cdot t_i\cdot y_i^t)+t_k^{com}$ and the current global round t. Therefore, the state observed by the agent at global round t is represented by a vector $s^t=\{\{F(\omega_i^t)\}_{i\in\mathcal{I}},\{t_k^t\}_{k\in\mathcal{K}},t\}$.

2) Action $\mathcal{A}$: Once the state s^t is observed, the cloud server (*i.e.*, the agent) determines the action a^t at global round t for FL training. It first decides whether client i is selected at global round t. In addition, it calculates the number of iterations for intra-cluster aggregations for each cluster at global round t. Thus, the action space can be defined as $a^t=\{\{y_i^t\}_{i\in\mathcal{I}},\{\tau^{k,t}\}_{k\in\mathcal{K}}\}$.

3) Reward $\mathcal{R}$: Given the action a^t, the agent will receive a reward $r^t\in\mathcal{R}$ from the environment. Jointly considering the goal of minimizing job completion time and the MDP basic logic, the reward should decrease when the job completion time increases. We also add several penalty values in the reward to punish the agent for choosing actions that violate constraints (4a)-(4c). Therefore, the reward $r^t\in\mathcal{R}$ at global round t is defined as:

$$r^t=-T_c^t-p_1^t-p_2^t-p_3^t, \tag{10}$$

where p_1^t indicates the penalty that the current global model loss gap $F(\omega^t)-F(\omega^*)$ does not meet the accuracy requirements ε, and is proportional to the distance from the accuracy requirement. p_2^t denotes the penalty when violating the constraint (4b), and increases linearly with the growth of the privacy cost. When the constraint (4c) is violated, p_3^t is added as a penalty, and will increase according the number of extra selected clients. Otherwise, p_1, p_2 and p_3 are set to zero.

Design of SACD-based Algorithm. In the clustered FL scenario, it is challenging to model state transition probabilities for FL training, especially under the influence of non-IID data. Furthermore, high dimensional continuous state spaces and discrete action spaces lead to intractable convergence performance [36]. To overcome these challenges, we exploit the soft actor-critic with discrete actions (SACD) in our algorithm to handle the above MDP problem. The structure of SACD-based algorithm, *SAA*, is illustrated in Fig. 2, which combines the advantages of stochastic policy and off-policy (*i.e.*, reuse the previously experience for efficiency). The actor-critic architecture consists of an actor network, a Q-network and a target Q-network. The Q-network and the target Q-network both have a pair of critic networks. The use of two critic networks is to mitigate positive bias in the policy improvement step, which is beneficial to reduce overestimation. Furthermore, two critic networks can significantly speed up the training, especially on complex tasks [37]. The main idea of *SAA* is to obtain an optimal policy π^* that maximizes the entropy objective:

$$\pi_\theta^*=\arg\max_\pi\sum_{t=1}^{T}\mathbb{E}_{(s^t,a^t)}\sim\kappa_\pi[\gamma^t(r^t+\alpha\mathcal{H}(\pi(\cdot|s^t)))], \tag{11}$$

where $\mathcal{H}(\cdot)$ denotes the entropy of the policy π at state s^t and is calculated as $\mathcal{H}(\pi(\cdot|s^t))=-\log\pi(\cdot|s^t)$. κ_π is the distribution of trajectories induced by policy π. The temperature parameter α controls the relative importance between the entropy term and the reward. According to policy π, an action decision a^t is made by the actor for the current state. Instead of taking the output of the actor network as the action, we use the action processed by the output discretization. The clustered FL environment takes the action and gives back a reward r^t and then transfers to the next state s^{t+1}. The transition (*i.e.*, $\{s^t,a^t,r^t,s^{t+1}\}$) is stored into the replay buffer $\mathcal{O}_b$ for training. The actor and the critics will be trained according to the transitions sampled randomly from the replay buffer $\mathcal{O}_b$ until convergence.

