Cache-Enabled Federated Learning Systems

Yuezhou Liu
Northeastern University
Boston, MA, USA
liu.yuez@northeastern.edu

Lili Su
Northeastern University
Boston, MA, USA
l.su@northeastern.edu

Carlee Joe-Wong
Carnegie Mellon University
Pittsburgh, PA, USA
cjoewong@andrew.cmu.edu

Stratis Ioannidis
Northeastern University
Boston, MA, USA
ioannidis@ece.neu.edu

Edmund Yeh
Northeastern University
Boston, MA, USA
eyeh@ece.neu.edu

Marie Siew
Carnegie Mellon University
Pittsburgh, PA, USA
msiew@andrew.cmu.edu

ABSTRACT

Federated learning (FL) is a distributed paradigm for collaboratively learning models without having clients disclose their private data. One natural and practically relevant metric to measure the efficiency of FL algorithms is the total wall-clock training time, which can be quantified by the product of the average time needed for a single iteration and the number of iterations for convergence. In this work, we focus on improving FL efficiency with respect to this metric through *caching*. Specifically, instead of having all clients download the latest global model from a parameter server, we select a subset of clients to access, with a smaller delay, a somewhat stale global model stored in caches. We propose *CacheFL* – a cache-enabled variant of FedAvg, and provide theoretical convergence guarantees in the general setting where the local data is imbalanced and heterogeneous. Armed with this result, we determine the caching strategies that minimize total wall-clock training time at a given convergence threshold for both stochastic and deterministic communication/computation delays. Through numerical experiments on real data traces, we show the advantage of our proposed scheme against several baselines, over both synthetic and real-world datasets.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; *Parallel algorithms*; *Distributed algorithms*; • **Networks** → *Network services*.

KEYWORDS

Federated Learning, caching, system design, training efficiency

ACM Reference Format:
Yuezhou Liu, Lili Su, Carlee Joe-Wong, Stratis Ioannidis, Edmund Yeh, and Marie Siew. 2023. Cache-Enabled Federated Learning Systems. In *The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3565287.3610264

MobiHoc '23, October 23–26, 2023, Washington, DC, USA

ACM ISBN 978-1-4503-9926-5/23/10…$15.00
https://doi.org/10.1145/3565287.3610264

1 INTRODUCTION

Federated learning (FL), proposed by McMahan et al. [22], is a promising paradigm that enables multiple entities (e.g, mobile devices, hospitals, and banks) to jointly train a machine learning model on their individual data without sending private data to a central location. Though proposed recently, FL has already been successful in many real-world applications, such as Gboard mobile keyboard [12] and Android Messages [7]. FL typically involves a parameter server and a collection of clients, each with its local data. The training procedure is divided into iterations, each of which includes the following three steps [22, 30]: 1) *Initialization of local model*: each client downloads the current global model. 2) *Local updates*: each client refines the downloaded global model by passing through local data for multiple epochs, and uploads the locally updated model to the server. 3) *Aggregation*: the server aggregates all the locally updated models to produce a new global model.

One main challenge of implementing FL in real world systems is resource heterogeneity: clients may have heterogeneous computation capabilities and communication (downlink/uplink) capacities. As a result, the time needed to complete a full iteration can be dominated by stragglers, leading to long training time and resource underutilization at faster clients. *Mitigating the straggler issue and reducing the time per iteration* is a critical and open problem in designing FL algorithms and systems [30, 36]. A number of works examine the straggler issue; we review them in Sec. 2.

In this work, we propose a new technique based on *caching* and design *cache-enabled federated learning systems*, aiming to reduce the overall wall-clock training time of FL algorithms by reducing per-iteration training time. Caching originates from computer systems, where copies of frequently used commands and data are stored in memory [1], and is applied in network systems such as content delivery networks (CDNs) [24] and information-centric networks (ICNs) [25] for faster data delivery. In the proposed cache-enabled FL systems, we deploy caches that store the global model at clients or the server. The stored copies of the global model are updated from time to time during the training process after a new global model is available. Clients can get such model copies faster, reducing the time needed to complete each iteration. Nevertheless, this reduction in per-iteration time comes with a price: *cached models may be stale*, which can lead to a degradation in convergence rate and an increase in the total iterations needed.

In light of this, we wish to design the best caching strategy that minimizes the overall wall-clock training time. This gives rise

to several challenges. First, determining the number of iterations needed to guarantee an error floor in the presence of stale updates is required; this, in itself, is a challenging task. Clients' data heterogeneity and imbalance make this even more difficult, as the effect of specific clients experiencing staleness to convergence should be quantified. Second, having determined the impact of caching on iterations, one needs to carefully design caching strategies that decide which clients compute their local updates from a cached global model, while others fetch the latest global model. The caching strategy should strike a favorable trade-off between (a) the time a client needs to retrieve a global model, affecting the time per iteration, and (b) the total number of iterations, which is necessary to minimize the overall wall-clock time of training.

Contributions. Our contributions can be summarized as follows:

- To the best of our knowledge, this is the *first work that considers caching of intermediate global models as a method to improve FL efficiency*. We design several cache-enabled FL systems by placing caches at clients or the parameter server, and discuss how *per-iteration training time* is reduced and how FL algorithms can be implemented in these systems.
- We *analyze the convergence of CacheFL (Federated Averaging [22] in proposed caching systems)* by providing convergence bounds as functions of clients using the cached models. We show that the number of iterations needed is increased by a factor depending on the data volume of these clients.
- We propose and solve optimization problems to find client caching strategies that appropriately trade off between reducing per-iteration time and increasing total iterations, minimizing the *overall wall-clock time* of FL algorithms.
- Through *extensive experimentation* on real mobile network traces, we show the advantages of the proposed scheme in faster per-iteration completion and wall-clock convergence over baselines for both i.i.d. and non-i.i.d. local data distributions, and for both synthetic and real-world datasets.

From a technical standpoint, prior FL convergence bounds generally assume either fully synchronous or entirely asynchronous model initialization and aggregation in each iteration, while our work extends and refines their analysis to quantify the effect of specific clients' use of stale models. Such analysis is particularly important as clients' data can be imbalanced and heterogeneous; the convergence bound depends on which clients experience staleness in caching systems.

Organizations. We present related works in Section 2. In Section 3, we introduce cache-enabled FL systems and the CacheFL algorithm. We make a convergence analysis of CacheFL in Section 4. In Section 5, we discuss optimization problems that minimize the overall wall-clock training time of CacheFL. We present numerical results in Section 6 and conclude in Section 7.

2 RELATED WORK

Hetergeneous and imbalanced data. Several algorithms have been proposed to incorporate heterogeneous and imbalanced data in FL, and guarantee good convergence rates. FedAvg [22] reduces iterations by letting clients train the local models for multiple epochs instead of one before aggregation. FedProx [20] and FedDyn [5] use regularization terms in local cost functions to limit the impact of heterogeneous local data. SCAFFOLD [15] uses control variates to correct the client-drift due to heterogeneity of local data in local updates, and FedCOMGATE [10] takes a gradient tracking approach to achieve the same goal. These algorithms can be implemented in the proposed caching systems by letting some clients make local updates based on cached global models. We discuss the convergence of FedAvg in our caching systems, while analysis of other algorithms in such systems is a possible future direction.

Reducing per-iteration training time. Effects of heterogeneous computation capabilities can be reduced by allowing clients to perform variable amounts of local computation, through different local epochs [20] or mini-batch sizes [19]. To reduce communication delay, algorithms with compression mechanisms [10, 29, 37, 38] or subspace methods [28] are proposed, where the models are compressed or projected to a subspace, respectively, before being transmitted between the server and clients. In FL with wireless communication, client selection, wireless resource blocks and transmit power are optimized under resource constraints to minimize the communication delay [3, 4]. To mitigate the straggler issue, each iteration can be stopped strategically with only a subset of local models being aggregated [2, 43]. Another way is to allow clients to asynchronously aggregate the local model at the server and start the next iteration without waiting for others [35, 36]. The proposed caching method is an independent and new direction that reduces per-iteration time, which can be combined with aforementioned techniques such as compression, wireless resource optimization, and asynchronism.

Networking and caching for FL. While there exist many works that use FL to optimize network operations as well as caching decisions in networks [33, 39–41], only a few works study the usage of networking and caching techniques to boost machine learning or federated learning performance. A few works [21, 31] study the optimal routing of data samples for distributed ML in networks to maximize model accuracy. In [44], caching for trained models is studied to minimize the inference error of mobile users. Caching for clients' gradients is considered in [9], where cached gradients are used for aggregation when clients drop out. Caching for stale statistics in vertical FL (VFL) is recently explored to enable local updates and improve communication efficiency [6]. VFL involves clients that hold disjoint features but overlap on the instances, meaning local training require the exchanges of intermediate computation statistics among clients. Caching can help reduce the cost of this communication. Our work is the first that proposes *caching of global models* to improve training efficiency.

3 CACHE-ENABLED LEARNING SYSTEMS

In this section, we introduce the general learning problem in FL as a preliminary, and then introduce the designs for several cache-enabled FL systems for use in different network scenarios. The goal of our system design is to enable FL algorithms to solve federated learning problems with smaller overall wall-clock time, i.e.,

$$\text{TIME} = T \times \mathcal{T}, \tag{1}$$

where T is the number of iterations for an FL algorithm to converge to an error floor, and $\mathcal{T}$ is the time needed for all participating clients to finish a single iteration, i.e., per-iteration training time [16]. We also propose CacheFL as an algorithm in the cache-enabled systems by extending the vanilla Federated Averaging [22].

3.1 Federated Learning Problem

Federated learning aims to train a single model by minimizing the model's empirical risk, i.e., the training loss over all data samples distributed across multiple clients. Consider a set of clients $k \in \mathcal{K}$, $|\mathcal{K}| = K$, each having a local dataset $\mathcal{D}_k = \{(\mathbf{u}_i, v_i)\}_{i=1}^{|\mathcal{D}_k|}$, where $\mathbf{u}_i$ and v_i are the features and label of the i-th sample, respectively. Let $\mathcal{D} = \cup_k \mathcal{D}_k$. We aim to minimize the following objective function:

$$F(\mathbf{w}) = \sum_{k\in\mathcal{K}} d_k F_k(\mathbf{w}), \tag{2}$$

where $d_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ and $F_k(\mathbf{w})$ are the local cost functions:

$$F_k(\mathbf{w}) = \frac{1}{|\mathcal{D}_k|} \sum_{(\mathbf{u}_i, v_i)\in\mathcal{D}_k} f(\mathbf{w}; \mathbf{u}_i, v_i), \quad k \in \mathcal{K}, \tag{3}$$

where $\mathbf{w}$ is the parameter vector of the model and $f(\mathbf{w}; \mathbf{u}_i, v_i)$ is the loss function associated with the i-th data sample. The loss function depends on the machine learning model and can be either convex (e.g., logistic regression) or non-convex (e.g., neural networks).

In general, FL algorithms consist of multiple iterations, each comprising the following steps: First, clients initialize their local models by downloading the current global model from the server. Second, each client tries to minimize its local cost F_k (usually for a few epochs) and produces an updated local model. Finally, clients upload their local models to the server, where they are aggregated to produce a new global model. In this way, clients jointly minimize the global objective F without sharing their local data.

