CASA: Clustered Federated Learning with Asynchronous Clients

Boyi Liu
SKLCCSE Lab
Beihang University
Beijing, China
boyliu@buaa.edu.cn

Yiming Ma
SKLCCSE Lab
Beihang University
Beijing, China
yimingma@buaa.edu.cn

Zimu Zhou
School of Data Science
City University of Hong Kong
Hong Kong, China
zimuzhou@cityu.edu.hk

Yexuan Shi
SKLCCSE Lab
Beihang University
Beijing, China
skyxuan@buaa.edu.cn

Shuyuan Li
SKLCCSE Lab
Beihang University
Beijing, China
lishuyuan@buaa.edu.cn

Yongxin Tong
SKLCCSE Lab
Beihang University
Beijing, China
yxtong@buaa.edu.cn

Abstract

Clustered Federated Learning (CFL) is an emerging paradigm to extract insights from data on IoT devices. Through iterative client clustering and model aggregation, CFL adeptly manages data heterogeneity, ensures privacy, and delivers personalized models to heterogeneous devices. Traditional CFL approaches, which operate synchronously, suffer from prolonged latency for waiting slow devices during clustering and aggregation. This paper advocates a shift to asynchronous CFL, allowing the server to process client updates as they arrive. This shift enhances training efficiency yet introduces complexities to the iterative training cycle. To this end, we present CASA, a novel CFL scheme for Clustering-Aggregation Synergy under Asynchrony. Built upon a holistic theoretical understanding of asynchrony's impact on CFL, CASA adopts a bi-level asynchronous aggregation method and a buffer-aided dynamic clustering strategy to harmonize between clustering and aggregation. Extensive evaluations on standard benchmarks show that CASA outperforms representative baselines in model accuracy and achieves 2.28-6.49× higher convergence speed.

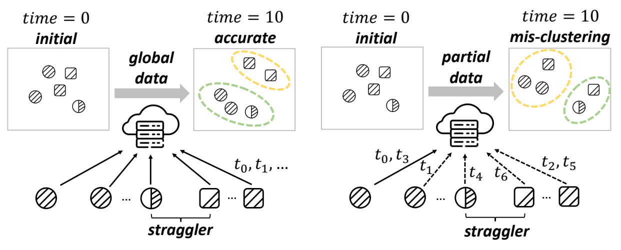
CCS Concepts

• **Computing methodologies → Learning paradigms**.

Keywords

Clustered Federated Learning; Asynchronous Federated Learning; Sparse Training

ACM Reference Format:
Boyi Liu, Yiming Ma, Zimu Zhou, Yexuan Shi, Shuyuan Li, and Yongxin Tong. 2024. CASA: Clustered Federated Learning with Asynchronous Clients. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671979



(a) Synchronous CFL **(b) Asynchronous CFL**

Figure 1: Asynchrony can lead to mis-clustering, which may further impair federated training of cluster-wise models.

1 Introduction

Clustered federated learning (CFL) is a promising solution to harness the *decentralized*, *heterogeneous* data of IoT devices for collective intelligence. In CFL, devices (*clients*) with similar data distributions are grouped to train cluster-wise models under *server* coordination, while keeping their datasets localized [13, 23, 26, 28, 36]. This strategy is particularly effective in IoT applications, where the data is often heterogeneous yet exhibits natural *clusterability* [18, 26]. For instance, human activities vary across users but may share notable spatiotemporal similarities [29]. By leveraging the inherent clusterability of heterogeneous data, CFL not only simplifies operations but also creates accurate, personalized models, with widespread applicability in smart homes [43], mobile healthcare [26], and intelligent transportation [34].

Despite CFL's efficacy in handling *data heterogeneity*, it struggles when confronting *system heterogeneity* inherent in the diverse computation and communication capabilities of IoT devices [44]. This problem stems from the *synchronous* operations, where the server awaits simultaneous updates from all clients for clustering and training [13, 23, 26, 28, 36]. Consequently, slow devices, often termed *stragglers*, force the server into waiting states, leading to extended *training latency*. For example, synchronous aggregation may take 3-12× longer to reach the target accuracy in presence of stragglers that are 5× slower than others [19].

A promising solution is to integrate *asynchrony* into CFL, where the server processes client updates as they arrive, which have been adopted in federated training of a single model [5, 6, 12, 19, 41].

Although such asynchronous mode eliminates long waiting time, it may severely undermine the effectiveness of CFL. As illustrated in Fig. 1, integrating asynchrony means clustering clients with partial and outdated information, which not only induces clustering errors, but also deteriorate the training of cluster-wise models. Specifically, asynchronous CFL faces the following challenges.

- *How to guarantee effective CFL with asynchronous clients?* CFL iteratively optimizes client clustering and model aggregation [13, 23, 26, 28, 36], which converges in the synchronous setting [23]. Yet it is unknown whether such *convergence* still holds in asynchronous environments.
- *How to adapt client clustering and model aggregation for asynchronous CFL?* Asynchrony necessitates *staleness* management in both clustering and aggregation. Although staleness control has been studied in federated learning without clustering, *e.g.*, by decaying outdated updates [7, 19, 27, 41], these designs are inapplicable to CFL due to the extra interplay between clustering and aggregation.

In this paper, we present CASA (**C**lustering-**A**ggregation **S**ynergy under **A**synchrony), a new CFL scheme for asynchronous clients. We analyze the impact of asynchrony on client clustering and model aggregation separately, as well as on their interplay, via a unified analytical framework. We further derive the conditions for convergent CFL in asynchronous environments. On this basis, we develop a buffer-aided dynamic clustering algorithm and a bi-level asynchronous aggregation scheme for effective and efficient asynchronous CFL. In addition, we also harness sparse training to actively mitigate the impact of stragglers. Evaluations on standard benchmarks show that our methods achieve comparable accuracies to synchronous CFL methods but converge faster by 2.28-6.49×. Compared with prior asynchronous schemes, we improve the accuracy by up to 30.34% and are up to 39.11× faster to reach the target accuracy.

Our main contributions are as follows:

- To our knowledge, this is one of the first work on clustered federated learning in a fully asynchronous context, promoting the CFL deployment to heterogeneous IoT devices.
- We present CASA, which features both theoretical analysis and practical server-side designs for fast and accurate CFL in asynchronous environments.
- Evaluations show that CASA is both more accurate and faster than the state-of-arts, which holds promise to simultaneously address data and system heterogeneity in federated learning.

2 Related Work

Clustered Federated Learning. CFL alleviates *data heterogeneity* in FL by grouping clients with similar data distributions, thereby enhancing homogeneous learning within clusters [13, 23, 26, 28, 36]. Due to its simplicity and efficacy, this tactic prevails in personalization in federated learning [4, 35], among other strategies [11, 33, 46]. Research on CFL explores client similarity measures [22, 26, 28] and clustering algorithms [2, 18, 22, 36, 42] to improve clustering accuracy and enhance its interplay with model training [13, 23, 42]. Common client similarity metrics include cosine similarity [28, 36, 42], Euclidean distance [2, 22], KL divergence [26], etc. The clustering algorithms span from naive k-means [10, 22, 26] to hierarchical clustering [2, 18, 20, 28, 42]. For instance, CFL [28] bi-partitions clients into clusters until the training stabilizes. IFCA [13] iteratively refines the clusters via training loss minimization. ICFL [42] adopts both incremental clustering and spectral clustering to dynamically discover the clustering structure. These studies operate in *synchronous* settings, leaving the coherence between clustering and training in the *asynchronous* context unexplored.

Our work adopts the standard cosine distance of model weights to measure client similarity [28, 36, 42], and focus on the *clustering algorithms* that function under asynchrony. A few proposals explored the semi-asynchronous case [49]. However, a comprehensive analysis on the interactions between clustering and training in fully asynchronous environments is missing.

Asynchronous Federated Learning. AFL enhances the efficiency of FL in face of *system heterogeneity.* While classic FL relies on synchronous model aggregation, introducing considerable latency due to delays from stragglers, AFL activates model aggregation upon receiving client updates, effectively minimizing idle wait times [5, 6, 12, 19, 41]. Semi-asynchronous variants have also been proposed to further reduce the client-server communication overhead [31, 40, 49]. However, asynchronous aggregation introduces stale gradients, which can destabilize learning or even cause model divergence [5, 19, 40, 41]. Accordingly, AFL algorithms often *decay* stale updates during model aggregation [7, 19, 27, 41] to mitigate their impacts on convergence. For example, FedAsync [41] first introduces the decay function to combat stale updates. FedBuff [25] buffers recent updates and aggregates the averaged gradients into global model. PORT [30] considers both divergence of model updates and staleness when updating model parameters. TimelyFL [47] adjusts the partial training rate to boost slow devices and reduce staleness. FedASMU [19] distributes the fresh global model to clients during training to control staleness, accompanied with a time-aware decay function to ensure convergence.

Our work also focuses on managing staleness, but for CFL rather than the generic FL that merely trains a single global model. Due to the unique interplay between clustering and training, we design a new decay function that benefits clustering and training and ensures their synergy. We also hitchhike the decay function for sparse training, which actively controls the impact of stragglers.

3 Problem Statement

Clustered federated learning [13, 23, 26, 28, 36]is a pivotal solution to handle non-IID data in federated learning. It groups clients based on the similarity of their data distribution. Within each cluster, clients collaboratively learn a shared model while the raw data remains locally on device.