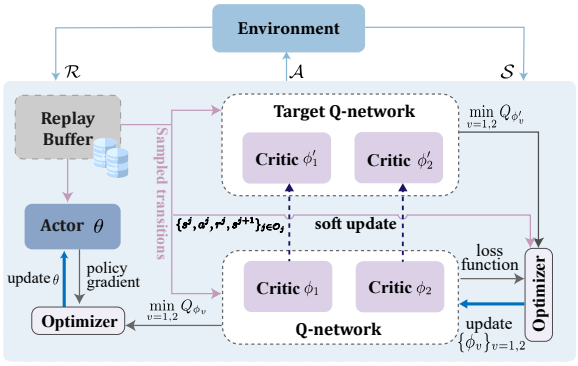


Fig. 2: The design of *SAA*

Parameter Updates. In the training process, the agent randomly samples a mini-batch transitions $\{s^j,a^j,r^j,s^{j+1}\}_{j\in\mathcal{O}^j}$ from the replay buffer and updates the SACD network parameters. The Q-networks update the parameters ϕ_v independently by minimizing the loss function, which is given by:

$$J_Q(\phi_v) = \mathbb{E}_{(s^j,a^j)\sim\mathcal{O}^j}[\frac{1}{2}(Q_{\phi_v}(s^j,a^j) - \hat{Q}^j)^2], \quad (12)$$

with the target Q-values:

$$\hat{Q}^j = r^j + \gamma(\min_{v=1,2} Q_{\phi'_v}(s^{j+1}) - \alpha \log \pi_\theta(s^{j+1})). \quad (13)$$

The target Q-network $\{Q_{\phi'_v}\}_{v=1,2}$ copies periodically from the Q-network $\{Q_{\phi_v}\}_{v=1,2}$. Therefore, ϕ_v can be adjusted in the direction of $\nabla J_{\phi_v}(\phi_v)$, which is based on the stochastic gradient descent (SGD) method.

Moreover, the pair of critics of the Q-network use Q-value functions $(Q_{\phi_1}(\cdot), Q_{\phi_2}(\cdot))$ to evaluate the actions made by the actor. With the Q-values of the Q-networks, the actor network can be learned by minimizing the objective function:

$$J_\pi(\theta) = \mathbb{E}_{s^j\sim\mathcal{O}^j}[\pi^j(s^j)^T[\alpha \log(\pi_\theta(s^j)) - \min_{v=1,2} Q_{\phi_v}(s^j)]], \quad (14)$$

and the temperature parameter α can be optimized by minimizing the entropy objective function:

$$J(\alpha) = \pi^j(s^j)^T[-\alpha \log(\pi_\theta(s^{j+1})) - \hat{H}], \quad (15)$$

where $\hat{H}$ is a constant, and denotes the target entropy. Same as the Q-networks, the parameter of the actor network and temperature can also be updated via the SGD method[4].

Algorithm 2 Soft Actor-critic with Discrete Actions Based Algorithm (SAA)

1: Initialize the weights of the actor network θ, the Q-networks ϕ_1, ϕ_2 and the target Q-networks $\phi'_1 \leftarrow \phi_1$, $\phi'_2 \leftarrow \phi_2$, the replay buffer $\mathcal{O}_b$, hyperparameters γ, α, δ, ℓ_Q, ℓ_π, ℓ_α, $\hat{H}$;
2: **for** $ep = 1$ to E **do**
3: Initialize the clustered FL environment, and receive the initial state s^1
4: **for** $t = 1$ to T **do**
5: Get the discrete actions a^t based on the current state;
6: Apply actions a^t to the clustered FL environment, and obtain the reward r^t turned out from the clustered FL environment;
7: Observe the next state s^{t+1}, then store $\{s^t, a^t, r^t, s^{t+1}\}$ in the replay buffer $\mathcal{O}_b$;
8: Randomly sample a mini-batch of $\mathcal{O}^j$ experiences $\{s^j, a^j, r^j, s^{j+1}\}_{j\in\mathcal{O}^j}$ from replay buffer $\mathcal{O}_b$;
9: Update the Q-network: $\phi_v \leftarrow \phi_v - \ell_Q \nabla\phi_v J_Q(\phi_v), v = 1, 2$;
10: Update the weights of actor network: $\theta \leftarrow \theta - \ell_\pi \nabla\pi J_\pi(\theta)$;
11: Update the temperature parameter: $\alpha \leftarrow \alpha - \ell_\alpha \nabla\alpha J(\alpha)$;
12: Soft update the target Q-network: $\phi'_v \leftarrow \delta\phi_v + (1-\delta)\phi'_v, v = 1, 2$.
13: **end for**
14: **end for**