3.2 Cache-enabled system design

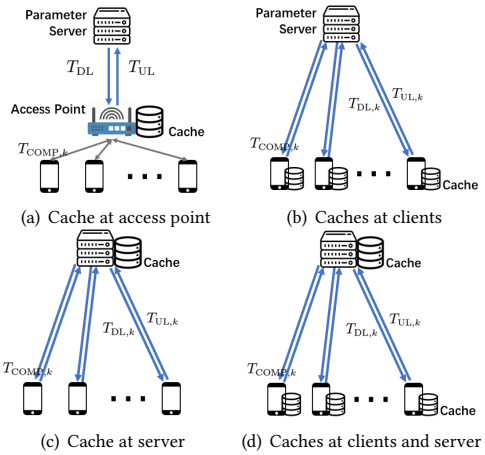
In FL, the per-iteration training time $\mathcal{T}$ is determined by the time for the slowest client (straggler) to finish downloading the current global model for initialization, making local updates, and uploading the updated model. Let $T_{\text{DL},k}$, $T_{\text{UL},k}$, and $T_{\text{COMP},k}$ be the time needed for client k to download the global model from the server, upload the local model to the server, and finish local computation, respectively. Without caching, the per-iteration completion delay for client k is

$$\mathcal{T}_k = T_{\text{DL},k} + T_{\text{COMP},k} + T_{\text{UP},k}, \tag{4}$$

We propose cache-enabled systems, where we mitigate the straggler issue by letting the stragglers obtain the global model from caches for initialization to reduce $\mathcal{T}_k$ and also $\mathcal{T}$.

System Design. We propose different caching systems by placing the caches at different locations, as appropriate for different network conditions and device computation capacities. In all schemes, we store the global model in a cache or caches, which is updated in each iteration after a new global model is available, thus bounding the staleness of the model in caches by *one iteration*. Aggregation is still synchronized as in classic FL where the server aggregates the local models it receives from all clients. On the other hand, caching allows clients to asynchronously initialize their local models in each iteration. In particular, by caching the global model, we give more flexibility to clients in getting the global model for initialization and make possible *parallelism* in some FL steps, thus reducing the time for finishing an iteration.

We denote by $\mathcal{K}_{\text{server}}$ and $\mathcal{K}_{\text{cache}}$, $\mathcal{K}_{\text{server}} \cup \mathcal{K}_{\text{cache}} = \mathcal{K}$, the sets of clients that use the latest global model and those that use the global model from the caches to initialize their local models, respectively, in each iteration. We refer to the strategy of partitioning $\mathcal{K}$ to $\mathcal{K}_{\text{server}}$ and $\mathcal{K}_{\text{cache}}$ as *client partition*, which is assumed to be static during the training. The proposed caching systems are shown in Fig. 1 and the reduced $\mathcal{T}_k$ by caching in each system is summarized in Table 1.



(a) Cache at access point (b) Caches at clients

(c) Cache at server (d) Caches at clients and server

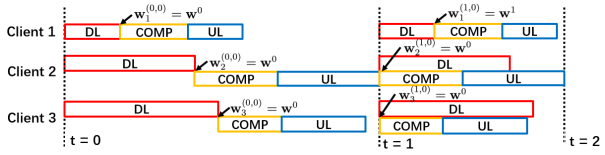
Figure 1: Cache-enabled FL systems. Each system has a set of clients $k \in \mathcal{K}$. The value near a communication link is the time for transmitting a model over this link and $T_{\text{COMP},k}$ is the computation time for k to finish its local updates.

Table 1: Per-iteration delay of different caching systems

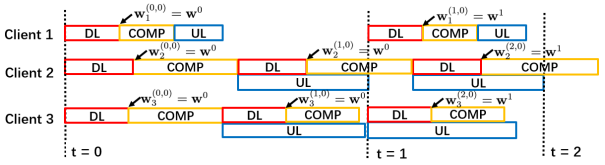
Caching systems	Per-iter delay of $k \in \mathcal{K}_{\text{cache}}$ ($\mathcal{T}_{\text{cache},k}$)
Cache at AP	$\max\{T_{\text{DL}}, T_{\text{COMP},k} + T_{\text{UL}}\}$
Caches at clients	$\max\{T_{\text{DL},k}, T_{\text{COMP},k} + T_{\text{UL},k}\}$
Cache at server	$\max\{T_{\text{UL},k}, T_{\text{COMP},k} + T_{\text{DL},k}\}$
Caches at both clients and server	$\min\{\max\{T_{\text{DL},k}, T_{\text{COMP},k} + T_{\text{UL},k}\}, \max\{T_{\text{UL},k}, T_{\text{COMP},k} + T_{\text{DL},k}\}\}$

3.2.1 Cache at access point (AP) (Fig. 1(a)). Consider a scenario where clients are located in proximity and communicate with the server through the same access point (AP), where a cache is placed. At the t-th iteration, when the server sends the latest model $\mathbf{w}_t$ to the clients, passing through the AP, the content in the cache is replaced by $\mathbf{w}_t$. Before that, the model in the cache is $\mathbf{w}_{t-1}$ from the previous iteration. Clients with more limited computation capabilities can start their local updates earlier by directly using $\mathbf{w}_{t-1}$ in the cache, saving the downloading time T_{DL} from the server to the AP. Other faster clients can wait for the latest global model $\mathbf{w}_t$. We ignore the delay on last-mile links between clients and the AP, as it can be absorbed into local computation time $T_{\text{COMP},k}$.

Assume that all clients share the same uplink/downlink delay between the server and AP in this case, then the slowest clients (with the largest $T_{\text{COMP},k}$) should be assigned to $\mathcal{K}_{\text{cache}}$ to reduce the per-iteration training time $\mathcal{T}$; otherwise, $\mathcal{T}$ is determined by these slowest clients regardless of whether other clients use the cache. Let $T_{\text{server}} = \max_{k\in\mathcal{K}_{\text{server}}} T_{\text{COMP},k}$ and $T_{\text{cache}} = \max_{k\in\mathcal{K}_{\text{cache}}} T_{\text{COMP},k}$. Without caching, the overall training time per iteration is $\mathcal{T} = T_{\text{DL}} + T_{\text{cache}} + T_{\text{UL}}$. With caching, the time is reduced to $\mathcal{T} = \max\{T_{\text{cache}} + T_{\text{UL}}, T_{\text{DL}} + T_{\text{server}} + T_{\text{UL}}\}$, leading to a time reduction of



(a) Caches at clients



(b) Cache at server

Figure 2: Example timing diagrams of two cases. In both cases, client 1 is in $\mathcal{K}_{\text{server}}$, and clients 2&3 are in $\mathcal{K}_{\text{cache}}$. Due to cache initialization, the per-iteration delay is $T_{\text{DL},2}+T_{\text{COMP},2}+T_{\text{UL},2}$ for both cases at $t=0$, while it is reduced to $T_{\text{COMP},2}+T_{\text{UL},2}$ and $T_{\text{DL},2}+T_{\text{COMP},2}$ in Fig. 2(a) and 2(b), respectively, starting from $t=1$.

$\min\{T_{\text{DL}}, T_{\text{cache}} - T_{\text{server}}\}$. When clients are heterogeneous in their computation capabilities (e.g., some clients may be mobile phones and some may be laptops with GPUs), such that we can find a client partition under which $T_{\text{cache}} - T_{\text{server}} \geq T_{\text{DL}}$, then we save time T_{DL} for an iteration by caching. Otherwise, we save $T_{\text{cache}} - T_{\text{server}}$.

3.2.2 Caches at clients (Fig. 1(b)). In practice, clients may be located in different areas and have heterogeneous communication delays $T_{\text{DL},k}$ and $T_{\text{UL},k}$. To generalize the *cache at AP* case, we can maintain a cache at each client. At the start of the t-th iteration, the server sends the latest global model $\mathbf{w}_t$ to all the clients, and the model is stored in the clients' caches. In this iteration, clients $k \in \mathcal{K}_{\text{server}}$ wait for $T_{\text{DL},k}$ of downlink transmission time and start local updates based on $\mathbf{w}_t$, while $k \in \mathcal{K}_{\text{cache}}$ directly access their caches for $\mathbf{w}_{t-1}$ and start immediately from $\mathbf{w}_{t-1}$. The downloading of the latest global model $\mathbf{w}_t$ to $k \in \mathcal{K}_{\text{cache}}$ is done in parallel while doing local updates and uploading the updated local model, reducing the delay of this client for an iteration from $\mathcal{T}_k$ given by (4) to

$$\mathcal{T}_{\text{cache},k} = \max\{T_{\text{DL},k}, T_{\text{COMP},k} + T_{\text{UL},k}\}, \tag{5}$$

saving time $T_{\text{DL},k}$ if $T_{\text{DL},k} \leq T_{\text{COMP},k}+T_{\text{UL},k}$ (e.g., client 2 in Fig. 2(a)), and $T_{\text{COMP},k} + T_{\text{UL},k}$ otherwise (e.g., client 3 in Fig. 2(a)). Globally, the per-iteration delay $\mathcal{T}$ is reduced from $\max_{k\in\mathcal{K}}\{\mathcal{T}_k\}$ to

$$\mathcal{T} = \max\left\{\{\mathcal{T}_{\text{cache},k}\}_{k\in\mathcal{K}_{\text{cache}}}, \{\mathcal{T}_k\}_{k\in\mathcal{K}_{\text{server}}}\right\}. \tag{6}$$

3.2.3 Cache at server (Fig. 1(c)). Sometimes, the uplink can be more expensive and slower than the downlink (e.g., mobile devices may have limited transmission power), which motivates us to save the uplink transmission time by putting a cache at the server.

The content in the cache at the server is replaced by the latest global model, each time the server finishes global aggregation. With this cache, after finishing local updates, a client k can either wait until all clients upload their updated models (at least for $T_{\text{UL},k}$) and then download the latest global model, or immediately download the model from the cache at the server and start the next round of local updates. By using the cache, uploading the local models of round $t-1$ can happen in parallel with the global model downloading and local computation for round t, reducing the delay of this client for an iteration from $\mathcal{T}_k$ given by (4) to

$$\mathcal{T}_{\text{cache},k} = \max\{T_{\text{UL},k}, T_{\text{COMP},k} + T_{\text{DL},k}\}. \tag{7}$$

A timing diagram is provided in Fig. 2(b). The per-iteration time $\mathcal{T}$ is also given by (6), with $T_{\text{cache},k}$ given by (7).

3.2.4 Caches at both server and clients (Fig. 1(d)). Further combining the previous cases leads to new FL systems. For example, we can have caches at both clients and server (combining Fig. 1(b) and 1(c)). Based on the relationships of downlink/uplink transmission delay and computation delay, clients $k \in \mathcal{K}_{\text{cache}}$ can decide to get the global model from any of the caches to best reduce their delay in an iteration. By (5) and (7), the minimum achievable delay for k in an iteration is given by the minimum of $\max\{T_{\text{DL},k}, T_{\text{COMP},k} + T_{\text{UL},k}\}$ and $\max\{T_{\text{UL},k}, T_{\text{COMP},k} + T_{\text{DL},k}\}$. Thus, client k may decide to get the global model from the cache at the client when $\max\{T_{\text{DL},k}, T_{\text{COMP},k} + T_{\text{UL},k}\} \leq \max\{T_{\text{UL},k}, T_{\text{COMP},k} + T_{\text{DL},k}\}$, and from the cache at the server otherwise.