CFL Workflow. In a typical CFL framework, clients learn cluster-wise models via *bi-level* optimization, iteratively minimizing the *clustering error* and *training loss* till model convergence [22, 23, 42]. This process operates in three steps per *round* (see Fig. 2): *(i) Local Training* (L-phase), in which clients perform local model training and upload the local updates to the server; *(ii) Client Clustering* (C-phase), where the server groups clients into clusters according to their uploaded model weights; *(iii) Model Aggregation* (A-phase), wherein client model weights are aggregated within each cluster, followed by sending the updated global (cluster-wise) model to

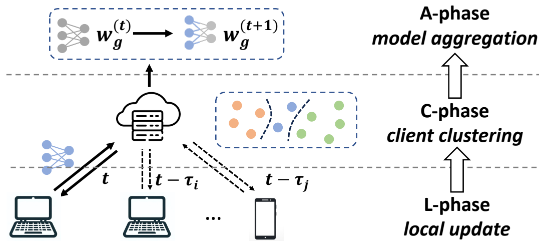


Figure 2: A typical CFL framework, which consists of local update (L-phase), client clustering (C-phase), and model aggregation (A-phase). Asynchrony induces outdated local updates to both the C- and A-phases.

clients. Importantly, the workflow is *synchronous*, where the server waits updates from all clients for clustering and aggregation.

Formally, assume n clients $\{c_1, c_2, \cdots, c_n\}$ with local dataset $\{D_1, D_2, \cdots, D_n\}$ are grouped into K clusters $\{u_1, u_2, \cdots, u_K\}$. All clients in cluster u_k share and collaboratively train a global model $w_{g,k}$ to minimize the following *training objective* $\mathcal{P}$:

$$\min_{w_{g,1},\ldots,w_{g,K}} \mathcal{P} = \sum_{k=1}^{K} \sum_{c_i \in C_k} \frac{|D_i|}{|D|} \mathbb{E}[\mathcal{L}(w_{g,k}; D_c)] \tag{1}$$

where C_k denotes the clients in cluster u_k. Meanwhile, CFL also minimizes the following *clustering error* $\mathcal{H}$:

$$\min_{w_{g,1},\ldots,w_{g,K},w_1,\ldots,w_n} \mathcal{H} = \sum_{k=1}^{K} \sum_{c_i \in C_k} \frac{|D_i|}{|D|} \|w_i - w_{g,k}\|_2^2 \tag{2}$$

where $w_{g,k}$ is the global model of cluster u_k, and w_i is the local parameters hold by client c_i.

CFL under Asynchrony. As mentioned in Sec. 1, we address *asynchronous* CFL to enhance training efficiency amidst stragglers. Adapting to asynchrony calls for significant modifications to both clustering and aggregation.

- *Client Clustering (C-phase)*: Prior CFL methods [13, 23, 26, 28, 36] base clustering decisions on full client availability. The asynchronous clients invalidate this assumption, necessitating new strategies for accurate and timely clustering.
- *Model Aggregation (A-phase)*: Stale updates from slower clients hinders the training convergence and accuracy [5, 19, 40, 41]. Asynchronous aggregation can manage staleness via weight decay [7, 19, 27, 41]. Yet its effectiveness remains unexplored when an extra clustering stage is involved.

Assumptions and Scope. We focus on *server-side* solutions to minimize interventions to clients. We use FedAvg [24] as the basic optimizer given its pervasive adoption, but our strategies should also apply to other optimizers *e.g.*, FedProx [17]. While privacy is vital in CFL, our primary goal is to improve training efficiency and accuracy, keeping privacy at levels akin to FedAvg. Explorations on specific attacks and defenses are beyond our scope.

4 Method

This section presents CASA (**C**lustering-**A**ggregation **S**ynergy under **A**synchrony), an asynchronous CFL scheme. We analyze the

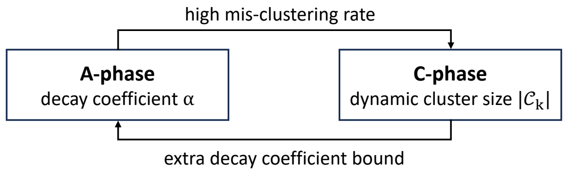

high mis-clustering rate
A-phase
decay coefficient α
C-phase
dynamic cluster size $|\mathcal{C}_k|$
extra decay coefficient bound

Figure 3: Impact of asynchrony on CFL.

impact of asynchrony on the generic CFL framework (Sec. 4.1), and propose new designs on model aggregation (Sec. 4.2) and client clustering (Sec. 4.3) to retain their synergy. Finally, we further enhance CASA by incorporating sparse training (Sec. 4.4).

4.1 Understanding Asynchronous CFL

Although the bi-level optimization for CFL (Sec. 3) converges in the synchronous setting [23], it is unknown whether such clustering-aggregation synergy holds in asynchronous environments. As depicted in Fig. 3, asynchrony affects both the A- and C-phases as well as their interplay. This necessitates the exploration of new requirements on clustering and aggregation to preserve their coherence.

Direct Impact. Asynchrony in CFL brings critical modifications to both client clustering and model aggregation:

- *C-phase*: Measuring client similarity becomes complex with partial client information. In synchronous settings, similarity between clients c_i and c_j in round t might be computed as $A_{ij} = \cos(w_i^{(t)}, w_j^{(t)})$. However, with asynchronous updates, this measure shifts to $A'_{ij} = \cos(w_i^{(t-\tau_i)}, w_j^{(t-\tau_j)})$, based on model parameters received in different rounds, *i.e.*, $t - \tau_i \neq t - \tau_j$. This leads to a deviation from the actual similarity of local data distributions between c_i and c_j due to misaligned global model $w_{g,k}^{(t-\tau_i)}$ and $w_{g,k}^{(t-\tau_j)}$.
- *A-phase*: In asynchronous aggregation, a decay coefficient α is introduced to manage staleness [7, 19, 27, 41]. The aggregation process becomes:

$$w_{g,k}^{(t+1)} = (1-\alpha) w_{g,k}^{(t)} + \alpha w_i^{(t-\tau)} \tag{3}$$

where the decay coefficient α is crucial for the convergence of model training.

Compound Impact. The changes in individual steps also affect the bi-level framework's ability to optimize the training and clustering objectives *i.e.*, Eq. (1) and Eq. (2), as our analysis shows.

Theorem 1. *(Clustering Error under Asynchrony). When clustering relies on a similarity matrix A' derived with asynchronous model parameters, the mis-clustering rate p is bounded by:*

$$p = O(\lambda\alpha\sqrt{\sum_{i=1}^{n}(\sum_{j=1}^{n}\|\tau_i - \tau_j\|^2)}) \tag{4}$$

where $\lambda = \eta Q \theta U$, and η is the learning rate, Q is the local training steps, U is the upper bound of gradient, θ is the upper bound of staleness (details in Appendix A.1.1).

Theorem 1 depicts the impact of asynchronous aggregation on clustering accuracy. In the synchronous case, $\|\tau_i - \tau_j\|$ is close to zero, resulting in negligible clustering error. Conversely, in asynchronous settings, $\|\tau_i - \tau_j\|$ may be large due to client heterogeneity, thus impacting clustering accuracy. Also, the mis-clustering rate p is upper-bounded by the decay coefficient α.

Theorem 2. *(Convergence of Training Objective). The training objective $\mathcal{P}$ decreases monotonically, and thus the CFL framework converges under asynchrony, if the following condition is met:*

$$\alpha \leq \frac{\Omega(t)h_i}{|C_k|} \tag{5}$$

where $|C_k|$ is the size of cluster u_k, h_i is the computational capacity of c_i, and $\Omega(t)$ is a time-decreasing function (details in Appendix A.1.2).

Theorem 2 implies that clustering affects the design of model aggregation. Specifically, the decay coefficient α should adapt to the cluster size $|C_k|$. This adjustment is reasonable because larger clusters often exhibit greater staleness [9]. Moreover, Theorem 2 suggests that α should be dynamic and client-specific, given that both $\Omega(t)$ and $|C_k|$ varies over time, and h_i differs across clients.

Summary. We make the following observations.

- The CFL framework remains effective under asynchrony with new requirements on clustering and aggregation.
- Clustering must be adaptable to asynchronous data arrival and manage mis-clustering effectively.
- Aggregation should account for both global information *e.g.*, cluster scale and round, and individual client characteristics when managing stale weights.

Algorithm 1 illustrates our CASA framework, which follows the standard L-C-A workflow. Next, we introduce practical designs to fulfill the new requirements on clustering and aggregation. We start with aggregation since the clustering algorithm is built upon the decay coefficient. All the proofs are in Appendix A.1.

```
Algorithm 1: CASA
Input: Clients' model parameters $w_1, ..., w_n$, the single
       global model parameters $w_{g,1}$
Output: Clients' model parameters $w_1, ..., w_n$, global model
        parameters after clustering $w_{g,1}, ..., w_{g,K}$
1  Server Process:
2  for asynchronous round $t$ do
3      Receive update $w_i^{(t-\tau_i)}$ from client $c_i$
4      if $c_i$ is new client then
5          $k \leftarrow$ argmax $(rep_k, w_i^{(t-\tau_i)})$ as Eq. (15)
6          Assign $c_i$ to cluster $C_k$
7      // C-phase
8      Update similarity matrix $A$ and cluster based on
         Algorithm 3
9      Update buffer space based on Algorithm 2
10     // A-phase
11     Asynchronous aggregation as Eq. (9)
12 Client Process:
13 while True do
14     Receive global model $w_{g,k}^{(t)}$
15     // L-phase
16     Local training $w_i^{(t)} \leftarrow w_{g,k}^{(t)} - \nabla\mathcal{L}(w_{g,k}^{(t)}, D_i)$
17     Upload weight $w_i^{(t)}$ to server asynchronously
```

4.2 Bi-Level Asynchronous Aggregation

Principles. Stale model updates are often decayed by α (see Eq. (3)) during asynchronous aggregation for convergent training of generic FL [5, 19, 40, 41]. Prior research [27, 41] mainly sets α based on *staleness alone*, which is oversimplified for CFL. As highlighted in Sec. 4.1, the decay coefficient should be configured considering various *cluster-* and *client-specific* factors to enhance *clustering* accuracy and *training* convergence. To this end, we suggest a bi-level decay coefficient design that strategically separates and manages the complex dependencies on these factors.