***Algorithm Details*.** We propose an SACD based algorithm, *SAA*, in Alg. 2 to solve problem (4). It works as follows:

- *Firstly,* the parameters of the actor, the Q-networks, and the target Q-networks, which activate and evaluate the actions of the agent, are initialized (line 1).
- *Secondly,* the agent observes the state information $s^t = \{\{l_i^t\}_{i\in\mathcal{I}}, \{l_k^t, t_{r,k}^t\}_{k\in\mathcal{K}}, t\}$ from the external clustered FL environment, and determines the discrete actions $a^t = \{\{y_i^t\}_{i\in\mathcal{I}}, \{\tau^{k,t}\}_{k\in\mathcal{K}}\}$ based on current policy π^t (line 6). Note that some tricks are used to improve the effectiveness. The input (*i.e.*, state of the agent) should be normalized to avoid overfitting. After the discrete actions are conducted, the clustered FL environment interacts with the agent, and then it feedbacks the corresponding reward r^t to the agent (line 7). After one round of inter-cluster aggregation, the clustered FL environment switches to the next state s^{t+1}, and stores the tuple $\{s^t, a^t, r^t, s^{t+1}\}$ in the replay buffer $\mathcal{O}_b$, which contains the experiences of historical tuples (line 8).
- *Thirdly,* throughout the entire episode, the agent conducts the following procedures to update its actor and critic networks. The agent samples the random mini-batch of $\mathcal{O}^j$ transitions among $\mathcal{O}_b$ (line 9). Hence, the parameters θ, ϕ_v, α of the actor, the Q-networks, and the temperature parameter can be updated by the gradient descent method (lines 10-12). ℓ_π, ℓ_Q and ℓ_α denote the learning rate of the actor, the Q-networks, and the temperature parameter respectively.
- *Finally,* the target Q-network parameters are updated under the concept of soft update (line 13). δ denotes the update factor of the target Q-networks.

[4]The value of the parameters are described in Sec. V.

V. Evaluation

A. Experiment setup

System settings. We consider a clustered FL system with 40 clients, 4 cluster heads and a cloud server. Each client employs mini-batch SGD for training with batch size 16 both for Logistic and LeNet models, and clients use the standard dataset MNIST [38] to train. The number of local iterations for each client is uniformly distributed in the range $[1, 5]$. The number of iterations for inter-cluster aggregation is set to 1. The learning rate of the clustered FL is 0.01 and decays exponentially at a rate of 0.995 with every epoch. The clients are randomly distributed in a square with an area of $500m \times 500m$. For the communication parameters, similar to [30], the channel gain is $g_i = -128.1 - 37.6\log_{10}(d_i)$, where d_i denotes the transmission distance between the client i and the cluster head. For the communication channel between the clients and the cluster head, the bandwidth of clients B_i is distributed in $[1, 10]$ MHz. The transmit power p_i is distributed in $[20, 40]$ dBm, and the noise power N_0 is -104 dBm. For the local computation model, the parameter U_i is uniformly distributed in $[1, 3] \times 10^4$ cycles/sample, and CPU cycle frequency f_i is uniformly distributed in $[1, 2]$ GHz [39]. For the communication time of uploading the cluster model to the cloud server, we assume it is 10 times larger than that of uploading the local model to the cluster heads [19].

For the differential privacy, the privacy budget ρ_i of clients is uniformly distributed in the range $[1, 1.5]$, and the privacy preference ζ_i is uniformly distributed in the range $[100, 200]$. The cost budget C^{max} is set to 6×10^5 [40].

For the DRL, the input dimension of the actor and the critic is the dimension of the state. The network of actor and

critics includes two layers. The output dimension of the first and second layer of the actor are both 128, and the output dimension of the first and second layer of the critics are 128 and 64 respectively. Other important hyperparameters are listed in Table II.