Discussion. In these systems, cached global models are used by some clients $k \in \mathcal{K}_{\text{cache}}$ to reduce their per-iteration delay, thus mitigating the straggler issue and reducing $\mathcal{T}$. This technique is quite efficient when $T_{\text{DL},k}$ and $T_{\text{COMP},k} + T_{\text{UL},k}$ or $T_{\text{UL},k}$ and $T_{\text{COMP},k} + T_{\text{DL},k}$ are of a similar order of magnitude. As an example, when $T_{\text{DL},k} \approx T_{\text{COMP},k} + T_{\text{UL},k}$, by using the global model in the cache (at client k), client k can reduce its delay by half (compare (4) and (5)). On the other hand, this technique fails to efficiently reduce the per-iteration delay of a client when one part of the delay is much larger than other parts, making parallelism less fruitful (e.g., $T_{\text{UL},k}$ is extremely large due to an uplink connection failure). This should be solved by excluding client k in the client recruitment phase [26] or not aggregating the local model of k in certain training iterations (for temporary connection failures). In Sec. 6, we further show with experiments that the proposed systems can effectively reduce per-iteration training time, using statistics of computation and communication delay from real mobile networks. All aforementioned caching systems are easily implementable with low storage requirements. At the server or AP or each client device, we need only space to store a model parameter vector.

3.3 CacheFL algorithm

We next introduce the CacheFL algorithm by extending FedAvg [22] in our proposed cache-enabled systems, as described above.

3.3.1 FedAvg. minimizes $F(\mathbf{w})$ iteratively as follows:

Initialization of local models. At the start of each iteration $t = 1, \ldots, T-1$, the server broadcasts the current global model $\mathbf{w}^t$ to the clients to initialize their local models,

$$\mathbf{w}_k^{(t,0)} = \mathbf{w}^t, \quad \forall k \in \mathcal{K}. \tag{8}$$

Local Updates. Based on $\mathbf{w}_k^{(t,0)}$, clients attempt to minimize their local loss functions by stochastic gradient descent (SGD). To reduce the communication overhead, each client runs SGD for $\tau_{\max}$ epochs, before sending the local model to the server. Let $\mathbf{w}_k^{(t,\tau)}$ be the local model of client k at the t-th iteration and τ-th epoch. For each

Algorithm 1: CacheFL (cache-enabled FedAvg)

Input: Initial model $\mathbf{w}^0$, learning rate η, batch size B

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  for *Client* $k \in \mathcal{K}$ **do**
    Let $\mathbf{w}_k^{(t,0)} = \mathbf{w}^t$, if $k \in \mathcal{K}_{\text{server}}$
    Let $\mathbf{w}_k^{(t,0)} = \mathbf{w}^{t-1}$, if $k \in \mathcal{K}_{\text{cache}}$
    for $\tau = 0, \ldots, \tau_{max} - 1$ **do**
      SG: $\mathbf{g}_k^{(t,\tau)} = \frac{1}{B}\sum_{i=1}^{B} \nabla f(\mathbf{w}_k^{(t,\tau)}; \mathbf{u}_i, v_i)$
      Local update: $\mathbf{w}_k^{(t,\tau+1)} = \mathbf{w}_k^{(t,\tau)} - \eta \mathbf{g}_k^{(t,\tau)}$
    end
  end
  for *Server after receiving* $\mathbf{w}_k^{(t,\tau_{max})}$ *for all* $k \in \mathcal{K}$ **do**
    Update global model $\mathbf{w}^{t+1} = \sum_{k\in\mathcal{K}} d_k \mathbf{w}_k^{(t,\tau_{\max})}$
  end
end
return $\mathbf{w}^T$

$k \in \mathcal{K}$ and $\tau = 0, \ldots, \tau_{\max} - 1$, the client updates its local model as

$$\mathbf{w}_k^{(t,\tau+1)} = \mathbf{w}_k^{(t,\tau)} - \eta \mathbf{g}_k^{(t,\tau)}, \tag{9}$$

where η is the learning rate and $\mathbf{g}_k^{(t,\tau)}$ is the mini-batch stochastic gradient (SG), with batch size B, computed as

$$\mathbf{g}_k^{(t,\tau)} = \frac{1}{B}\sum_{(\mathbf{u},v)\in\mathcal{B}} \nabla f(\mathbf{w}_k^{(t,\tau)}; \mathbf{u}, v), \tag{10}$$

where $\mathcal{B}$ is a random subset ($|\mathcal{B}| = B$) sampled from $\mathcal{D}_k$. $\mathbf{g}_k^{(t,\tau)}$ is an unbiased estimator of local gradient, i.e., $\mathbb{E}[\mathbf{g}_k^{(t,\tau)}] = \nabla F_k(\mathbf{w}_k^{(t,\tau)})$.

Aggregation. After the local updates, each client k sends $\mathbf{w}_k^{(t,\tau_{\max})}$ to the server. At the end of iteration t, i.e., when the server receives all the local models from the clients, the server aggregates the local models to update the global model for the next iteration by

$$\mathbf{w}^{t+1} = \sum_{k\in\mathcal{K}} d_k \mathbf{w}_k^{(t,\tau_{\max})}. \tag{11}$$

3.3.2 CacheFL Learning Algorithm. We now extend traditional federated learning algorithms to cache-enabled systems. The essential change is that some of the clients perform local updates based on a global model stored in the cache, instead of the latest one. At the start of each iteration $t = 1, \ldots, T-1$, each client initializes the local model as following:

$$\mathbf{w}_k^{(t,0)} = \begin{cases} \mathbf{w}^t, & \text{if } k \in \mathcal{K}_{\text{server}}, \\ \mathbf{w}^{t-1}, & \text{if } k \in \mathcal{K}_{\text{cache}}, \end{cases} \tag{12}$$

instead of (8) as in classic FL algorithms. For $k \in \mathcal{K}_{\text{cache}}$, $\mathbf{w}^{t-1}$ is obtained from the cache, and for $k \in \mathcal{K}_{\text{server}}$, $\mathbf{w}^t$ is sent from the server. At the beginning of the training process, we have $\mathbf{w}_1^{(0,0)} = \cdots = \mathbf{w}_{|\mathcal{K}|}^{(0,0)} = \mathbf{w}^0$. If we consider FedAvg as the implemented algorithm in our proposed systems, then we replace the initialization step (8) by (12), while each client still makes local updates according to (9) and the server aggregates the updated local models according to (11). We call the resulting algorithm CacheFL (Alg. 1). We can easily implement other FL algorithms in the proposed caching systems, as we make no specific assumptions about the local updates and aggregation steps in our system designs. We note that all proposed caching systems lead to the same algorithm, as different cache deployments only affect the per-iteration delay reduction (see Table 1), instead of the resulting algorithm.

We have shown the benefits of caching in reducing the per-iteration training time $\mathcal{T}$. On the other hand, this benefit comes with the price of letting clients in $\mathcal{K}_{\text{cache}}$ make updates on a stale global model. One would naturally ask: 1) Can FL algorithms still converge with the use of caching? 2) If they converge, what is the effect of using a stale global model on the number of overall iterations for convergence? 3) Is the benefit worth the price? We formally answer the first two questions in the next section, by analyzing the convergence of CacheFL and providing a convergence bound as a function of the client partition ($\mathcal{K}_{\text{server}}$ and $\mathcal{K}_{\text{cache}}$). In Sec. 5, we further discuss the last question and show how to optimize the trade-off by formulating and solving optimization problems.

4 CONVERGENCE ANALYSIS OF CACHEFL

We analyze CacheFL (Alg. 1) from a theoretical perspective and provide a convergence bound for any given client partition.[1] We show that the bound is increased by a factor depending on the data volume fraction of clients $k \in \mathcal{K}_{\text{cache}}$.

4.1 Assumptions

Formally, we introduce the following assumptions for the convergence analysis, which are commonly made in the FL literature (e.g. [30]). Specifically, assumption (i&ii) is about the algorithm choice in CacheFL; assumptions (iii-v) are about the properties of the local cost functions, and assumption (vi) is about the data heterogeneity across clients. These assumptions allow the local data to be imbalanced and heterogeneously distributed.

(i) Each client $k \in \mathcal{K}$ participates in every iteration.

(ii) As in (2), the objective function $F(\mathbf{w})$ is the weighted average of local cost, i.e., $F(\mathbf{w}) = \sum_{k\in\mathcal{K}} d_k F_k(\mathbf{w})$, where $d_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$, to incorporate imbalanced local data.

(iii) Local cost functions $F_k(\mathbf{w})$ are convex and L-smooth,

$$F_k(\mathbf{w}) \le F_k(\mathbf{w}') + \langle \nabla F_k(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle + \frac{L}{2}||\mathbf{w} - \mathbf{w}'||^2$$

(iv) The stochastic gradient $\mathbf{g}_k^{(t,\tau)}$ is unbiased with uniformly bounded variance, namely

$$\mathbb{E}[||\mathbf{g}_k^{(t,\tau)} - \nabla F_k(\mathbf{w}_k^{(t,\tau)})|\mathbf{w}_k^{(t,\tau)}||^2] \le \sigma^2. \tag{13}$$

(v) The stochastic gradients are bounded in expectation,

$$\mathbb{E}[||\mathbf{g}_k^{(t,\tau)}||^2] \le G^2. \tag{14}$$

(vi) The weighted average of the dissimilarities between gradients of the local functions and the gradient of the global function is bounded, namely

$$\sum_{k\in\mathcal{K}} d_k ||\nabla F_k(\mathbf{w}) - \nabla F(\mathbf{w})||^2 \le \zeta^2. \tag{15}$$

[1] As most FL algorithms are variants of FedAvg, the convergence bound of CacheFL can be extended to incorporate other algorithms.

In this analysis, we do not consider partial client participation and non-convex loss functions (assumptions (i) and (iii)), to focus attention on the effects of caching. We provide an initial result of partial client participation at the end of this section, showing that our scheme still guarantees convergence without the full client participation assumption. In experiments, we try both convex and non-convex loss functions and compare our scheme with client selection related methods, e.g., eliminating stragglers. A more detailed discussion of CacheFL with partial client participation and non-convex losses will be a future work. Inequality (13) reflects the sample dissimilarity within local datasets. Due to the heterogeneity of local datasets, $\mathbf{g}_k^{(t,\tau)}$ is not an unbiased estimator of the global gradient in general, i.e., $\nabla F_k(\mathbf{w}_k^{(t,\tau)}) \neq \nabla F(\mathbf{w}_k^{(t,\tau)})$, and we have (15) to describe this heterogeneity.

4.2 Theoretical Results

We base our analysis on the proofs for FedAvg in [30] and [8] and extend them to incorporate the use of caching for global models. Besides bounding the convergence, we also need to quantify the dependence on the clients that use the cache, i.e., the bound should be a function of client partition.

To handle iterates from multiple clients, a concept of shadow/virtual sequence is adopted (commonly used in and originating from decentralized optimization), defined as

$$\overline{\mathbf{w}}^{(t,\tau)} := \textstyle\sum_{k\in\mathcal{K}} d_k \mathbf{w}_k^{(t,\tau)}, \tag{16}$$

where $d_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$. Letting $\overline{\mathbf{w}}^T = \frac{1}{\tau_{\max} T}\sum_{t=0}^{T-1}\sum_{\tau=1}^{\tau_{\max}} \overline{\mathbf{w}}^{(t,\tau)}$ and $\mathbf{w}^*$ be the minimizer of objective function F, we show that $\mathbb{E}[F(\overline{\mathbf{w}}^T)] - F(\mathbf{w}^*)$ is upper bounded by a quantity decreasing with T.