Cluster-Level Decay. Following Theorem 2 but ignoring the client-specific factors, we set the cluster-wise decay for cluster u_k as:

$$\alpha_{c,k}^{(t)} = \frac{\alpha_0 \Omega(t)}{\log(|C_k|)} \tag{6}$$

where α_0 is the initial value of α. The rationales are three-fold.

- We expect the cluster-level decay to penalize the *average* staleness within the cluster, whereas the stragglers would receive extra penalty by the client-level decay. The cluster-level decay also facilitates clustering decisions (see Sec. 4.3).
- From Theorem 2, α should be bounded by a time-decreasing function $\Omega(t)$. This is because α is bounded by the expectation of local gradients (details in Appendix A.1.2). The local gradients are expected to diminish in convergent training, which is characterized by the round-decaying function $\Omega(t)$. Motivated by learning rate scheduling [45], we instantiate $\Omega(t)$ as an *exponential* decay. Uniquely, we show that the decay coefficient α should depend on not only the *relative* staleness, but also the *absolute* rounds in CFL.
- From Theorem 2, α should decrease as the cluster size $|C_k|$ increases. We set α as inversely proportional to $\log(|C_k|)$ because directly applying $|C_k|$ may make α susceptible to dynamic clustering, which affects training stability.

Client-Level Decay. The client-level decay handles the client-specific staleness on top of the cluster-level decay. From Theorem 2, the client-specific staleness is captured by h_i, which is the proportion of local updates completed by a client per round. However, it is difficult for the server to monitor the resources of clients and accurately estimate h_i [31]. Alternatively, we only impose extra decay when the client staleness τ_i exceeds a threshold $r_c^{(t)}$:

$$\alpha_i^{(t)} = \begin{cases} \alpha_{c,k}^{(t)}, & \text{if } \tau_i \leq r_c^{(t)} \\ \alpha_{c,k}^{(t)}/\sqrt{\tau_i}, & \text{if } \tau_i > r_c^{(t)} \end{cases} \tag{7}$$

Such binary classification of staleness is feasible since our proof on convergent training only assumes *bounded* staleness (details in Appendix A.1.2). Unlike previous work [41] that pre-defines the maximal tolerable staleness, we heuristically set $r_{c,k}^{(t)}$ as

$$r_{c,k}^{(t)} = |C_k|(2 - \Omega(t)). \tag{8}$$

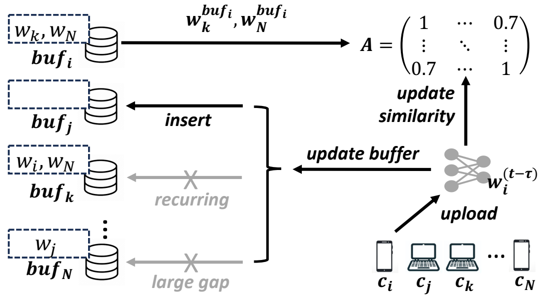


Figure 4: Buffer-aided similarity calculation.

This is because the staleness tends to increase with cluster scale [9], while its impact on training decreases over rounds [38]. Finally, the traditional asynchronous aggregation, *i.e.*, Eq. (3) becomes:

$$w_{g,k}^{(t+1)} = \alpha_i^{(t)} w_i^{(t-\tau_i)} + (1 - \alpha_i^{(t)}) w_{g,k}^{(t)}. \tag{9}$$

4.3 Buffer-Aided Dynamic Clustering

Principles. Clustering in CFL aims to group clients with similar local data distributions for effective training of cluster-wise models. We measure *client similarity* via the cosine distance of their model weights, as common in CFL literature [2, 28, 36, 42], and focus on the *clustering algorithms* suited for the asynchronous environments. As pointed out in Sec. 4.1, asynchrony leads to misaligned model weights for similarity calculation, necessitates incremental clustering with partial information, and enlarges clustering error due to asynchronous aggregation. To address these challenges, we propose a *buffer-aided similarity calculation* scheme, and utilize a *multi-partitioning iterative clustering* approach, where the partitioning criterion is *staleness-dependent*. These designs enable timely and accurate clustering (see case studies in Appendix 5.2.3).

Buffer-Aided Similarity Calculation. From Theorem 1, the clustering error is bounded by the overall *misalignment* between model weights in time, *i.e.*, $\sum_{i=1}^{n} \sum_{j=1}^{n} \|\tau_i - \tau_j\|^2$. An intuitive solution is to *buffer* the model weights for each client and calculate the client similarity using the *most aligned* model versions rather than the *most recent* ones. Specifically, on receiving an update $w_i^{(t)}$ from client c_i, we select clients $c_j, ..., c_k$ whose start time is close to that of c_i, and insert $w_i^{(t)}$ into their buffer space $buf_j, ..., buf_k$. Then we use model parameters inside buffer buf_i for similarity calculation:

$$A_{i,j} = \cos(w_i^{(t)}, w_j^{buf_i}). \tag{10}$$

Finally, we clear the buffer space of c_i, and wait for new updates.

To support the buffer-aided similarity calculation, one needs to store *all* historical client updates, which is unscalable in practice. Accordingly, we optimize the buffer storage below (see Fig. 4).

- *Redundant Update Pruning*. The server directly discards redundant updates without buffering them. Client c_i's model parameters $w_i^{(t-\tau_i)}$ are considered redundant to client c_j in two cases: *(i) large gap*: $t - \tau_i$ is within the range r_c of client c_j's starting round, *i.e.*, $\tau_j - \tau_i \geq r_c$, since large gap leads to high clustering error (see Theorem 1). *(ii) recurring*: buffer buf_j already contains client c_i's model parameters from a previous instance.
- *Prioritized Buffer Allocation*. Given a buffer budget M, we optimize buffer allocation as follows:

$$\max \sum_{i=1}^{n} \sum_{j \in buf_i} \frac{\mathbf{1}_{\{cal_{i,j}=1\}}}{\alpha_i^{(t)}}, \text{ s.t.} \sum_{i=1}^{n} |buf_i| \leq M \tag{11}$$

where $\mathbf{1}_{\{cal_{i,j}=1\}}$ indicates whether the similarity between c_i and c_j has already been computed, and $\alpha_i^{(t)}$ is the client-level decay defined in Eq. (7). This is because *(i)* we need all the pairwise similarities to make clustering decisions, and *(ii)* slower clients are more outdated and have less chances to update the similarity matrix, thus of higher priority for updating. Eq. (11) is an online knapsack problem and can be solved in two steps: *(i)* Sort clients' buffers following the order of two keywords, *i.e.*, # of similarities not computed, and $\alpha_i^{(t)}$. *(ii)* Greedily select the first M buffers. Algorithm 2 illustrates our buffer allocation scheme.

Algorithm 2: Prioritized buffer allocation.

```
Input: Received client update $w_i^{(t-\tau_i)}$
Output: New buffer for clients $buf_1, ..., buf_n$
1 for client $c_j \in C_k$ do
2     if $w_i^{(t-\tau_i)}$ not redundant then
3         Add $w_i^{(t-\tau_i)}$ to $buf_j$
4 if $\sum_{i=1}^{n} |buf_i| > M$ then
5     Sort all buffers based on
6     uncomputed or not as the primary key,
7     decay function $\alpha_i$ as the secondary key,
8     Delete last $\sum_{i=1}^{n} |buf_i| - M$ items
```

Multi-Partitioning Iterative Clustering. Most CFL methods assume a fixed number of clusters [13, 22], which is unfit for dynamics in the asynchronous environments. One solution is to incrementally adjust the number of clusters via iterative bi-partitioning [28, 42]. Yet the bi-partitioning can be slow to converge. To boost the clustering efficiency, we apply a multi-partitioning-based approach.