TABLE II: Hyperparameters of *SAA*

Parameter	Value
Total episode EP for Logistic	800
Total episode EP for LeNet	400
Discounting factor γ	0.99
soft update factor δ	0.01
Size of replay buffer $\mathcal{O}_b$	5000
Temperature factor init.	1.
Learning rate ℓ_Q, ℓ_π, ℓ_α	0.0001
Optimizer	Adam
Activation function	ReLU

Benchmarks. To evaluate the performance of *MCFL*, the following three benchmarks are compared.

- *FEDAVG* [1]: *FEDAVG* is the one-layer traditional FL algorithm.
- *H-BASE* [19]: *H-BASE* is a clustered FL algorithm, which randomly selects clients and trains on a fixed iterations of intra-cluster aggregation (*i.e.*, $\tau^{k,t} = 5$).
- *HACCS* [18]: *HACCS* is a clustered FL algorithm, which groups clients based on their data distribution. Each cluster is assigned a probability computed according to the weighted accuracy and the training time. It chooses the cluster based on the the probability, and selects the client with the shortest training time from the cluster. Note that $\tau^{k,t}$ of *HACCS* is set to 5.

B. Evaluation Results

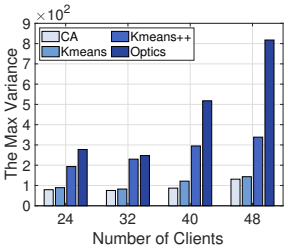


Fig. 3: Performance of *CA*.

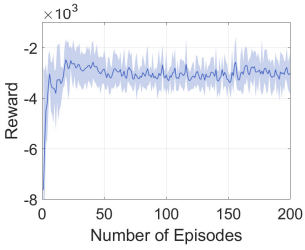


Fig. 4: Training on SAA.

***Performance of* CA.** Alg. 1 (*CA*) is compared to three classic clustering algorithms: the based K-means [41], the K-means++ [42] and the Optics [43]. The value of the maximum intra-cluster variance under different numbers of clients (I) is shown in Fig. 3. It can be observed that *CA* achieves the smallest maximum variance, which means *CA* can efficiently reduce the straggler effect within the cluster. *CA* reduces the maximum intra-cluster variance by 15% compared with K-means, 65% compared with K-means++ and 80% compared with Optics. In addition, the value of the maximum intra-cluster variance increases when I increases.

***Convergence of* SAA.** Fig. 4 depicts the convergence curve in terms of average reward during the training process of *SAA*. An episode begins with the initialization of a FL job and ends when the job achieves the target accuracy or reaches the maximum global round. Clearly, as the episode increases, the average reward value of *SAA* grows and eventually converges at around 50 episodes.

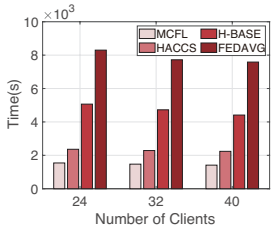


Fig. 5: Job completion time under different I.

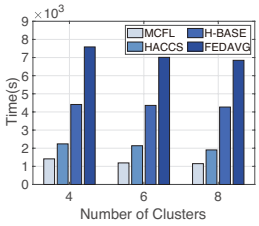
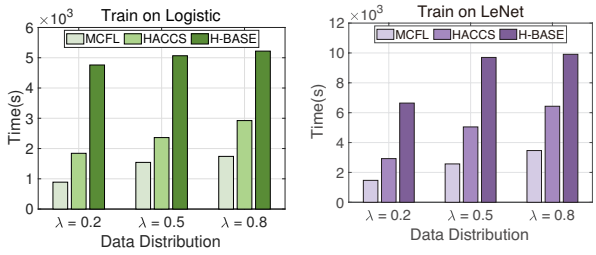


Fig. 6: Job completion time under different K.