Following [30], we first show that there is a progress in each iteration, by proving that $\mathbb{E}[\frac{1}{\tau_{\max}}\sum_{\tau=1}^{\tau_{\max}} F(\overline{\mathbf{w}}^{(t,\tau)}) - F(\mathbf{w}^*)]$ is bounded by the difference of a potential function evaluated at the start and the end of each iteration, plus additional small error terms:

Lemma 1. *If the client learning rate satisfies $\eta \le \frac{1}{4L}$, then*

$$\begin{aligned}&\mathbb{E}\left[\tfrac{1}{\tau_{max}}\textstyle\sum_{\tau=1}^{\tau_{max}} F(\overline{\mathbf{w}}^{(t,\tau)}) - F(\mathbf{w}^*)|\mathcal{F}^{(t,0)}\right]\\ \le&\tfrac{1}{2\eta\tau_{max}}\left(W^{(t,0)} - \mathbb{E}\left[W^{(t,\tau_{max})}|\mathcal{F}^{(t,0)}\right]\right) + \eta\sigma^2\textstyle\sum_{k\in\mathcal{K}} d_k^2\\ &+ \tfrac{L}{\tau_{max}}\textstyle\sum_{\tau=0}^{\tau_{max}-1}\sum_{k\in\mathcal{K}} d_k\mathbb{E}\left[||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2|\mathcal{F}^{(t,0)}\right]\end{aligned}$$

where $W^{(t,\tau)} := ||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}^*||^2$ and $\mathcal{F}^{(t,0)}$ is the σ-field representing all historical information up to the start of iteration t.

Proof Sketch. It follows from Lemma 1 in [30], and further incorporates imbalanced local dataset sizes. See details in Appendix A.1. □

Next, we show that all client iterates remain close to the virtual/shadow sequence, i.e., the weighted average $\sum_{k\in\mathcal{K}} d_k||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2$ is bounded in expectation.

Lemma 2. *Let $V^{(t,\tau)} = \sum_{k\in\mathcal{K}} d_k||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2$, we have*

$$\mathbb{E}\left[V^{(t,\tau)}\right] \le \tau_{max}^2\eta^2G^2 + 2\eta^2G^2 + 2\tau_{max}\eta^2\zeta^2.$$

Proof Sketch. It is based on Lemma E.2 in [8], and makes necessary extensions to incorporate caching. With the use of the cached global model ($\mathcal{K}_{\text{cache}} \neq \emptyset$), we do not have $V^{(t,0)} = 0$ as in classic FedAvg, thus should bound $V^{(t,0)}$ carefully. See details in Appendix A.2. □

Combining Lemma 1 and Lemma 2 leads to the following theorem that gives the convergence bound of CacheFL:

Theorem 1. *Under the aforementioned assumptions (i)-(v), if the client learning rate satisfies $\eta \le \frac{1}{4L}$, then one has*

$$\begin{aligned}\mathbb{E}\left[F(\overline{\mathbf{w}}^T)\right] - F(\mathbf{w}^*) \le&\frac{\Delta^2}{2\eta\tau_{max}T} + \eta\sigma^2 d_{sq} + 2\tau_{max}\eta^2L\zeta^2\\ &+ 2\eta^2LG^2 + \tau_{max}^2\eta^2LG^2,\end{aligned} \tag{17}$$

where $\Delta := \sqrt{1+d_{cache}}\cdot||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||$, $d_{sq} = \sum_{k\in\mathcal{K}} d_k^2$, and $d_{cache} = \sum_{k\in\mathcal{K}_{cache}} d_k$. If further choosing the learning rate as

$$\eta = \min\left\{\tfrac{1}{4L}, \tfrac{\Delta}{d_{sq}^{\frac{1}{2}}\tau_{max}^{\frac{1}{2}}T^{\frac{1}{2}}\sigma}, \tfrac{\Delta^{\frac{2}{3}}}{\tau_{max}^{\frac{2}{3}}C}, \tfrac{\Delta^{\frac{2}{3}}}{\tau_{max}^{\frac{1}{3}}C}, \tfrac{\Delta^{\frac{2}{3}}}{\tau_{max}C}\right\},$$

where $C = T^{\frac{1}{3}}L^{\frac{1}{3}}G^{\frac{2}{3}}$, we have

$$\begin{aligned}\mathbb{E}[F(\overline{\mathbf{w}}^T)] - F(\mathbf{w}^*) \le&\frac{2L\Delta^2}{\tau_{max}T} + \frac{2d_{sq}^{\frac{1}{2}}\sigma\Delta}{\sqrt{\tau_{max}T}} + \frac{3L^{\frac{1}{3}}\zeta^{\frac{2}{3}}\Delta^{\frac{4}{3}}}{\tau_{max}^{\frac{1}{3}}T^{\frac{2}{3}}}\\ &+ \frac{3L^{\frac{1}{3}}G^{\frac{2}{3}}\Delta^{\frac{4}{3}}}{\tau_{max}^{\frac{2}{3}}T^{\frac{2}{3}}} + \frac{2L^{\frac{1}{3}}G^{\frac{2}{3}}\Delta^{\frac{4}{3}}}{T^{\frac{2}{3}}}.\end{aligned} \tag{18}$$

Proof. Combining Lemma 1 and Lemma 2, we have

$$\begin{aligned}&\mathbb{E}\left[\tfrac{1}{\tau_{\max}}\textstyle\sum_{\tau=1}^{\tau_{\max}} F(\overline{\mathbf{w}}^{(t,\tau)}) - F(\mathbf{w}^*)|\mathcal{F}^{(t,0)}\right]\\ \le&\tfrac{1}{2\eta\tau_{\max}}\left(W^{(t,0)} - \mathbb{E}\left[W^{(t,\tau_{\max})}|\mathcal{F}^{(t,0)}\right]\right) + \eta\sigma^2\textstyle\sum_{k\in\mathcal{K}} d_k^2\\ &+ \tau_{\max}^2\eta^2LG^2 + 2\eta^2LG^2 + 2\tau_{\max}\eta^2L\zeta^2.\end{aligned} \tag{19}$$

Let $d_{\text{server}} := \sum_{k\in\mathcal{K}_{\text{server}}} d_k$, by convexity, $||\overline{\mathbf{w}}^{(t,0)} - \mathbf{w}^*||^2 = ||d_{\text{cache}}(\overline{\mathbf{w}}^{(t-2,\tau_{\max})} - \mathbf{w}^*) + d_{\text{server}}(\overline{\mathbf{w}}^{(t-1,\tau_{\max})} - \mathbf{w}^*)||^2 \le d_{\text{cache}}||\overline{\mathbf{w}}^{(t-2,\tau_{\max})} - \mathbf{w}^*||^2 + d_{\text{server}}||\overline{\mathbf{w}}^{(t-1,\tau_{\max})} - \mathbf{w}^*||^2$, for $t \ge 2$. We also have $||\overline{\mathbf{w}}^{(1,0)} - \mathbf{w}^*||^2 = ||d_{\text{cache}}(\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*) + d_{\text{server}}(\overline{\mathbf{w}}^{(0,\tau_{\max})} - \mathbf{w}^*)||^2 \le d_{\text{cache}}||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||^2 + d_{\text{server}}||\overline{\mathbf{w}}^{(0,\tau_{\max})} - \mathbf{w}^*||^2$. Thus, by telescoping, we have

$$\textstyle\sum_{t=0}^{T-1}(W^{(t,0)} - W^{(t,\tau_{\max})}) \le (1+d_{\text{cache}})||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||^2.$$

By above, telescoping (19) with t from 0 to $T-1$ and using the convexity of F, finishes the proof. □

Discussion. Compared to the convergence bound of FedAvg (e.g., Theorem 1 in [30]), the main difference is that CacheFL has an extra term $\sqrt{1+d_{\text{cache}}}$ in the definition of Δ. If no client uses the cache ($d_{\text{cache}} = 0$), we recover the convergence bound for FedAvg in [30]. If all clients use the cache ($d_{\text{cache}} = 1$), running CacheFL is equivalent to running one FedAvg at the odd iterations and running another FedAvg at the even iterations. Thus, CacheFL intuitively needs twice the iterations to achieve the same error as FedAvg. We also see this from the convergence bound, where we have $\Delta^2_{\text{CacheFL}} = 2||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||^2 = 2\Delta^2_{\text{FedAvg}}$. As T increases the expected error is dominated by the second term of the RHS in (18). Thus, having $\Delta^2_{\text{CacheFL}} = 2\Delta^2_{\text{FedAvg}}$ leads to the same result that CacheFL needs twice iterations for the same error.

For arbitrary d_{cache}, by (18) and letting the expected error be smaller than ϵ, we have the following bound on T,

$$T(\epsilon) = \mathcal{O}\left(\frac{L\Delta^2}{\tau_{\max}\epsilon} + \frac{d_{\text{sq}}\sigma^2\Delta^2}{\tau_{\max}\epsilon^2} + \frac{L^{\frac{1}{2}}\zeta\Delta^2}{\tau_{\max}^{\frac{1}{2}}\epsilon^{\frac{3}{2}}} + \frac{L^{\frac{1}{2}}G\Delta^2}{\tau_{\max}\epsilon^{\frac{3}{2}}} + \frac{L^{\frac{1}{2}}G\Delta^2}{\epsilon^{\frac{3}{2}}}\right), \quad (20)$$

where $\Delta^2 = (1+d_{\text{cache}})||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||^2$, which gives the dependence of T on d_{cache}, i.e., $T \propto 1 + d_{\text{cache}}$. In Sec. 5, we use this dependence to approximate the total number of iterations required for training.

Partial client participation. While we mainly focus on full client participation in this section, we also provide the following initial result with client sampling: in each iteration t, a subset $\mathcal{S}^{(t)}$ of clients with $|\mathcal{S}^{(t)}| = S$, is sampled uniformly from K clients for training. For $k \in \mathcal{S}^{(t)}$, the local update is given by $\mathbf{w}_k^{(t,\tau+1)} = \mathbf{w}_k^{(t,\tau+1)} - \mu\frac{K}{S}\mathbf{g}_k^{(t,\tau)}$, such that $\mathbb{E}_{\mathcal{S}^{(t)}}\frac{1}{K}\sum_{k\in\mathcal{S}^{(t)}}\frac{K}{S}\mathbf{g}_k^{(t,\tau)} = \frac{1}{K}\sum_{k\in\mathcal{K}}\mathbf{g}_k^{(t,\tau)}$.

Theorem 2. *Under the aforementioned assumptions (ii)-(v) with the above client sampling and local update methods, if the client learning rate satisfies $\eta \le \frac{1}{4L}$ and $d_k = \frac{1}{K}$ for all $k \in \mathcal{K}$, then*

$$\mathbb{E}\left[F(\overline{\mathbf{w}}^T)\right] - F(\mathbf{w}^*) \le \frac{\Delta^2}{2\eta\tau_{max}T} + \eta\frac{\sigma^2}{K} + 2\tau_{max}\eta^2L\zeta^2 + 2\eta^2\frac{K}{S}LG^2 + 3\frac{S}{K-S}\tau_{max}^2\eta^2LG^2, \quad (21)$$

where $\Delta := \sqrt{1 + \frac{S}{K}\frac{K_{cache}}{K}} \cdot ||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||$.