Specifically, we consider clients in a cluster as a graph $G(V, E)$, with clients as vertex set V and their pairwise similarities as edges set E. Our goal is to decide whether there is an appropriate gap in the current graph G for partitioning into R subgraphs $\{G_1, G_2, ..., G_R\}$. To estimate the number of subgraphs, we calculate the maximum eigengap of G with the Laplacian matrix of its similarity matrix A:

$$L = I - D^{-1/2} A D^{-1/2} \tag{12}$$

Let $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ be the eigenvalue of L. We select the maximum eigengap to retrieve the best R partitions [37].

$$R = \arg\max_{k} \lambda_{k+1} - \lambda_k, \ \ s.t.\{\lambda_1, \lambda_2, ..., \lambda_n\} \leftarrow SVD(L) \tag{13}$$

From Theorem 1, we should only perform clustering when the (cluster-level) decay $\alpha_{c,k}$ is not too large. Accordingly, we partition clients into R clusters only when:

$$\alpha_{c,k} < (\lambda_{R+1} - \lambda_R)^{\gamma} \tag{14}$$

Algorithm 3: Multi-partitioning iterative clustering.

```
Input: Similarity matrix $A$, cluster-level aggregation
       parameter $\alpha_{c,k}^{(t)}$, received client update $w_i^{(t-\tau_i)}$
Output: New cluster list $u_1, ..., u_K$
1 for client $c_j \in buf_i$ do
2     Update similarity matrix $A_{i,j} \leftarrow \cos(w_i^{(t-\tau_i)}, w_j^{buf_i})$
3 Laplacian matrix $L \leftarrow I - D^{-1/2}AD^{-1/2}$
4 $\lambda_1, \lambda_2, ...\lambda_n \leftarrow SVD(L)$
5 $R \leftarrow \arg\max_k \ \lambda_{k+1} - \lambda_k$
6 if $\alpha_{c,k}^{(t)} < (\lambda_{R+1} - \lambda_R)^\gamma$ then
7     $u_{k1}, u_{k2}, ..., u_{kR} \leftarrow SpectralClustering(u_k, R)$
8     Replace $u_k$ with $u_{k1}, u_{k2}, ..., u_{kR}$
```

where $\lambda_{R+1} - \lambda_R$ is the maximum eigengap of similarity matrix A, and γ is a hyperparameter to scale the eigengap comparable to $\alpha_{c,k}$. Algorithm 3 shows the overall clustering workflow.

Newcomer Assignment. Correctly clustering new clients in the context of asynchronous clustered federated learning presents a challenging problem. The reasons include:

- *Weight gap*: The inconsistent update frequency among clients means that, rather than being assigned to a cluster more similar in terms of data, a new client might be more easily assigned to a cluster that has been updated less frequently.
- *Center shift*: In asynchronous cases, the global model does not effectively represent the information of all clients within a cluster, as it is dominated by fast clients.

To address these issues, we leverage historical information to assist the clustering of newcomers. For each cluster u_k, we filter out the historical information of clients in the newcomer's sampled buffer buf_i. We aggregate historical information in $buf_i \cap u_k$ as Eq. (15) to obtain a representative information of the cluster, and then select an appropriate cluster for the newcomer based on this representation.

$$rep_k = \sum_{c_j \in buf_i \cap u_k} \frac{|D_j|}{|D_{buf_i \cap u_k}|} w_j^{buf_i} \tag{15}$$

Discussions. We make the following notes on our buffer-aided dynamic clustering scheme.

- Buffers have been utilized for stable training of a single model (without clustering) in asynchronous FL [25, 31, 40]. Our work is orthogonal as we apply buffers in clustering. Our clustering scheme could be integrated with buffer-based model aggregation for better training.
- Although our clustering algorithm is built upon prior multi-partitioning methods, the key novelty is to trigger clustering via Eq. (14). It allows high clustering accuracy guarantee with Theorem 1, and is noise-resilient since the maximum eigengap is typically large [37].

4.4 Mitigating Staleness via Sparse Training

Principles. We further improve CASA by *actively* mitigating staleness, *i.e.*, allocating lower training workload to slower clients. The basic idea is to incorporate *sparse training* [1, 48], where personalized masks are assigned to clients to reduce their local training workload, thus enhancing the stability of the model convergence.

Designs. We implement the idea as CASA+, which has two designs:

Firstly, during asynchronous aggregation, we only mix the masked weights into the global model:

$$w_{g,k}^{(t+1)} \odot m_i^{(t-\tau_i)} = ((1-\alpha_i^{(t)})w_{g,k}^{(t)} + \alpha_i^{(t)} w_i^{(t-\tau_i)}) \odot m_i^{(t-\tau_i)} \tag{16}$$

where $m_i^{(t-\tau_i)}$ is the mask for client c_i. $m_i^{(t-\tau_i)}$ is obtained by masking the smallest $\mathbb{S}_i^{(t)}$-proportion of local parameters $w_i^{(t)}$. Eq. (16) limits the influence of stale updates on global model with aggregating partial parameters, which also boosts convergence.

Secondly, we configure the sparsity rate $\mathbb{S}_i^{(t)}$ as:

$$\mathbb{S}_i^{(t)} = \frac{\mathbb{S}_0}{2}(\alpha_{c,k}^{(t)} - \alpha_i^{(t)})(1 + \cos((\frac{(t-1)\pi}{R_{end}}))) \tag{17}$$

which applies a cosine annealing function [21] to the divergence of cluster- and client-level decay coefficient, where R_end is predefined total rounds. This ensures that fast clients are unlikely to be masked, because local $\alpha_i^{(t)}$ equals to cluster-level decay. The reason to control the sparsity rate via the decay coefficient is as follows. *(i)* Clients with higher staleness will receive a higher degree of sparsity. *(ii)* Sparsification impairs similarity calculation, and sparsity rate before clustering should be less than that after clustering.

5 Experiments

5.1 Experimental Setup

Compared Methods. We compare CASA with representative CFL and AFL methods. We also extend the CFL baselines to the asynchronous setting. Specifically, we apply decay function in model aggregation of these CFL baselines, and buffer the recent model parameters from each client for similarity calculation. The server clusters or check clustering condition per asynchronous round.

- **Standalone**: Each client trains its model with its local dataset only without federated training.
- **FedAvg** [24]: generic FL that trains a global model via weighted averaging of local model parameters.
- **FedProx** [17]: generic FL that applies a proximal to boost the convergence.
- **FedAsync** [41]: classic AFL that aggregates model with staleness-based weight decay.
- **FedBuff** [25]: AFL that applies buffer to aggregate the most recent K gradients.
- **CFL** [28]: synchronous CFL that adopts bi-partitioning clustering based on mean and maximum norm of gradients.
- **CFL-Async**: asynchronous extension of **CFL** [28].
- **ICFL** [42]: synchronous CFL that adopts bi-partitioning and incremental clustering.
- **ICFL-Async**: asynchronous extension of **ICFL** [42].
- **IFCA** [13]: synchronous CFL that iteratively clusters clients based on minimizing loss.
- **IFCA-Async**: asynchronous extension of **IFCA** [13].

Table 1: Overall performance. *Acc* is the overall accuracy at convergence, *Time* is the time to reach the target accuracy. "/" means that the method fails to reach the target accuracy. For IFCA, the item in parentheses means its pre-set cluster number k.

Type	Method	MNIST		CIFAR-10		FEMNIST		IMU		HARBox	
		Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
N/A	Standalone	98.26	4.9	82.6	35.2	93.95	/	90.00	41.9	69.48	/
	FedAvg	97.92	80.24	69.49	/	98.34	85.01	85.71	89.08	82.90	251.42
	FedProx	97.31	104.68	65.83	/	98.14	114.34	87.57	96.05	81.80	390.96
	CFL	99.42	30.63	90.50	209.55	98.75	75.14	86.29	94.93	85.06	170.99
Sync		99.40(3)	11.21	89.10(3)	209.62	97.77(2)	145.34	94.28(2)	80.73	83.73(2)	334.34
	IFCA(k)	99.50(4)	12.8	90.88(4)	77.88	96.59(5)	237.46	92.67(3)	73.28	85.06(4)	226.53
		99.48(5)	5.61	91.14(5)	80.39	95.37(8)	279.85	91.81(4)	148.6	87.09(6)	220.98
	ICFL	98.68	12.18	84.19	36.49	95.35	121.65	93.23	30.45	82.58	126.43
	FedAsync	97.46	109.53	67.66	/	96.97	183.57	86.86	64.9	78.72	158.97
	FedBuff	99.25	53.31	61.11	/	98.21	82.63	80.00	282.9	81.62	55.7
	CFL-Async	99.23	9.54	89.97	145.8	98.68	36.67	87.71	69.3	82.43	59.8
Async		99.28(3)	9.53	83.41(3)	195.30	98.39(2)	81.07	89.61(2)	143.1	77.58(2)	252.60
	IFCA-Async(k)	98.88(4)	8.83	88.99(4)	80.70	97.78(5)	118.27	85.62(3)	143.1	78.13(4)	215.20
		99.32(5)	8.57	87.98(5)	83.20	97.62(8)	126.77	89.14(4)	87.77	76.81(6)	236.30
	ICFL-Async	98.82	5	83.30	25.3	94.66	/	91.52	81.83	79.65	92.00
Ours	CASA	**99.52**	**2.80**	**91.45**	23.4	**98.97**	36.2	**95.33**	37.47	**87.38**	54.8
	CASA+	99.34	4.80	90.64	**20.3**	98.53	**35.03**	94.57	**22.71**	87.21	**53.6**

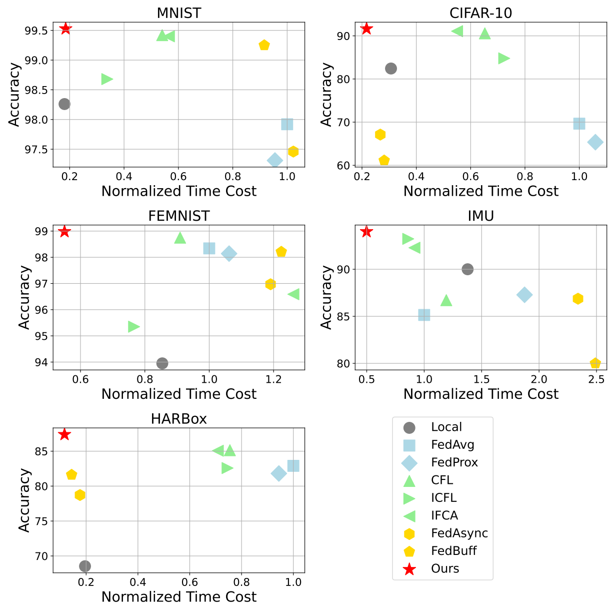


Figure 5: Model accuracy vs. normalized time cost.