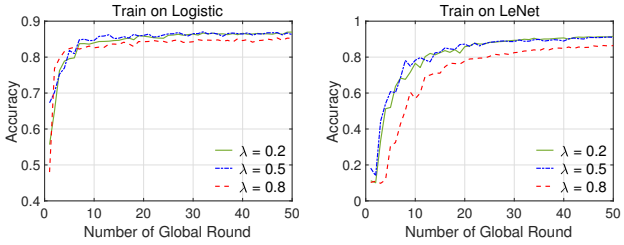
Clustered vs. non-clustered. As shown in Fig. 5 and Fig. 6, we compare the job completion time of the traditional *FEDAVG* method with three clustered FL algorithms, *H-BASE*, *HACCS* and *MCFL*, under different I and different K. Using the MNIST dataset, each client trains a Logistic model. In Fig. 5, when the number of clusters $K = 4$, we plot the job completion time under different numbers of clients (I) when the desired accuracy requirement is reached. It shows that the clustered FL leads to a significant reduction of job completion time with the same desired accuracy requirement. In Fig. 6, the number of clients is fixed, *i.e.*, $I = 40$, and the job completion time under different numbers of clusters (K) is compared. We can also observe the similar results. Because the non-clustered *FEDAVG* suffers more from stragglers who slow down the job completion time. Specifically, the job completion time of *FEDAVG* is 2 times that of *H-BASE*, 3 times that of *HACCS* and 5 times that of *MCFL*.



(a) (b)

Fig. 7: Job completion time under different λ.

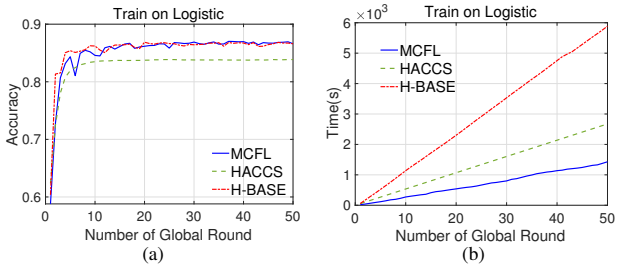
Performance of job completion time. Fig. 5 and Fig. 6 also show the performance of *MCFL* on job completion time under different numbers of clients (I) and different numbers of clusters (K). We can see that *MCFL* outperforms the two benchmark algorithms, *H-BASE* and *HACCS*, and reduces the job completion time by 35% compared with *HACCS* and by 70% compared with *H-BASE*. Besides, as the number of clients increases, the overall job completion time tends to decrease. One reason is that a larger number of clients provides a higher probability of selecting a client with both good data quality and short training time. Furthermore, the job completion time of *MCFL* decrease slightly with the increase of K. Since fewer clients are grouped in one cluster with the increase of K, which leads to a lower probability of having stragglers in one cluster.



(a) (b)

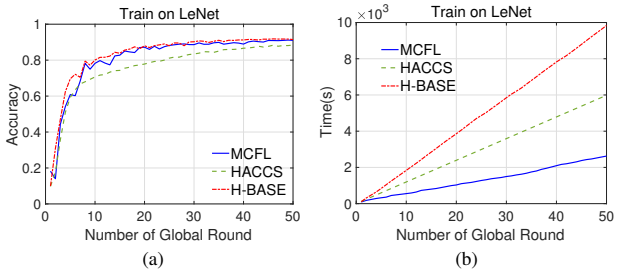
Fig. 8: Accuracy under different λ.

Next, Fig. 7 shows the job completion time under different data distribution λ. For clarity, we use one specific metric $\lambda \in [0, 1]$ to represent different levels of non-IID data. For instance, $\lambda = 0.2$ indicates that 20% of the data belongs to one corresponding label and the remaining 80% of the data belongs to other labels. Under this setting, we exploit three levels of non-IID data, 0.2, 0.5 and 0.8, which exhibit an increasing non-IID intensity. We can see that *MCFL* has the shortest job completion time and can adapt to different non-IID levels, both in Logistic and LeNet. As the level of non-IID increases, the job completion time shows an upward trend. This is because the model convergence speed is slower when λ increases.



(a) (b)

Fig. 9: Accuracy comparison with the benchmark algorithms on Logistic.



(a) (b)

Fig. 10: Accuracy comparison with the benchmark algorithms on LeNet.