Proof. See details in Appendix B. □

5 OPTIMIZING OVERALL TRAINING TIME

In this section, we formulate optimization problems to decide the client partition strategy that minimizes the overall wall-clock training time of CacheFL by minimizing the following trade-off: As shown in Sec. 3, the global per-iteration training time $\mathcal{T}$ is a non-increasing set function of $\mathcal{K}_{\text{cache}}$. If more clients use the cache to reduce their delay, then there is a chance to decrease $\mathcal{T}$ more. On the other hand, as discussed in Sec. 4, an increase of the clients in $\mathcal{K}_{\text{cache}}$ leads to an increase in the number of needed iterations T.[2]

By (20), we upper bound the total number of iterations needed for CacheFL to reach a certain error as

$$T = a \times (1 + d_{\text{cache}}) = a \times (1 + \textstyle\sum_{k\in\mathcal{K}} d_k x_k), \quad (22)$$

where a is a constant and $x_k \in \{0,1\}$, $k \in \mathcal{K}$, are the decision variables indicating whether client k uses the cache, i.e., $\mathcal{K}_{\text{cache}} = \{k|x_k = 1, k \in \mathcal{K}\}$ and $\mathcal{K}_{\text{server}} = \{k|x_k = 0, k \in \mathcal{K}\}$. Let $\mathbf{x} = \{x_k\}_{k\in\mathcal{K}}$ be the vector of decision variables that we will optimize.

Let $T_{\text{DL},k}^t$, $T_{\text{UL},k}^t$, and $T_{\text{COMP},k}^t$ be the time needed for client k to download the global model from the server, upload the local model to the server, and finish local computation at t-th iteration, respectively. As discussed in Sec. 3, the time for $k \in \mathcal{K}_{\text{server}}$ to finish the t-th iteration is

$$\mathcal{T}_k^t = T_{\text{UL},k}^t + T_{\text{COMP},k}^t + T_{\text{DL},k}^t,$$

and the time for $k \in \mathcal{K}_{\text{cache}}$ to finish the t-th iterations (i.e., $\mathcal{T}_{\text{cache},k}^t$) varies for different caching schemes and is summarized in Table 1.

[2]This section considers full client participation, using Theorem 1, but it is ready to be extended to the partial client participation case.

Algorithm 2: Prob. (23)

Input: $x_1 = \cdots = x_K = 0$, $\mathbf{x}_{\text{out}} = \mathbf{x}_0 = \mathbf{x}$
1 Compute $T_{\min} = \text{TIME}(\mathbf{x}_0)$, $\mathcal{T}_{\max} = \mathcal{T}(\mathbf{x}_0)$
2 **for** $k = 0, 1, \ldots, K$ **do**
3 If $\mathcal{T}_{\max} > \mathcal{T}_k$: **Break**
4 Set $x_k = 1$ and $\mathbf{x}_k = \mathbf{x}$
5 Compute $\mathcal{T}_{\max} = \mathcal{T}(\mathbf{x}_k)$
6 If $T_{\min} > \text{TIME}(\mathbf{x}_k)$: $T_{\min} = \text{TIME}(\mathbf{x}_k)$, $\mathbf{x}_{\text{out}} = \mathbf{x}_k$
7 **end**
8 **return** $\mathbf{x}_{\text{out}}$

Then, the training time for t-th iteration is,

$$\mathcal{T}^t = \max\left\{\left\{\mathcal{T}_{\text{cache},k}^t \cdot x_k\right\}_{k\in\mathcal{K}}, \left\{\mathcal{T}_k^t \cdot (1 - x_k)\right\}_{k\in\mathcal{K}}\right\}$$

We formulate optimization problems considering different cases of transmission and computation delay, i.e., deterministic, offline random, and online random:

Deterministic case. We start with the transmission delay and computation delay being fixed, i.e., $T_{\text{DL},k}^t = T_{\text{DL},k}$, $T_{\text{UL},k}^t = T_{\text{UL},k}$, and $T_{\text{COMP},k}^t = T_{\text{COMP},k}$ for all t. Thus, we have $\mathcal{T}^t = \mathcal{T}$ being constant across t. This assumption relates to cross-silo FL, where clients have static and reliable links with the server, and allocate abundant computation resources for local updates [42]. We formulate the following problem to minimize the total wall-clock time of training:

$$\min \quad \text{TIME} = \mathcal{T} \cdot T \quad (23)$$

$$\text{s.t.} \quad x_k \in \{0,1\}, \forall k \in \mathcal{K} \quad (24)$$

Though the problem is an integer optimization problem, which is often hard to solve in reasonable time, we exploit the structure of (23) to derive an efficient algorithm. Assume the clients being indexed in decreasing order with the value $\mathcal{T}_k = T_{\text{DL},k} + T_{\text{COMP},k} + T_{\text{UL},k}$. Prob. (23) can be solved by trying at most $K+1$ candidate solutions $\mathbf{x}_k$ for $k = 0, \ldots, K$ such that each of them set the first k decision variables to 1 and the remaining to 0 (i.e., $x_1 = \cdots = x_k = 1$ and $x_{k+1} = \cdots = x_K = 0$). The final solution is the candidate with minimum TIME. We formalize the procedure in Alg. 2.

Theorem 3. *Alg. 2 finds the optimal solution of Problem* (23) *with $\mathcal{O}(K)$ time complexity for sorted clients.*

Proof. $\mathcal{T}$ and T are non-increasing and non-decreasing set functions of $\mathcal{K}_{\text{cache}}$. Moreover, for $i < j$ and $x_i = 0$, setting x_j to 0 or 1 leads to the same $\mathcal{T}$, as $\mathcal{T}_i \ge \mathcal{T}_j \ge \mathcal{T}_{\text{cache},j}$. Then, for any solution $\mathbf{x}'$ aside from the candidates, suppose the first k coordinates are 1, and the $(k+1)$-th is not, then we can set all coordinates after $k+1$ to zero, making $\mathcal{T}$ unchanged but T decrease, leading to a better solution, our candidate $\mathbf{x}_k$. Thus, trying $\mathbf{x}_k$ for $k = 0, \ldots, K$ guarantees that we can find the optimal solution. This traversal can stop earlier when finding $\mathcal{T}(\mathbf{x}_k) \ge \mathcal{T}_{k+1}$ (line 5 in Alg. 2), as further assigning client $i > k$ to $\mathcal{K}_{\text{cache}}$ will not further reduce $\mathcal{T}$. □

Offline random case. We now assume that the transmission delay and computation delay are stationary random processes indexed by t, with $\mathbb{E}[T_{\text{DL},k}^t] = \mu_{\text{DL},k}$, $\mathbb{E}[T_{\text{UL},k}^t] = \mu_{\text{UL},k}$, and $\mathbb{E}[T_{\text{COMP},k}^t] = \mu_{\text{COMP},k}$. This assumption corresponds especially to cross-device

FL that runs during off-peak hours, where the delay can vary from time to time but their statistics are rather stable. In this case, the statistics of the random delay can be collected beforehand (offline).

We wish to minimize the expected wall-clock time of training, i.e., $\mathbb{E}[\mathcal{T}^t] \cdot T$. However, in general, we cannot compute the closed-form expression of $\mathbb{E}[\mathcal{T}^t]$, the expectation of the maxima of random variables, which motivates us to consider approximations. For such expectation and with any distribution of the delays, Hamza's theory [11] provides a bound. As $\mathcal{T}^t_{\text{cache},k}$ has different expressions for different caching schemes, we choose the caches at clients case (Fig. 1(b)) as an example, where $\mathcal{T}^t_{\text{cache},k}$ is given by (5). For other systems, we can obtain similar bounds with Hamaza's theory.

Lemma 3 (Theorem 1 in [11]). *For $\mathcal{T}^t_{cache,k}$ given by* (5)*, we have*

$$\overline{\mu} \le \mathbb{E}[\mathcal{T}^t] \le \overline{\mu} + \frac{3K-1}{3K}\mu_{max}, \text{ where}$$
$$\overline{\mu} = \tfrac{1}{3K}\left(\textstyle\sum_{k\in\mathcal{K}}(M_{1,k}+M_{2,k})\cdot x_k + \sum_{k\in\mathcal{K}} M_{3,k}\cdot(1-x_k)\right),$$
$$\mu_{max} = \max\left\{\left\{M_{1,2,k}\cdot x_k\right\}_{k\in\mathcal{K}}, \left\{M_{3,k}\cdot(1-x_k)\right\}_{k\in\mathcal{K}}\right\},$$

and $M_{1,2,k} = \max\{M_{1,k}, M_{2,k}\}$, with $M_{1,k}$, $M_{2,k}$, and $M_{3,k}$ be the expectations of the maxima of $3K$ copies of the random variables $T^t_{UL,k} + T^t_{COMP,k}$, $T^t_{DL,k}$ and $T^t_{UL,k} + T^t_{COMP,k} + T^t_{DL,k}$, respectively.

Thus, one can approximate $\mathbb{E}[\mathcal{T}^t]$ by $\overline{\mu}$ or $\mu_{\max}$. A remaining challenge is that $M_{1,k}$, $M_{2,k}$ and $M_{3,k}$ usually also have no closed-form values for arbitrary K. For certain distributions, nice approximations exist [13]: if the communication and computation delay are Gaussian random variables: $T^t_{\text{DL},k} \sim N(\mu_{\text{DL},k}, \sigma^2_{\text{DL},k})$, $T^t_{\text{UL},k} \sim N(\mu_{\text{UL},k}, \sigma^2_{\text{UL},k})$, and $T^t_{\text{COMP},k} \sim N(\mu_{\text{COMP},k}, \sigma^2_{\text{COMP},k})$, when $K \to \infty$, we have

$$M_{1,k} = \mu_{\text{UL},k} + \mu_{\text{COMP},k} + \sqrt{2(\sigma^2_{\text{UL},k} + \sigma^2_{\text{COMP},k})}\cdot\sqrt{\log K},$$

and $M_{2,k}$, $M_{3,k}$ can be computed similarly. For arbitrary distribution and smaller K, we use sampling to estimate $M_{1,k}$, $M_{2,k}$ and $M_{3,k}$, as the statistics of each delay are already known in the offline random case, and the sampling is needed only once before the training.

If we approximate $\mathbb{E}[\mathcal{T}^t]$ by $\mu_{\max}$, we have

$$\min \quad \mu_{\max}\cdot T \qquad \text{subject to Eq. (24)} \tag{25}$$

which has the same form as Prob. (23), and the same algorithm with $O(K)$ complexity can find its optimal solution.

Another option is approximating $\mathbb{E}[\mathcal{T}^t]$ by $\overline{\mu}$. We have

$$\min \quad \overline{\mu}\cdot T \qquad \text{subject to Eq. (24)} \tag{26}$$

which is a quadratic unconstrained binary optimization (QUBO) problem, and can be solved by traditional combinatorial optimization methods, such as Simulated Annealing.