Datasets. We test two tasks. *(i)* image classification (IC): MNIST [16], CIFAR10 [15], and FEMNIST [3]; *(ii)* human activity recognition (HAR): IMU [26], and HARBox [26].

Metrics. We assess the methods with three metrics: *(i) Accuracy*: the accuracy on local datasets when converged; *(ii) Time to Target Accuracy*: the latency to reach a given accuracy *i.e.*, 95% for MNIST, 80% for CIFAR-10, 95% for FEMNIST, 80% for IMU, 75% for HARBox; *(iii) Time to Convergence*: the time required to convergence.

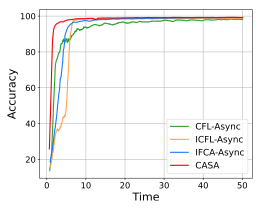
Other details on experimental setups are in Appendix A.2.

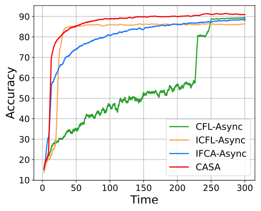
5.2 Main Results

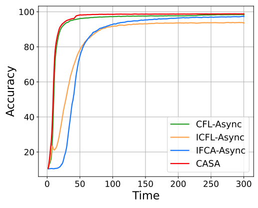
5.2.1 Time-to-Accuracy. Table. 1 summarizes the time to reach a target accuracy and the accuracy at convergence. Fig. 5 highlights the trade-off between model accuracy and normalized latency (w.r.t. FedAvg) at convergence. We make the following observations.

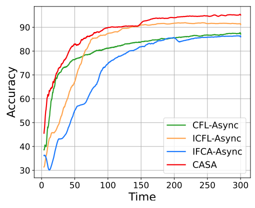
- *Gains over Synchronous FL & CFL.* Synchronous CFL outperforms the generic counterpart in both accuracy and latency in most cases, particularly with *label-skew*, *e.g.*, on MNIST and CIFAR-10. We achieve higher accuracy than generic FL, *e.g.* FedAvg (0.63% to 21.96%), and FedProx (0.83% to 30.34%), and comparable accuracy to the CFL baselines, even though we rely on partial information for clustering. However, our methods are significantly faster, reducing the time to reach the target accuracy by 2.00-37.39× on MNIST, 1.79-10.33× on CIFAR-10, 2.15-7.99× on FEMNIST, 1.34-4.23× on IMU, and 2.36-7.29× on HARBox. From Fig. 5, our CASA reduces up to 3.12×, 3.40×, 2.28×, 2.41×, 6.49× convergence time while maintaining a comparable and even higher accuracy than the synchronous CFL baselines.
- *Gains over Asynchronous FL & CFL.* AFL methods (without clustering) suffer from low accuracy and long latency on highly non-IID data. Concretely, the accuracy of our methods is 2.00-23.79% higher than FedAsync, and 0.27-30.34% higher than FedBuff. They also take more time to reach the target accuracy, *e.g.* up to 39.11× on MNIST, 5.25× on FEMNIST, 12, 76× on IMU, and 2.97× on HARBox, and they fail to reach the target accuracy on CIFAR-10. Our methods are also faster than the asynchronous CFL baselines, and improves the accuracy by up to 2.68%, 27.53%, 2.56%, 7.72%, 6.83% at convergence on the five datasets.

5.2.2 Training Curves. This experiment tracks the training dynamics of asynchronous CFL methods. From Fig. 6, CASA converges faster and achieves a higher test accuracy than other methods. IFCA-Async managed to capture the correct clustering structures on label-skewed datasets like MNIST and CIFAR-10, but fails on


(a) MNIST


(b) CIFAR-10


(c) FEMNIST


(d) IMU

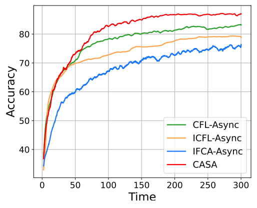
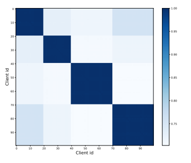
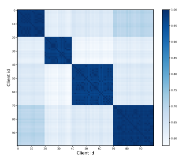
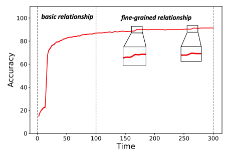

(e) HARBox

Figure 6: Training time and test accuracy of asynchronous CFL methods.


(a) Sims at $t = 100$


(b) Sims at $t = 300$



(c) Accuracy at two stages

Figure 7: Analysis of effectiveness of clustering.

feature-skewed datasets like FEMNIST, IMU and HARBox. This is because IFCA-Async determines cluster identity based on training loss, yet the difference in loss is small in feature-skewed datasets. Furthermore, the global model in IFCA-Async can be biased due to asynchrony (see Fig. 10b). CFL-Async performs well in part of datasets. This is because α affects clustering in the asynchronous scenarios, and such impact differs across datasets (see Fig. 10a).

5.2.3 *Effectiveness of Clustering.* This experiment assesses the effectiveness of CASA's clustering. We test on CIFAR-10 and observe the similarity matrix of clients' model parameters clients at different times. The data partitioning strategy is the same as in Appendix A.2.4, with 4 clusters, each holding a few unique labels.

As shown in Fig. 7a, at $t = 100$, a fairly clear clustering structure has emerged, but the inner relationship within a cluster has not been discovered yet. As shown in Fig. 7b, by $t = 300$, we further capture the data distributions within each cluster, resulting in finer-grained and more accurate clusters.

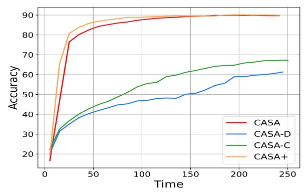
Fig. 7c further illustrates the accuracy during training. At $t = 100$ when there is clear but not fine-grained clustering relationship, the overall accuracy reaches 86.5%. With training and mining of clients' similarity, the large clusters are partitioned into sub-clusters with higher cluster similarities. The partitioning boosts the accuracy, with a final accuracy of 91.6%.

5.2.4 *Impact of System Heterogeneity.* This experiment tests another two system heterogeneity setups on HARBox. *Scenario A*: It is a general case where the maximum speed gap is 5, and the clients' local training latencies are randomly sampled from this range [19]. *Scenario B*: It is the case with severe system heterogeneity, with 30% clients taking 10× the training time of fast clients.

Table. 2 summarizes the convergence accuracy and training latency. CASA still outperforms the baselines. Compared with the general case, CASA achieves a comparable accuracy with merely 1.08× higher latency, when facing severe system heterogeneity.

Table 2: Impact of system heterogeneity on HARBox.

Method	Scenario A (General)		Scenario B (Severe)	
	Acc	Time	Acc	Time
Standalone	70.47	498.9	70.43	531.2
FedAvg	83.18	1115.93	81.15	2015.19
FedProx	81.32	1190.16	80.19	2056.94
CFL	83.27	977.43	84.17	1614.84
IFCA	84.45	707.24	85.85	1687.42
ICFL	82.09	1019.7	82.12	2095.65
FedAsync	80.23	419.1	79.32	489.3
FedBuff	82.14	400.1	80.32	528.8
Ours	**87.61**	**346.6**	**87.81**	**374.6**


(a) CIFAR-10

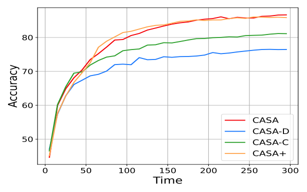

(b) HARBox

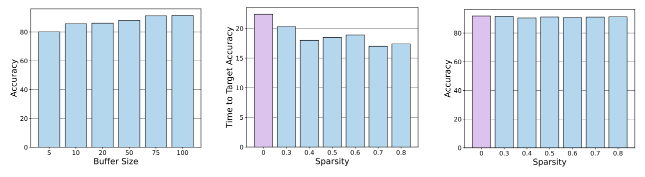
Figure 8: Contributions of individual components.

Conversely, other baselines (except standalone, which only trains locally) are 1.17× to 2.39× shower in Scenario B. Thus the results show that CASA is robust to severe system heterogeneity.

5.3 Ablation Study

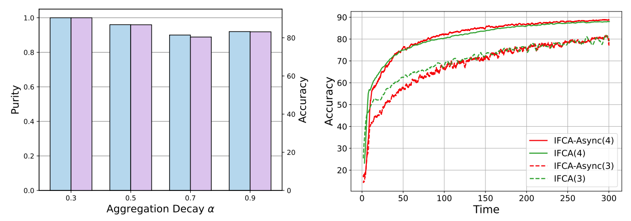
5.3.1 *Contributions of Individual Components.* In addition to CASA and CASA+, we test the following variants to understand the effective of different designs: *(i)* CASA-C, which is CASA without clustering; and *(ii)* CASA-D, which is CASA without bi-level α.