Performance of accuracy. Fig. 8∼Fig. 12 illustrate the upcurve of the model accuracy for 50 rounds, where Fig. (9b) and Fig. (10b) are the corresponding accumulative time curves. For Logistic, Fig. (8a) shows that larger λ leads to the smaller accuracy achieved by *MCFL*. As for LeNet, Fig. (8b) demonstrates that a larger λ even slows down the convergence speed. Fig. 9 and Fig. 10 demonstrate the increase of accuracy for 50 global rounds as well as the corresponding completion time. On both Logistic and LeNet, it shows that *MCFL* can achieve a similar accuracy compared to *H-BASE* at the speed of 70% faster than *H-BASE* in each global round. And *MCFL* is able to achieve 5% higher accuracy than *HACCS* while saving 35% of the job completion time. Besides, Fig. 11 and Fig. 12 exploit the accuracy fluctuations under different numbers of clusters (K) and clients (I). It is observed that accuracy improves slightly as the number of clusters increases. For inter-cluster aggregation, more clusters cause more cluster models to be uploaded to the cloud server at the same time, thus improving the accuracy of the global model. However, when the I increases, it does not show a significant increasing trend. This is because the probability of selecting stragglers will increase with the growth of the number of clients. As a result, when $I = 40$, the model converges at a lower speed. In general, *MCFL* can significantly reduce the job completion time while reaching the accuracy requirement.

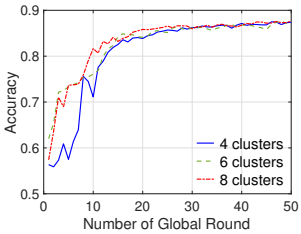


Fig. 11: Accuracy under different K using Logistic.

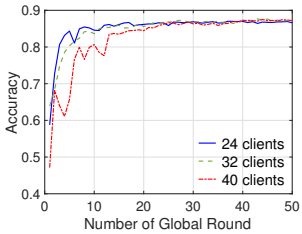


Fig. 12: Accuracy under different I using Logistic.

Differential Privacy. Fig. 13 illustrates the privacy cost under different λ. The privacy cost generated by training on Logistic and LeNet increases when the value of λ increases but it will not exceed C^{max} (indicated by the purple line). Since higher non-IID levels require more iterations to reach convergence, resulting a higher privacy cost. The privacy cost is hardly affected by model differences. Furthermore, we study the effect of privacy budget in the FL training process, shown in Fig. 14. We test the variation of accuracy on the Logistic and LeNet model respectively, where the privacy budget of clients ρ_i is distributed in the range $[0, 0.5]$ and range $[1, 1.5]$. It shows that the privacy budget of clients is inversely proportional to model quality because the noise added to the local model is larger when the privacy budget increases. Moreover, the Logistic model is more obviously disturbed by the DP mechanism.

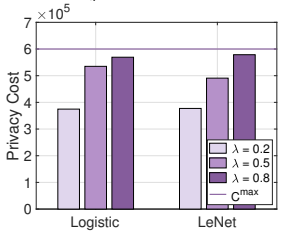


Fig. 13: Privacy cost under different λ.

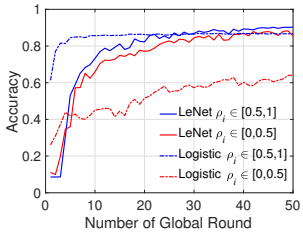


Fig. 14: Effect of ρ_i.

VI. Conclusion

We observe that there are two main factors that affect the FL job completion time: non-IID data of clients and

the straggler effect. To this end, we propose a clustered FL framework, *MCFL*, to minimize the FL job completion time while guaranteeing the FL model accuracy. *MCFL* consists of the following two stages. *MCFL* first constructs clusters, each containing clients with similar computing and communications capabilities to mitigate the straggler among clients in the cluster. Second, at each global round, *MCFL* selects a suitable subset of clients with high quality data for each cluster to handle non-IID data, and further determines the number of iterations for intra-cluster aggregation for each cluster to reduce the effect of stragglers among clusters. The extensive testbed experiments based on real-world data verify that *MCFL* can significantly reduce the job completion time, compared with existing FL frameworks.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of PMLR AISTATS*, 2017, pp. 1273–1282.