Online random case. We further consider the transmission and computation delay as non-stationary random processes. The statistics are changing as in cross-device FL with clients having limited computing resources (training processes need to compete with other processes) and unreliable wireless communication links [30].

In this case, client partition needs to change over time with the evolution of delay processes, and we should predict the delay for each iteration using past observations. Gaussian Process Regression (GPR) is a candidate method for prediction, which is nonparametric and provides an analytical way to measure the uncertainty of the prediction [14]. While other prediction methods are also possible, their selection is out of the scope of this paper.

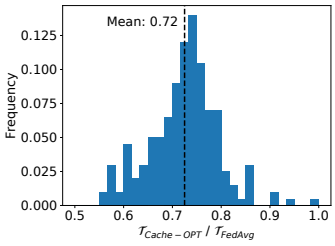


Figure 3: Distribution of per-iteration delay of Cache-OPT. Compared with FedAvg, CacheFL-OPT reduces the per-iteration delay by 28% on average.

At the t-th iteration, given the predicted delay $\hat{T}^t_{\text{DL},k}$, $\hat{T}^t_{\text{UL},k}$, and $\hat{T}^t_{\text{COMP},k}$, we can compute $\hat{\mathcal{T}}^t_{\text{cache},k}$ (which varies for different caching schemes) and $\hat{\mathcal{T}}^t_k = \hat{T}^t_{\text{COMP},k} + \hat{T}^t_{\text{UL},k} + \hat{T}^t_{\text{DL},k}$. Then, the predicted per-iteration training time for this iteration is,

$$\hat{\mathcal{T}}^t = \max\left\{\left\{\hat{\mathcal{T}}^t_{\text{cache},k}\cdot x_k\right\}_{k\in\mathcal{K}}, \left\{\hat{\mathcal{T}}^t_k\cdot(1-x_k)\right\}_{k\in\mathcal{K}}\right\}.$$

To decide the appropriate client partition in each iteration t, we formulate the following heuristic problem,

$$\min \quad \hat{\mathcal{T}}^t\cdot(1+\textstyle\sum_{k\in\mathcal{K}} d_k x_k) \qquad \text{s.t. Eq. (24)} \tag{27}$$

This problem is again of the same form as Prob. (23) and can be solved in $O(K)$ time. Note that by (20), the remaining number of iterations at any iteration can also be approximated as a constant times $1+d_{\text{cache}}$. Thus, by solving the above problem at each iteration t, one greedily minimizes the remaining wall-clock training time, assuming the estimation $\hat{T}^t_{\text{DL},k}$, $\hat{T}^t_{\text{UL},k}$, and $\hat{T}^t_{\text{COMP},k}$ are the delay for the following iterations.

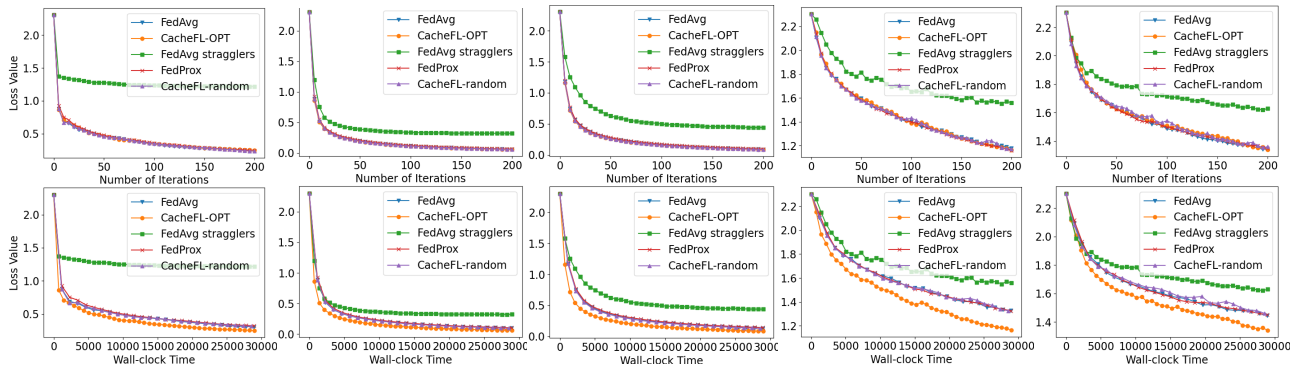
Extending the convergence analysis in Sec. 4, we can show that CacheFL can still converge if we change the client partition from time to time. From the practical design perspective, we may set $\mathcal{K}_{\text{server}} = \mathcal{K}$ for the first few iterations, as the estimation is not accurate enough. Changing the partition of $\mathcal{K}_{\text{cache}}$ and $\mathcal{K}_{\text{server}}$ by solving Prob. (27) can also be done less frequently, only when there is a large change in the prediction of the delay, instead of in every iteration. The server can adjust caching strategy in real-time without significant overhead, as clients can collect past observations of their own delays and use them to predict future delays. This information can be sent to the server when the server and clients exchange control information.

6 NUMERICAL EVALUATION

We now present empirical results of CacheFL compared with several baselines, to show the advantage of cache-enabled systems.[3]

Experiment Setting. We consider synthetic and real datasets, curated from prior work in FL: synthetic data Synthetic(1,1) [20], and real image data MNIST [18], CIFAR-10 [17], and FMNIST [34]. We study both convex and non-convex classification problems on

[3] Our code and data are publicly available at https://github.com/NormanLiu/CacheFL.



(a) Synthetic(1,1) (b) MNIST-iid (c) MNIST-noniid (d) CIFAR10-iid (e) CIFAR10-noniid

Figure 4: Convergence comparison of different algorithms in iterations and wall-clock time, with logistic regression.

these datasets using logistic regression, MLP, and CNN with cross-entropy loss. We consider $K = 50$ clients.[4] The number of data samples at each client is imbalanced, decided by the Zipf distribution with power 2 and scaled by 50, leading to the minimum and maximum local dataset sizes of 50 and 700. Synthetic(1,1) generates non-i.i.d. data for each client. For MNIST, CIFAR-10 and FMNIST, we consider both i.i.d. and non-i.i.d. data partitions. In the latter case, each client has samples of 2 classes out of 10.

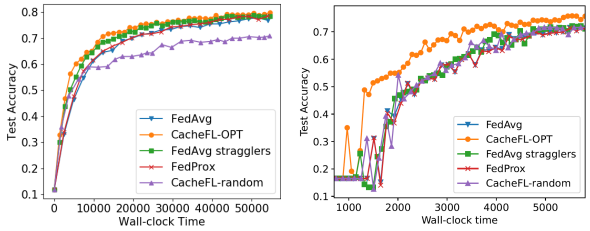
To compare the wall-clock training time, we simulate the uplink/downlink transmission delay and computation delay as follows: The computation delay in each iteration is randomly sampled from a real-world trace, collected as in [27], with values from 2 to 25. The downlink/uplink transmission throughput of client k in each iteration is randomly sampled from the Mobiperf trace [23], with resulting transmission delay ranging from 2 to 50. The compared FL algorithms include FedAvg [22] and the following:

- CacheFL-OPT: The proposed scheme given by Alg. 1, with $\mathcal{K}_{\text{cache}}$ and $\mathcal{K}_{\text{server}}$ decided by solving Prob. (25).
- CacheFL-random: Alg. 1, with a random client partition, changed every 10 iterations.
- FedAvg-stragglers: FedAvg with clients in $\mathcal{K}_{\text{cache}}$ (as stragglers) being excluded from the training process.
- FedProx [20], with penalty constant set to 0.1.

The learning rate is set to 0.05 for CNN and decided by grid-search as in [32] for logistic regression and MLP. The number of local epochs is 5. The mini-batch size is 10 for logistic regression and 5 for MLP and CNN.

Per-iteration training time. We show how CacheFL-OPT can efficiently reduce the per-iteration training time in Fig. 3, which gives the distribution of $\mathcal{T}_{\text{CacheFL-OPT}}/\mathcal{T}_{\text{FedAvg}}$ in 200 iterations of training, where $\mathcal{T}_{\text{CacheFL-OPT}}$ and $\mathcal{T}_{\text{FedAvg}}$ are the per-iteration training time of CacheFL-OPT and FedAvg (no caching). CacheFL-OPT can reduce the per-iteration delay by 28% on average and by 45% (nearly reducing it by half) in the best iteration.

Convergence in iterations and wall-clock time. We compare different algorithms in training logistic regression in Fig. 4. The first row and the second row compare the convergence in iterations and wall-clock time, respectively. For all datasets, CacheFL-OPT has a convergence rate close to those of FedAvg and FedProx, even though clients in $\mathcal{K}_{\text{cache}}$ make local updates based on cached global models, meaning that the penalty for caching in terms of the number of iterations for convergence is not significant and the saving in the time per iteration will be the deciding factor. Thus, CacheFL-OPT has the fastest convergence in wall-clock time due to its ability to reduce $\mathcal{T}$. This advantage of CacheFL-OPT is more significant in harder learning problems that need more iterations to converge (e.g. Fig 4(d) and 4(e)). Moreover, though FedAvg-stragglers reduces $\mathcal{T}$ by excluding stragglers, its training loss is much larger than others, especially on non-i.i.d. local datasets, due to having fewer samples for training. Lastly, CacheFL-random fails to improve the training time over FedAvg and FedProx, with no optimization in the client partition. The randomly decided client partition cannot efficiently reduce $\mathcal{T}$ while possibly reducing the convergence rate by letting too many clients use outdated global models. This emphasizes the need of optimizing client partition (as in Sec. 5).



(a) MLP on non-i.i.d. MNIST (b) CNN on non-i.i.d. FMNIST

Figure 5: Convergence comparisons in wall-clock time, when training MLP and CNN, respectively.

In Fig. 5, we present the results of training an MLP on non-i.i.d. MNIST and training a CNN on non-i.i.d. FMNIST, respectively. CacheFL-OPT still has the fastest convergence in wall-clock time when solving non-convex problems, especially for CNN on FMNIST, while CacheFL-random performs badly, further emphasizing the need of optimizing client partition (as discussed in Sec. 5).

[4] We consider full client participation in the experiments. Additional experiments with partial client participation and with a larger number of clients are included in Appendix C.

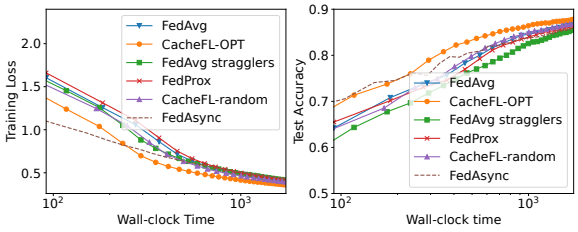


Figure 6: Comparison with asynchronous FL, with logistic regression on non-i.i.d. MNIST.

Comparison with asynchronous FL. Asynchronism is another mechanism to mitigate the straggler issue. In Fig. 6, we compare CacheFL-OPT with FedAsync [35], the asynchronous implementation of FedAvg where clients aggregate their local models asynchronously to the global model with weights being dependent on the staleness. FedAsync proceeds faster in the beginning, as fast clients finish more local updates and model aggregation because of asynchronism. However, FedAsync becomes slower than CacheFL-OPT later due to larger staleness in clients' local models.