As shown in Fig. 8, both CASA and CASA+ outperform CASA-C and CASA-D. Due to lack of clustering, CASA-C fails to generate personalized models adapted to local data distributions, leading to lower model accuracy. Without the dynamical decay coefficient, CASA-D not only induces erroneous clustering but also suffers from slow convergence, hence exhibiting the poorest performance. Both CASA and CASA+ outperforms CASA-D in accuracy, *e.g.* up to 34.23% in CIFAR-10, up to 12.37% in HARBox. CASA-C performs better than CASA-D, but still worse than CASA and CASA+, with 23.62% and 7.74% lower accuracy, respectively.



(a) (b) (c)

Figure 9: Hyperparameter tests: impact of (a) buffer size on accuracy; impact of sparsity rate on (b) latency to target accuracy and (c) accuracy at convergence.



(a) Hierarchical clustering **(b) Dynamic clustering**

Figure 10: Impact of asynchrony on clustering.

5.3.2 *Impact of Buffer Size.*

This experiment shows the effectiveness of prioritized buffer allocation scheme (Sec. 4.3) by recording accuracy with 100 clients on CIFAR-10 given different buffer sizes.

From Fig. 9a, the accuracy rises to 91.21% with a buffer size of 75. Recall from Table. 1, the highest accuracy of asynchronous CFL baselines is 89.97%. That is, CASA surpasses other asynchronous CFL methods even when the buffer size is 75. Note that methods such as CFL-Async and ICFL-Async need a buffer at least equal to the client scale *i.e.*, $n = 100$ to store the most recent model parameters of each client and calculate the pair-wise client similarity. Therefore, our prioritized buffer allocation mechanism yields higher accuracy at lower storage overhead.

5.3.3 *Impact of Sparsity Rate.*

This experiment shows the impact of sparsity rate on CASA+ (Sec. 4.4), which mitigates staleness with sparse training. We measure the latency to the target accuracy and the accuracy at convergence at various sparsity rate. Note that $\mathbb{S}_0 = 0$ means no sparse training, *i.e.*, CASA.

From Fig. 9b, introducing sparse training with a initial sparse rate $\mathbb{S}_0 = 0.3$ results in 1.09× shorter latency to target accuracy compared with CASA (the first bar). As we increase the sparsity rate $\mathbb{S}_0$, the reduction in latency of CASA+ is 1.19-1.32×. However, when the sparsity rate increases, the time required to reach the target accuracy does not always decrease. This is because an over-sparse model may not effectively contribute to the global model. With faster convergence, the accuracy of CASA+ only drops 0.29-1.28% as shown in Fig. 9c, which is acceptable.

5.3.4 *Impact of Asynchronous Aggregation on CFL.*

This experiment tests the impact of asynchronous aggregation on CFL schemes to highlight the challenges when shifting to asynchronous CFL. We select CFL [28] and IFCA [13], two representative synchronous CFL methods that adopt a *hierarchical* and a *dynamic* clustering strategy, respectively.

- *Impact on Hierarchical Clustering*. From Theorem 1, a larger α leads to a higher mis-clustering probability, particularly for naive methods such as hierarchical clustering. We validate this by testing CFL-Async on CIFAR-10 under various α values. We measure the clustering performance its purity [42] which stands for the accuracy of clustering and the training performance by the test accuracy. Fig. 10a shows the results. As α increases, the clustering error occurs. The mis-clustered clients will then introduce unwanted data heterogeneity into the cluster, which impairs training accuracy. For example, when $\alpha = 0.7$, CFL-Async suffers high mis-clustering rate, with an accuracy drop of 10.09% on CIFAR-10, respectively. Note that CASA avoid these problems because *(i)* clustering is only activated when α is small, which explicitly controls clustering errors, *(ii)* with buffer limiting $\tau_i - \tau_j$, according to Theorem 1, CASA is more robust with respect to α.
- *Impact on Dynamic Clustering*. Asynchrony makes dynamic clustering *e.g.*, IFCA sensitive to client arrival orders. This is because the global model is biased towards the most recent local updates due to weight decay. This would exert challenges to assign clients to the correct clusters. To validate this, we compare IFCA and IFCA-Async on CIFAR-10 with cluster number set to $k = 3$ and $k = 4$. As shown in Fig. 10b, IFCA-Async experiences more unstable convergence than IFCA. IFCA-Async reaches the target accuracy only 1.09× faster than IFCA when $k = 4$, and IFCA is even faster than IFCA-Async when $k = 3$. Therefore, the shift to asynchronous CFL does not necessarily improve the convergence speed, which motivates our designs.

6 Conclusion

This paper presents CASA, a new CFL scheme for asynchronous clients to boost training efficiency under both data and system heterogeneity. We systematically analyze the impact of asynchrony on CFL. We further propose a bi-level asynchronous aggregation method and a buffer-aided dynamic clustering strategy to pertain the synergy between client clustering and model aggregation in asynchronous environments. Extensive evaluations show that CASA outperforms existing CFL and AFL baselines in accuracy and achieves 2.28-6.49× faster convergence speed. We envision CASA as a practical solution for swift and personalized federated learning with heterogeneous IoT devices.

Acknowledgments

We are grateful to anonymous reviewers for their constructive comments. This work is partially supported by National Science Foundation of China (NSFC) under Grant No. U21A20516 and 62336003, Beijing Natural Science Foundation No. Z230001, Beihang University Basic Research Funding No. YWF-22-L-531, Didi Collaborative Research Program NO2231122-00047, and CityU APRC grant No. 9610633. Yongxin Tong is the corresponding author.

References

[1] Sameer Bibikar, Haris Vikalo, Zhangyang Wang, and Xiaohan Chen. 2022. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of AAAI*, Vol. 36. 6080–6088.

[2] Christopher Briggs, Zhong Fan, and Peter Andras. 2020. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In *Proceedings of IJCNN*. 1–9.

[3] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[4] Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. 2022. pFL-bench: A comprehensive benchmark for personalized federated learning. *Advances in Neural Information Processing Systems* 35 (2022), 9344–9360.

[5] Ming Chen, Bingcheng Mao, and Tianyi Ma. 2019. Efficient and robust asynchronous federated learning with stragglers. In *Proceedings of ICLR*.

[6] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *Proceedings of Big Data*. 15–24.

[7] Yang Chen, Xiaoyan Sun, and Yaochu Jin. 2019. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Transactions on Neural Networks and Learning Systems* 31, 10 (2019), 4229–4238.

[8] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting shared representations for personalized federated learning. In *Proceedings of ICML*. 2089–2099.

[9] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. 2012. Large scale distributed deep networks. *Advances in Neural Information Processing Systems* 25 (2012).

[10] Yiqun Diao, Qinbin Li, and Bingsheng He. 2023. Exploiting Label Skews in Federated Learning with Model Concatenation. *arXiv preprint arXiv:2312.06290* (2023).

[11] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems* 33 (2020), 3557–3568.

[12] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. 2023. A general theory for federated optimization with asynchronous and heterogeneous clients updates. *Journal of Machine Learning Research* 24, 110 (2023), 1–43.

[13] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19586–19597.

[14] Ling Huang, Donghui Yan, Nina Taft, and Michael Jordan. 2008. Spectral clustering with perturbed data. *Advances in Neural Information Processing Systems* 21 (2008).

[15] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[17] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of MLSys* 2 (2020), 429–450.

[18] Youpeng Li, Xuyu Wang, and Lingling An. 2023. Hierarchical Clustering-based Personalized Federated Learning for Robust and Fair Human Activity Recognition. *Proceedings of IMWUT* 7, 1 (2023), 1–38.

[19] Ji Liu, Juncheng Jia, Tianshi Che, Chao Huo, Jiaxiang Ren, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2023. FedASMU: Efficient Asynchronous Federated Learning with Dynamic Staleness-aware Model Update. *arXiv preprint arXiv:2312.05770* (2023).

[20] Jiachen Liu, Fan Lai, Yinwei Dai, Aditya Akella, Harsha V Madhyastha, and Mosharaf Chowdhury. 2023. Auxo: Efficient Federated Learning via Scalable Client Clustering. In *Proceedings of SoCC*. 125–141.

[21] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2021. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *Proceedings of ICML*. 6989–7000.

[22] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web* 26, 1 (2023), 481–500.

[23] Jie Ma, Guodong Long, Tianyi Zhou, Jing Jiang, and Chengqi Zhang. 2022. On the convergence of clustered federated learning. *arXiv preprint arXiv:2202.06187* (2022).

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*. 1273–1282.

[25] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. 2022. Federated learning with buffered asynchronous aggregation. In *Proceedings of AISTATS*. 3581–3607.

[26] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of MobiSys*. 54–66.

[27] Jungwuk Park, Dong-Jun Han, Minseok Choi, and Jaekyun Moon. 2021. Sageflow: Robust federated learning against both stragglers and adversaries. *Advances in Neural Information Processing Systems* 34 (2021), 840–851.

[28] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2020), 3710–3722.

[29] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of SenSys*. 127–140.

[30] Ningxin Su and Baochun Li. 2022. How Asynchronous can Federated Learning Be?. In *Proceedings of IWQoS*. 1–11.