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

[3] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," in *Proc. of IEEE ICASSP*, pp. 6523–6527.

[4] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *Proc. of PMLR ICML*, 2019.

[5] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for on-device federated learning." 2019.

[6] A. Khaled, K. Mishchenko, and P. Richtárik, "Tighter theory for local sgd on identical and heterogeneous data," in *Proc. of AISTATS*. PMLR, 2020, pp. 4519–4529.

[7] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Transactions on Mobile Computing*, 2022.

[8] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, pp. 1996–2009, 2021.

[9] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *Proc. of IEEE INFOCOM*, 2021.

[10] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[11] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proc. of ICLR 2020*, 2020.

[12] Z. Ji, L. Chen, N. Zhao, Y. Chen, G. Wei, and F. R. Yu, "Computation offloading for edge-assisted federated learning," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 9330–9344, 2021.

[13] X. Lu, Y. Liao, P. Lio, and P. Hui, "Privacy-preserving asynchronous federated learning mechanism for edge network computing," *IEEE Access*, vol. 8, pp. 48 970–48 981, 2020.

[14] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," in *Proc. of ACM/IEEE DAC*, 2021, pp. 997–1002.

[15] L. Zhang, D. Wu, and X. Yuan, "Fedzkt: Zero-shot knowledge transfer towards resource-constrained federated learning with heterogeneous on-device models," *arXiv preprint arXiv:2109.03775*, 2021.

[16] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. of PMLR*, 2021, pp. 2089–2099.

[17] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[18] J. Wolfrath, N. Sreekumar, D. Kumar, Y. Wang, and A. Chandra, "Haccs: Heterogeneity-aware clustered client selection for accelerated federated learning," in *Proc. of IEEE IPDPS*, 2022.

[19] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. of ICC*, 2020, pp. 1–6.

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

[21] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *Proc. of IEEE Big Data*, 2020.

[22] Y. Chen, S. Biookaghazadeh, and M. Zhao, "Exploring the capabilities of mobile devices in supporting deep learning," in *Proc. of ACM/IEEE SEC*, 2019.

[23] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *Proc. of IEEE INFOCOM*, 2020.

[24] C. Chen, Q. Weng, W. Wang, B. Li, and B. Li, "Semi-dynamic load balancing: Efficient distributed learning in non-dedicated environments," in *Proc. of ACM SoCC*, 2020.

[25] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[26] J. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," *Advances in Neural Information Processing Systems*, vol. 34, pp. 840–851, 2021.

[27] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *in Proc. of IEEE IJCNN*, 2020.

[28] N. Wang, R. Zhou, L. Su, G. Fang, and Z. Li, "Adaptive clustered federated learning for clients with time-varying interests," in *Proc. of IWQOS*, 2022, pp. 1–10.

[29] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. of IEEE INFOCOM*, 2021.

[30] W. Zhang, D. Yang, W. Wu, H. Peng, N. Zhang, H. Zhang, and X. Shen, "Optimizing federated learning in distributed industrial iot: A multi-agent approach," *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 3688–3703, 2021.

[31] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, "Adaptive federated learning and digital twin for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 5605–5614, 2020.

[32] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[33] G. Tzortzis and A. Likas, "The minmax k-means clustering algorithm," *Pattern Recognition*, vol. 47, pp. 2505–2516, 2014.

[34] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," *Theoretical Computer Science*, vol. 442, pp. 13–21, 2012.

[35] P. Sun, X. Chen, G. Liao, and J. Huang, "A profit-maximizing model marketplace with differentially private federated learning," in *Proc. of IEEE INFOCOM*, 2022.

[36] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[37] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[38] *The MNIST Dataset*, 1998, http://yann.lecun.com/exdb/mnist/.

[39] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2020.

[40] P. Sun, H. Che, Z. Wang, Y. Wang, T. Wang, L. Wu, and H. Shao, "Pain-fl: Personalized privacy-preserving incentive for federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 3805–3820, 2021.

[41] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 3, pp. 433–439, 1999.

[42] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[43] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.