7 CONCLUSION

In this paper, we design cache-enabled federated learning systems, which allow clients to reduce per-iteration delay by making local updates based on cached global models. We formulate and solve for caching strategies that minimize the overall wall-clock training time of FedAvg in the proposed systems.

Our current theoretical analysis focuses on convex loss functions. The extension to non-convex loss is a future direction. Moreover, the implementation of other algorithms in the proposed systems and the combination with other efficiency-improving techniques (e.g., compression) are also possible future directions.

ACKNOWLEDGMENTS

The authors gratefully acknowledge support from NSF (grants 2107062 and 2106891) and ARO grant W911NF-23-2-0014.

REFERENCES

[1] Duane Buck and Mukesh Singhal. 1996. An analytic study of caching in computer systems. *J. Parallel and Distrib. Comput.* 32, 2 (1996), 205–214.
[2] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. 2021. Communication-efficient federated learning with adaptive parameter freezing. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1–11.
[3] Mingzhe Chen, H Vincent Poor, Walid Saad, and Shuguang Cui. 2020. Convergence time optimization for federated learning over wireless networks. *IEEE Transactions on Wireless Communications* 20, 4 (2020), 2457–2471.
[4] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H Vincent Poor, and Shuguang Cui. 2020. A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications* 20, 1 (2020), 269–283.
[5] Alp Emre Durmus, Zhao Yue, Matas Ramon, Mattina Matthew, Whatmough Paul, and Saligrama Venkatesh. 2021. Federated Learning Based on Dynamic Regularization. In *International Conference on Learning Representations*.
[6] Fangcheng Fu, Xupeng Miao, Jiawei Jiang, Huanran Xue, and Bin Cui. 2022. Towards communication-efficient vertical federated learning training via cache-enabled local updates. *arXiv preprint arXiv:2207.14628* (2022).
[7] Google. 2022. How Messages improves suggestions with federated technology. https://support.google.com/messages/answer/9327902 Last accessed 22 July 2022.
[8] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. 2021. Local sgd: Unified theory and new efficient methods. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3556–3564.
[9] Xinran Gu, Kaixuan Huang, Jingzhao Zhang, and Longbo Huang. 2021. Fast federated learning in the presence of arbitrary device unavailability. *Advances in Neural Information Processing Systems* 34 (2021), 12052–12064.
[10] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. 2021. Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2350–2358.
[11] Kais Hamza, Peter Jagers, Aidan Sudbury, and Daniel Tokarev. 2009. The mixing advantage is less than 2. *Extremes* 12, 1 (2009), 19–31.
[12] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).
[13] Gautam Kamath. 2015. Bounds on the expectation of the maximum of samples from a gaussian. (2015). www.gautamkamath.com/writings/gaussian_max.pdf
[14] Mehmet Karaca, Tansu Alpcan, and Ozgur Ercetin. 2019. Smart Scheduling and Feedback Allocation over Non-stationary Wireless Channels. *arXiv preprint arXiv:1911.03632* (2019).
[15] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*. PMLR, 5132–5143.
[16] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).
[17] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* (1998), 2278–2324.
[19] Dongsheng Li, Yuxi Zhao, and Xiaowen Gong. 2021. Quality-Aware distributed computation and communication scheduling for fast convergent wireless federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 1–8.
[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.
[21] Yuezhou Liu, Yuanyuan Li, Lili Su, Edmund Yeh, and Stratis Ioannidis. 2022. Experimental design networks: A paradigm for serving heterogeneous learners under networking constraints. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 210–219.
[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR.
[23] Mobiperf. [n. d.]. https://www.measurementlab.net/tests/mobiperf/.
[24] Georgios Paschos, George Iosifidis, Giuseppe Caire, et al. 2020. Cache optimization models and algorithms. *Foundations and Trends® in Communications and Information Theory* 16, 3–4 (2020), 156–345.
[25] Dario Rossi and Giuseppe Rossini. 2011. Caching performance of content centric networks under multi-path routing (and more). *Relatório técnico, Telecom ParisTech* 2011 (2011), 1–6.
[26] Yichen Ruan, Xiaoxi Zhang, and Carlee Joe-Wong. 2021. How valuable is your data? optimizing client recruitment in federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 1–8.
[27] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. 2021. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3403–3411.
[28] Zai Shi and Atilla Eryilmaz. 2021. Communication-efficient Subspace Methods for High-dimensional Federated Learning. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 543–550.
[29] Sebastian U Stich and Sai Praneeth Karimireddy. 2020. The error-feedback framework: Better rates for SGD with delayed gradients and compressed updates. *Journal of Machine Learning Research* 21 (2020), 1–36.
[30] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. 2021. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917* (2021).
[31] Su Wang, Yichen Ruan, Yuwei Tu, Satyavrat Wagle, Christopher G Brinton, and Carlee Joe-Wong. 2021. Network-aware optimization of distributed learning for fog computing. *IEEE/ACM Transactions on Networking* 29, 5 (2021), 2019–2032.
[32] Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. 2020. Is local SGD better than minibatch SGD?. In *International Conference on Machine Learning*. PMLR.
[33] Yuting Wu, Yanxiang Jiang, Mehdi Bennis, Fuchun Zheng, Xiqi Gao, and Xiaohu You. 2020. Content popularity prediction in fog radio access networks: A federated learning based approach. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[34] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
[35] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).
[36] Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. 2021. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269* (2021).
[37] Hang Xu, Chen-Yu Ho, Ahmed M Abdelmoniem, Aritra Dutta, El Houcine Bergou, Konstantinos Karatsenidis, Marco Canini, and Panos Kalnis. 2020. *Compressed communication for distributed deep learning: Survey and quantitative evaluation.* Technical Report.
[38] Haibo Yang, Jia Liu, and Elizabeth S Bentley. 2021. CFedAvg: achieving efficient communication and fast convergence in non-iid federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt).* IEEE, 1–8.
[39] Shuai Yu, Xu Chen, Zhi Zhou, Xiaowen Gong, and Di Wu. 2020. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. *IEEE Internet of Things Journal* 8, 4 (2020), 2238–2251.
[40] Zhengxin Yu, Jia Hu, Geyong Min, Zi Wang, Wang Miao, and Shancang Li. 2021. Privacy-preserving federated deep learning for cooperative hierarchical caching in fog computing. *IEEE Internet of Things Journal* (2021).
[41] Zhengxin Yu, Jia Hu, Geyong Min, Zhiwei Zhao, Wang Miao, and M Shamim Hossain. 2020. Mobility-aware proactive edge caching for connected vehicles using federated learning. *IEEE Transactions on Intelligent Transportation Systems* 22, 8 (2020), 5341–5351.
[42] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20).* 493–506.
[43] Yupeng Zhang, Lingjie Duan, and Ngai-Man Cheung. 2022. Accelerating Federated learning on non-IID data against stragglers. In *2022 IEEE International Conference on Sensing, Communication, and Networking (SECON Workshops).*
[44] Xin-Ying Zheng, Ming-Chun Lee, and Y-W Peter Hong. 2021. Knowledge Caching for Federated Learning. In *2021 IEEE Global Communications Conference (GLOBECOM).* IEEE, 1–6.

A FULL CLIENT PARTICIPATION

This section includes the proofs of two lemmas for Theorem 1, which gives the convergence bound of CacheFL when we consider full client participation. The proofs are based on assumptions 1-5.

A.1 Proof of Lemma 1

By (9) and (16), we have $\overline{\mathbf{w}}^{(t,\tau+1)} = \overline{\mathbf{w}}^{(t,\tau)} - \eta \sum_{k\in\mathcal{K}} d_k \mathbf{g}_k^{(t,\tau)}$, and by parallelogram identity,

$$\sum_{k\in\mathcal{K}} d_k \langle \mathbf{g}_k^{(t,\tau)}, \overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^* \rangle = \frac{1}{2\eta}(||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}^*||^2 - ||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 - ||\overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^*||^2). \tag{28}$$

By convexity and L-smoothness of F_k, we have

$$\begin{aligned}
&F_k(\overline{\mathbf{w}}^{(t,\tau+1)}) \\
&\le F_k(\mathbf{w}_k^{(t,\tau)}) + \langle \nabla F_k(\mathbf{w}_k^{(t,\tau)}), \overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}_k^{(t,\tau)} \rangle \\
&\quad + \frac{L}{2}||\overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}_k^{(t,\tau)}||^2 \quad \text{(smoothness)} \\
&\le F_k(\mathbf{w}^*) + \langle \nabla F_k(\mathbf{w}_k^{(t,\tau)}), \overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^* \rangle \\
&\quad + \frac{L}{2}||\overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}_k^{(t,\tau)}||^2 \quad \text{(convexity)} \\
&\le F_k(\mathbf{w}^*) + \langle \nabla F_k(\mathbf{w}_k^{(t,\tau)}), \overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^* \rangle \\
&\quad + L||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 + L||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2,
\end{aligned} \tag{29}$$

where the last step is due to triangle inequality. Combining (28) and (29) yields

$$\begin{aligned}
&F(\overline{\mathbf{w}}^{(t,\tau+1)}) - F(\mathbf{w}^*) = \sum_{k\in\mathcal{K}} d_k (F_k(\overline{\mathbf{w}}^{(t,\tau+1)}) - F_k(\mathbf{w}^*)) \\
&\le \sum_{k\in\mathcal{K}} d_k \langle \nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \mathbf{g}_k^{(t,\tau)}, \overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^* \rangle \\
&\quad + L||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 + L \sum_{k\in\mathcal{K}} d_k ||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 \\
&\quad + \frac{1}{2\eta}(||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}^*||^2 - ||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 - ||\overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^*||^2).
\end{aligned} \tag{30}$$

Since $\mathbb{E}[\nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \mathbf{g}_k^{(t,\tau)} | \mathcal{F}^{(t,\tau)}] = 0$, we have

$$\begin{aligned}
&\mathbb{E}[\sum_{k\in\mathcal{K}} d_k \langle \nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \mathbf{g}_k^{(t,\tau)}, \overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^* \rangle | \mathcal{F}^{(t,\tau)}] \\
&= \mathbb{E}[\sum_{k\in\mathcal{K}} d_k \langle \nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \mathbf{g}_k^{(t,\tau)}, \overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)} \rangle | \mathcal{F}^{(t,\tau)}] \\
&\le \eta \mathbb{E}[|| \sum_{k\in\mathcal{K}} d_k (\nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \mathbf{g}_k^{(t,\tau)}) ||^2 | \mathcal{F}^{(t,\tau)}] \\
&\quad + \frac{1}{4\eta} \mathbb{E}[||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 | \mathcal{F}^{(t,\tau)}] \\
&\le \eta\sigma^2 \sum_{k\in\mathcal{K}} d_k^2 + \frac{1}{4\eta} \mathbb{E}[||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 | \mathcal{F}^{(t,\tau)}],
\end{aligned} \tag{31}$$

where the first inequality is by Young's inequality and the last one is by the bounded covariance assumption and independence across clients. Plugging (31) back to the conditional expectation of (30) yields

$$\begin{aligned}
&\mathbb{E}[F(\overline{\mathbf{w}}^{(t,\tau+1)}) - F(\mathbf{w}^*) | \mathcal{F}^{(t,\tau)}] \\
&\quad + \frac{1}{2\eta}(\mathbb{E}[||\overline{\mathbf{w}}^{(t,\tau+1)} - \mathbf{w}^*||^2 | \mathcal{F}^{(t,\tau)}] - ||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}^*||^2) \\
&\le \eta\sigma^2 \sum_{k\in\mathcal{K}} d_k^2 - (\frac{1}{4\eta} - L) \mathbb{E}[||\overline{\mathbf{w}}^{(t,\tau+1)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 | \mathcal{F}^{(t,\tau)}] \\
&\quad + L \sum_{k\in\mathcal{K}} d_k ||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 \\
&\overset{\eta \le \frac{1}{4L}}{\le} \eta\sigma^2 \sum_{k\in\mathcal{K}} d_k^2 + L \sum_{k\in\mathcal{K}} d_k ||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2
\end{aligned}$$