[31] Jingwei Sun, Ang Li, Lin Duan, Samiul Alam, Xuliang Deng, Xin Guo, Haiming Wang, Maria Gorlatova, Mi Zhang, Hai Li, et al. 2022. FedSEA: A Semi-Asynchronous Federated Learning Framework for Extremely Heterogeneous Devices. In *Proceedings of SenSys*. 106–119.

[32] Ahmet Ali Süzen, Burhan Duman, and Betül Şen. 2020. Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn. In *Proceedings of HORA*. 1–5.

[33] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems* 33 (2020), 21394–21405.

[34] Afaf Taik, Zoubeir Mlika, and Soumaya Cherkaoui. 2022. Clustered vehicular federated learning: Process and optimization. *IEEE Transactions on Intelligent Transportation Systems* 23, 12 (2022), 25371–25383.

[35] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[36] Saeed Vahidian, Mahdi Morafah, Weijia Wang, Vyacheslav Kungurtsev, Chen Chen, Mubarak Shah, and Bill Lin. 2023. Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces. In *Proceedings of AAAI*, Vol. 37. 10043–10052.

[37] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[38] Haoming Wang and Wei Gao. 2023. Tackling the Unlimited Staleness in Federated Learning with Intertwined Data and Device Heterogeneities. *arXiv preprint arXiv:2309.13536* (2023).

[39] Yansheng Wang, Yongxin Tong, Zimu Zhou, Ruisheng Zhang, Sinno Jialin Pan, Lixin Fan, and Qiang Yang. 2023. Distribution-Regularized Federated Learning on Non-IID Data. In *Proceedings of ICDE*. 2113–2125.

[40] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. 2020. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Trans. Comput.* 70, 5 (2020), 655–668.

[41] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).

[42] Yihan Yan, Xiaojun Tong, and Shen Wang. 2023. Clustered Federated Learning in Heterogeneous Environment. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[43] Zhikai Yang, Yaping Liu, Shuo Zhang, and Keshen Zhou. 2023. Personalized federated learning with model interpolation among client clusters and its application in smart home. *World Wide Web* (2023), 1–26.

[44] Mang Ye, Xiuwen Fang, Bo Du, Pong C Yuen, and Dacheng Tao. 2023. Heterogeneous Federated Learning: State-of-the-art and Research Challenges. *Comput. Surveys* (2023).

[45] Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. 2019. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878* (2019).

[46] Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. 2021. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems* 34 (2021), 10092–10104.

[47] Tuo Zhang, Lei Gao, Sunwoo Lee, Mi Zhang, and Salman Avestimehr. 2023. TimelyFL: Heterogeneity-aware Asynchronous Federated Learning with Adaptive Partial Training. In *Proceedings of CVPR*. 5063–5072.

[48] Wenhao Zhang, Zimu Zhou, Yansheng Wang, and Yongxin Tong. 2023. Dm-pfl: Hitchhiking generic federated learning for efficient shift-robust personalization. In *Proceedings of SIGKDD*. 3396–3408.

[49] Yu Zhang, Duo Liu, Moming Duan, Li Li, Xianzhang Chen, Ao Ren, Yujuan Tan, and Chengliang Wang. 2023. FedMDS: An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework. *IEEE Transactions on Parallel and Distributed Systems* 34 (2023), 1007–1019.

A appendix

A.1 Proofs

A.1.1 Proof of Theorem 1. We make the following assumptions as in [13, 19, 23, 39, 41].

Assumption 1. *(Unbiased gradient estimator and Bounded gradients). The expectation of stochastic gradient $\nabla l(w_i, \xi_i)$ is an unbiased estimator of the local gradient for each client, and expectation of L2 norm of $\nabla l(w_i, \xi_i)$ is bounded by a constant U:*

$$\mathbb{E}_{\xi_i \sim D_i}[\|\nabla l(w_i, \xi)\|_2] \leq U \tag{18}$$

Then we prove Theorem 1, which provides an upper-bound for the mis-clustering rate p.

Proof. In the synchronous setting, the similarity between two clients is calculated as $A_{ij} = \cos(w_i^{(t)}, w_j^{(t)})$ in round t. In asynchronous setting, for client c_i starting at round $t - \tau_i$ and client c_j starting at round $t - \tau_j$, the similarity calculation becomes $A'_{ij} = \cos(w_i^{(t-\tau_i)}, w_j^{(t-\tau_j)})$. The divergence between A_{ij} and A'_{ij} is:

$$A'_{ij} - A_{ij} = \leq \cos(w_j^{(t-\tau_i)}, w_j^{(t-\tau_j)}) = O(\|w_g^{(t-\tau_i)} - w_g^{(t-\tau_j)}\|) \tag{19}$$

From [41], the gap of global model parameters is bounded as

$$\|w_g^{(t)} - w_g^{(t-\tau)}\| \leq \alpha\gamma K H_{max} O(V_2) \tag{20}$$

where K is upper bound of staleness, γ is learning rate, H_{max} is maximum of local steps, V_2 is upper bound of gradients' norm.

Take Eq. (20) into our problem, we have:

$$A'_{ij} - A_{ij} = O((\tau_i - \tau_j)\alpha\eta Q\theta U) = O(\lambda\alpha(\tau_i - \tau_j)) \tag{21}$$

where Q refers to the maximum of local steps, η is the learning rate, and θ is the upper bound of staleness. For simplicity, we use λ to represent $\lambda = \eta Q\theta U$.

The Frobenius-norm of $A' - A$ can be bounded by Eq. (21):

$$\|A' - A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\|A'_{ij} - A_{ij}\|^2} = O(\lambda\alpha\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\|\tau_i - \tau_j\|^2}) \tag{22}$$

Given degree matrix D' as $D'_{ii} = \sum_{j=1}^{n}\cos(w_i^{(t-\tau_i)}, w_j^{(t-\tau_j)})$, the divergence of Forbenius-norm is

$$\|D' - D\|_F = \sqrt{\sum_{i=1}^{n}(D'_{ii} - D_{ii})^2} = O(\lambda\alpha\sqrt{\sum_{i=1}^{n}(\sum_{j=1}^{n}\|\tau_i - \tau_j\|^2)}) \tag{23}$$

The Frobenius-norm of matrix $L' - L$ can be rewritten as $\|A' - A - (D' - D)\|_F$. For item that meets $i \neq j$, there is $A'_{ij} - A_{ij} = L'_{ij} - L_{ij}$. The only difference lies in the elements on the diagonal. For an arbitrary i, as $A'_{ii} - A_{ii} = 1 - 1 = 0$, there exists:

$$\|L' - L\|_F = \sqrt{\|A' - A\|_F^2 + \|D' - D\|_F^2} = O(\lambda\alpha\sqrt{\sum_{i=1}^{n}(\sum_{j=1}^{n}\|\tau_i - \tau_j\|^2)}) \tag{24}$$

As proved in [14], the mis-clustering rate p of the spectral partitioning algorithm satisfies

$$p \leq \delta^2 = \|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\|^2 \leq \frac{4\|\mathbf{L}' - \mathbf{L}\|_\mathbf{F}}{\mathbf{v} - \sqrt{2}\|\mathbf{L}' - \mathbf{L}\|_\mathbf{F}} \tag{25}$$

where $\mathbf{v}_2$ is the unit-length second eignvectors of matrix L.

Theorem 1 is proved by telescoping all equations above. □

A.1.2 Proof of Theorem 2. For simplicity, we use λ_i to represent client c_i's weight $\frac{|D_i|}{|D|}$, and $\mathcal{P}^{t,C}, \mathcal{P}^{t,A}, \mathcal{P}^{t,L}$ denote the training objective $\mathcal{P}$ of C-, A-, L-phase in round t, respectively. To prove the convergence of CASA, we firstly make assumptions below.

Assumption 2. *(Convex). Each loss function l or $\mathcal{L}$ is convex.*

$$l(y) \geq l(x) + \langle \nabla l(x), y - x \rangle \tag{26}$$

Assumption 3. *(Lipschitz Smooth). Each loss function l or $\mathcal{L}$ is β-smooth.*

$$l(y) \leq l(x) + \langle \nabla l(x), y - x \rangle + \frac{\beta}{2}\|y - x\|_2^2 \tag{27}$$

Assumption 4. *(Bounded gradient variance). The variance of stochastic gradient $\nabla l(w_i, \xi_i)$ is bounded by σ^2,*

$$\mathbb{E}_{\xi_i \sim D_i}[\|\nabla l(w_i, \xi_i) - \nabla l(w_i)\|_2^2] = \mathbb{E}[\|\nabla l(w_i, \xi_i)\|_2^2] - \|\nabla l(w_i)\|_2^2 \leq \sigma^2 \tag{28}$$

Assumption 5. *(Bounded staleness). At asynchronous training round t, the server received an update $w_i^{(t-\tau_i)}$ from client c_i. The update staleness is bounded, meeting $\tau_i \leq \theta$.*

Lemma 1. *From C-phase to A-phase in an arbitrary round t, we have*

$$\mathcal{P}^{t,A} \leq \mathcal{P}^{t,C} + \sum_{k=1}^{K}\sum_{c_i \in u_k}\lambda_i\eta_i^{(t-\tau_i)}QU(\alpha_i^{(t-\tau_i)}\theta O(U) + U) \tag{29}$$

Proof. The distance between cluster u_k's global model and received parameters from client c_i should be:

$$\begin{aligned}\|w_{g,k}^{(t)} - w_i^{(t-\tau_i)}\|_2 &\leq \|w_{g,k}^{(t)} - w_{g,k}^{(t-\tau_i)}\|_2 + \|w_{g,k}^{(t-\tau_i)} - w_i^{(t-\tau_i)}\|_2 \\ &\leq \alpha_i^{(t-\tau_i)}\eta_i^{(t-\tau_i)}\theta Q O(U) + \eta_i^{(t-\tau_i)}QU\end{aligned} \tag{30}$$

For an arbitrary cluster u_k, we have:

$$\begin{aligned}&\sum_{c_i \in u_k}\mathbb{I}(q_i^{end} = Q)\lambda_i(\mathcal{L}(w_{g,k}^{(t+1)}, D_i) - \mathcal{L}(w_i^{(t-\tau_i)}, D_i)) \\ \leq &\sum_{c_i \in u_k}\mathbb{I}(q_i^{end} = Q)\lambda_i(\langle \nabla\mathcal{L}(w_{g,k}^{(t+1)}, D_i), w_{g,k}^{(t+1)} - w_i^{(t-\tau_i)}\rangle) \\ \leq &\sum_{c_i \in u_k}\lambda_i\eta_i^{(t-\tau_i)}QU(\alpha_i^{(t-\tau_i)}\theta O(U) + U)\end{aligned} \tag{31}$$

Lemma 1 is satisfied with

$$\mathcal{P}^{t,A} - \mathcal{P}^{t,C} = \sum_{k=1}^{K}\sum_{c_i \in u_k}\mathbb{I}(q_i^{end} = Q)\lambda_i(\mathcal{L}(w_{g,k}^{(t+1)}, D_i) - \mathcal{L}(w_i^{(t-\tau_i)}, D_i)) \tag{32}$$

□

LEMMA 2. *From the A-phase to L-phase in arbitrary communication round, we have:*

$$\mathbb{E}[\mathcal{P}^{t,L}] - \mathcal{P}^{t,A} \leq \sum_{k=1}^{K}\sum_{c_i\in u_k}\lambda_i \sum_{q=0}^{Q-1}((\frac{\beta(\eta_i^{(t)})^2}{2} - \eta_i^{(t)})\mathbb{E}[\|\nabla\mathcal{L}(w_i^q)\|_2^2] + \frac{\beta(\eta_i^{(t)})^2}{2}\sigma^2) \tag{33}$$

PROOF. For client c_i in local training, from step q to step $q+1$:

$$\mathcal{L}(w_i^{q+1}) - \mathcal{L}(w_i^q) \leq \langle \nabla\mathcal{L}(w_i^q), w_i^{q+1} - w_i^q\rangle + \frac{\beta}{2}\|w_i^{q+1} - w_i^q\|_2^2$$
$$= -\eta_i^{(t)}\langle\nabla\mathcal{L}(w_i^q), \nabla\mathcal{L}(w_i^q, \xi_i^q)\rangle + \frac{\beta(\eta_i^{(t)})^2}{2}\|\nabla\mathcal{L}(w_i^q, \xi_i^q)\|_2^2 \tag{34}$$

Take expectation on both sides for a random batch ξ_i^q at step q,

$$\mathbb{E}[\mathcal{L}(w_i^{q+1})] - \mathcal{L}(w_i^q) \leq (\frac{\beta(\eta_i^{(t)})^2}{2} - \eta_i^{(t)})\|\nabla\mathcal{L}(w_i^q)\|_2^2 + \frac{\beta(\eta_i^{(t)})^2}{2}\sigma^2 \tag{35}$$

For an arbitrary client c_i starting L-phase at step q_i, and ends at step q_i' this round, we telescope them and get:

$$\mathbb{E}[\mathcal{L}(w_i^{q_i'})] - \mathcal{L}(w_i^{q_i}) \leq \sum_{q=q_i}^{q_i'}((\frac{\beta(\eta_i^{(t)})^2}{2} - \eta_i^{(t)})\|\nabla\mathcal{L}(w_i^q)\|_2^2 + \frac{\beta(\eta_i^{(t)})^2}{2}\sigma^2) \tag{36}$$

Lemma 2 is satisfied telescoping all clients. □

Now we start to prove Theorem 2.

PROOF. From L-phase in asynchronous round $t-1$ to C-phase in asynchronous round $t-1$, the overall loss does not change, for the model parameters does not change, so:

$$\mathcal{P}^{t-1,L} = \mathcal{P}^{t,C} \tag{37}$$

According to Lemma 1 and Lemma 2 we can get:

$$\mathbb{E}[\mathcal{P}^{t,L}] - \mathcal{P}^{t-1,L} \leq \sum_{k=1}^{K}\sum_{c_i\in u_k}\lambda_i\sum_{q=q_i}^{q_i'}(\frac{\eta_i^{(t-\tau_i)}QU(\alpha_i^{(t-\tau_i)}\theta O(U) + U)}{h_i^n}$$
$$+ (\frac{\beta(\eta_i^{(t)})^2}{2} - \eta_i)\mathbb{E}[\|\nabla\mathcal{L}(w_i^q)\|_2^2] + \frac{\beta(\eta_i^{(t)})^2}{2}\sigma^2) \tag{38}$$

Given $h_i = \frac{q_i' - q_i}{Q}$, the right side of Eq. (38) is always negative when

$$\alpha_t^{(t)} \leq \frac{(2 - \beta\eta_i^{(t)})\mathbb{E}[\|\nabla\mathcal{L}(w_i^q)\|_2^2] - \beta\eta_i^{(t)}\sigma^2}{2\theta QUO(U)} \cdot h_i - \frac{U}{\theta O(U)} \tag{39}$$

In the ideal case, the expectation of local gradient $\mathbb{E}[\|\nabla\mathcal{L}(w_i^q)\|_2^2]$ gradually decreases with time, which can be represented by a time-decay function. As the upper bound of staleness θ is related to the cluster size in an asynchronous scenario [9], for simplicity, we can rewrite Eq. (39) as follows:

$$\alpha_i^{(t)} \leq \frac{\Omega(t)h_i}{|C_k|} \tag{40}$$

where $|C_k|$ refers to the scale cluster u_k, h_i indicates the system resource of device i, and $\Omega(t)$ is a function that monotonically decreases over time. □

A.2 Additional Experimental Settings

A.2.1 Experimental Environment. We conduct experiments on a machine with AMD Ryzen 9 5950X 16-Core Processor CPU, and a NVIDIA GeForce RTX 3090 GPU. The code are implemented with Python 3.9.13 and PyTorch 1.13.1.

A.2.2 Configuration of Federation. We set client number $n = 100$ for MNIST, CIFAR-10, FEMNIST, HARBox, and $n = 7$ for IMU. For FEMNIST, we select the top 100 clients with the largest size of local data. For HARBox, we select the first 100 clients from the 120 clients. We apply a CNN to MNIST, CIFAR-10 and FEMNIST as [8]; and a two-layer fully connected network for IMU and HARBox as [26].

The wall-clock time is simulated following [30]. By default, we set the local training latency of slow devices as 5× that of normal devices, with 30% slow devices among all clients. This is because the speed of mainstream IoT platforms may differ by 5×, *e.g.*, between Raspberry Pi and NVIDIA Nano [32]. Since the concept of *round* differs in synchronous and asynchronous settings, we train for 500 rounds in synchronous settings, and for 50, 000 rounds in asynchronous settings.

A.2.3 Hyperparameters. For all the datasets, we set the batch size to 10, and the learning rate η to 0.01.

For all asynchronous methods, the basic decay coefficient is set as $\alpha = 0.3$. Other method-specific hyperparameters are set as below.

- FedProx [17]: proximal regularization term $\mu = 0.05$.
- FedAsync [41]: *hinge* version, with $a = 1$ and $b = 4$.
- FedBuff [25]: buffer size $K = 10$.
- CFL [28]: mean gradient bound $\varepsilon_1 = 0.4$; max gradient bound $\varepsilon_2 = 0.7$.
- ICFL [42]: start value $\alpha^*(0) = 0.85$, increasing factor $\varepsilon = 4.0$.
- CFL-Async: the max gradient bound is set higher than the synchronous version to avoid frequent triggering of the clustering condition. We set $\varepsilon_2 = 1$.
- ICFL-Async: same as ICFL, except that the time decay rate becomes ×0.01 due to the asynchrony nature.

For our CASA, the basic decay coefficient is set $\alpha_0 = 2$, and the time function is set as $\Omega(t) = (\frac{e}{2.8})^{kt}$, with k set to 0.001 for most cases, $\gamma = 0.15$. To ensure $\alpha_i^{(t)}$ is strictly less than 1, $|C_k|$ is actually set to $|C_k| + 3$. And to ensure clustering quality, we select the first $n = 10$ eigenvalues, and calculate eigengap from them.

A.2.4 Configuration of Datasets. We experiment with two types of clustering relationships:

- *Feature-skew based.* We select one IC dataset FEMNIST [3] and two HAR datasets IMU and HARBox [26] as the feature-skew case. These datasets possess realistic feature correlations, such as the writing habits in FEMNIST and the different activity postures in HAR.
- *Label-skew based.* We partition two IC datasets MNIST [16] and CIFAR-10 [15] with labels manually. We separate $n = 100$ clients into 4 groups, with a proportion of $\{0.2, 0.2, 0.3, 0.3\}$ following [42]. The labels are split following the same proportion. For instance, label 0 and label 1 are only held by group 0. To simulate the non-IID setting, we set Dirichlet distribution with $\alpha = 1$ inside each cluster.