By the law of total expectation, telescoping τ from 0 to $\tau_{\max} - 1$ yields

$$\begin{aligned}
&\mathbb{E}[\frac{1}{\tau_{\max}} \sum_{\tau=1}^{\tau_{\max}} F(\overline{\mathbf{w}}^{(t,\tau)}) - F(\mathbf{w}^*) | \mathcal{F}^{(t,0)}] \\
&\le \frac{1}{2\eta\tau_{\max}}(||\overline{\mathbf{w}}^{(t,0)} - \mathbf{w}^*||^2 - \mathbb{E}[||\overline{\mathbf{w}}^{(t,\tau_{\max})} - \mathbf{w}^*||^2 | \mathcal{F}^{(t,0)}]) \\
&+ \eta\sigma^2 \sum_{k\in\mathcal{K}} d_k^2 + \frac{L}{\tau_{\max}} \sum_{\tau=0}^{\tau_{\max}-1} \sum_{k\in\mathcal{K}} d_k \mathbb{E}[||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 | \mathcal{F}^{(t,0)}].
\end{aligned}$$

A.2 Proof of Lemma 2

Let $\overline{\mathbf{g}}^{(t,\tau)} = \sum_{k\in\mathcal{K}} d_k \mathbf{g}_k^{(t,\tau)}$, we have

$$\begin{aligned}
V^{(t,\tau+1)} &= \sum_{k\in\mathcal{K}} d_k ||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)} - \eta\mathbf{g}_k^{(t,\tau)} + \eta\overline{\mathbf{g}}^{(t,\tau)}||^2 \\
&= V^{(t,\tau)} + 2\eta \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)} \rangle \\
&\quad + \eta^2 \sum_{k\in\mathcal{K}} d_k ||\mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)}||^2 \\
&= V^{(t,\tau)} + 2\eta \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \mathbf{g}_k^{(t,\tau)} \rangle \\
&\quad + \eta^2 \sum_{k\in\mathcal{K}} d_k ||\mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)}||^2
\end{aligned}$$

Take the conditional expectation on both sides,

$$\begin{aligned}
\mathbb{E}[V^{(t,\tau+1)}|\mathcal{F}^{(t,\tau)}] \leq V^{(t,\tau)} + \eta^2 \sum_{k\in\mathcal{K}} d_k \mathbb{E}[||\mathbf{g}_k^{(t,\tau)}||^2 |\mathcal{F}^{(t,\tau)}] \\
+ 2\eta \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) \rangle, \qquad (32)
\end{aligned}$$

where the inequality is derived using variance decomposition (the variance of X equal to the expectation of X squared minus the expected value of X squared), i.e.,

$$\begin{aligned}
\sum_{k\in\mathcal{K}} d_k ||\mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)}||^2 &= \sum_{k\in\mathcal{K}} d_k ||\mathbf{g}_k^{(t,\tau)}||^2 - ||\overline{\mathbf{g}}^{(t,\tau)}||^2 \\
&\leq \sum_{k\in\mathcal{K}} d_k ||\mathbf{g}_k^{(t,\tau)}||^2
\end{aligned}$$

The last term in (32) can be further bounded as

$$\begin{aligned}
&\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) \rangle \\
&= \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) \rangle \\
&\quad + \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) \rangle
\end{aligned}$$

where the first term is non-positive by convexity, and the second term equals to $\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) - \nabla F(\overline{\mathbf{w}}^{(t,\tau)}) \rangle$ as $\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F(\overline{\mathbf{w}}^{(t,\tau)}) \rangle = 0$. Thus, we have

$$\begin{aligned}
&\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) \rangle \\
&\leq \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) - \nabla F(\overline{\mathbf{w}}^{(t,\tau)}) \rangle \\
&\leq \sum_{k\in\mathcal{K}} d_k \Big(\frac{1}{2\eta\tau_{\max}} ||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}||^2 \\
&\quad + \frac{\eta\tau_{\max}}{2} ||\nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) - \nabla F(\overline{\mathbf{w}}^{(t,\tau)})||^2 \Big) \\
&\leq \frac{1}{2\eta\tau_{\max}} V^{(t,\tau)} + \frac{\eta\tau_{\max}}{2}\zeta^2
\end{aligned}$$

where the second inequality is by Young's inequality, and the last one is by (15). Plugging the above equation to (32) yields

$$\begin{aligned}
\mathbb{E}[V^{(t,\tau+1)}|\mathcal{F}^{(t,\tau)}] \leq &(1 + \frac{1}{\tau_{\max}}) V^{(t,\tau)} + \eta^2 \tau_{\max}\zeta^2 \\
&+ \eta^2 \sum_{k\in\mathcal{K}} d_k \mathbb{E}[||\mathbf{g}_k^{(t,\tau)}||^2 |\mathcal{F}^{(t,\tau)}].
\end{aligned}$$

Take full expectation on both sides, by (14), we obtain

$$\mathbb{E}[V^{(t,\tau+1)}] \leq (1 + \frac{1}{\tau_{\max}}) \mathbb{E}[V^{(t,\tau)}] + \eta^2 G^2 + \tau_{\max}\eta^2\zeta^2 \qquad (33)$$

By telescoping,

$$\mathbb{E}[V^{(t,\tau)}] \leq (1 + \frac{1}{\tau_{\max}})^\tau \mathbb{E}[V^{(t,0)}] + \frac{(1 + \frac{1}{\tau_{\max}})^\tau - 1}{\frac{1}{\tau_{\max}}} (\eta^2 G^2 + \tau_{\max}\eta^2\zeta^2)$$

Note that $V^{(t,0)}$ may not be zero with the cached model (different from that in classic FedAvg). Define $d_{\text{cache}} = \sum_{k\in\mathcal{K}_{\text{cache}}} d_k$ and $d_{\text{server}} = \sum_{k\in\mathcal{K}_{\text{server}}} d_k$ ($d_{\text{cache}} + d_{\text{server}} = 1$), we have

$$\begin{aligned}
&\mathbb{E}[V^{(t,0)}] \\
=&d_{\text{cache}} \mathbb{E}[||\mathbf{w}^{t-1} - (d_{\text{cache}}\mathbf{w}^{t-1} + d_{\text{server}}\mathbf{w}^t)||^2] \\
&+ d_{\text{server}} \mathbb{E}[||\mathbf{w}^t - (d_{\text{cache}}\mathbf{w}^{t-1} + d_{\text{server}}\mathbf{w}^t)||^2] \\
=&d_{\text{cache}} \mathbb{E}[||d_{\text{server}}(\mathbf{w}^t - \mathbf{w}^{t-1})||^2] + d_{\text{server}} \mathbb{E}[||d_{\text{cache}}(\mathbf{w}^t - \mathbf{w}^{t-1})||^2] \\
=&d_{\text{cache}} d_{\text{server}} \mathbb{E}||\mathbf{w}^t - \mathbf{w}^{t-1}||^2.
\end{aligned}$$

With full client participation, we further bound $\mathbb{E}||\mathbf{w}^t - \mathbf{w}^{t-1}||^2$ as

$$\begin{aligned}
&\mathbb{E}||\mathbf{w}^t - \mathbf{w}^{t-1}||^2 \\
=&\mathbb{E}||\sum_{k\in\mathcal{K}} d_k \mathbf{w}_k^{(t-1,\tau_{\max})} - \mathbf{w}^{t-1}||^2 \\
=&\mathbb{E}||\sum_{k\in\mathcal{K}_{\text{cache}}} d_k (\mathbf{w}^{t-2} - \sum_{\tau=0}^{\tau_{\max}-1} \eta\mathbf{g}_k^{(t-1,\tau)}) \\
&+ \sum_{k\in\mathcal{K}_{\text{server}}} d_k (\mathbf{w}^{t-1} - \sum_{\tau=0}^{\tau_{\max}-1} \eta\mathbf{g}_k^{(t-1,\tau)}) - \mathbf{w}^{t-1}||^2 \\
=&\mathbb{E}||d_{\text{cache}}(\mathbf{w}^{t-2} - \mathbf{w}^{t-1}) - \sum_{k\in\mathcal{K}} d_k \sum_{\tau=0}^{\tau_{\max}-1} \eta\mathbf{g}_k^{(t-1,\tau)}||^2 \\
\leq &\sum_{k\in\mathcal{K}} d_k \mathbb{E}[||\sum_{t'=1}^{t} d_{\text{cache}}^{t-t'} \sum_{\tau=0}^{\tau_{\max}-1} \eta\mathbf{g}_k^{(t'-1,\tau)}||^2 \\
\leq &(\sum_{t'=1}^{t} d_{\text{cache}}^{t-t'})^2 \tau_{\max}^2 \eta^2 G^2 \leq \frac{1}{d_{\text{server}}^2} \tau_{\max}^2 \eta^2 G^2
\end{aligned}$$

where the last inequality is derived by letting $t \to \infty$. Thus,

$$\mathbb{E}[V^{(t,\tau)}] \leq \frac{3d_{\text{cache}}}{d_{\text{server}}} \tau_{\max}^2 \eta^2 G^2 + 2\eta^2 G^2 + 2\tau_{\max}\eta^2\zeta^2 \qquad (34)$$

B PARTIAL CLIENT PARTICIPATION

This section includes the proof for Theorem 2 which gives the convergence bound of CacheFL with partial client participation, as well as the proofs for two necessary lemmas. The proofs are based on assumptions 2-5, while assumption 1 is replaced by the following client sampling assumption: We consider that in each iteration t, we uniformly sample S clients from a total of K clients for participation.

We denote the set of clients that participate in t-th iteration as $\mathcal{S}^{(t)}$. For $k \in \mathcal{S}^{(t)}$, the local update is given by $\mathbf{w}_k^{(t,\tau+1)} = \mathbf{w}_k^{(t,\tau+1)} - \mu \frac{K}{S} \mathbf{g}_k^{(t,\tau)}$, such that $\mathbb{E}_{\mathcal{S}^{(t)}} \frac{1}{K} \sum_{k\in\mathcal{S}^{(t)}} \frac{K}{S} \mathbf{g}_k^{(t,\tau)} = \frac{1}{K} \sum_{k\in\mathcal{K}} \mathbf{g}_k^{(t,\tau)}$. For other unsampled clients, we have $\mathbf{w}_k^{(t,\tau+1)} = \mathbf{w}_k^{(t,\tau+1)}$. We consider balanced local data, where $d_k = \frac{1}{K}$ for all $k \in \mathcal{K}$. Other parts of the algorithm remain unchanged.

We define virtual sequence $\overline{\mathbf{w}}^{(t,\tau)} = \frac{1}{K} \sum_{k\in\mathcal{K}} \mathbf{w}_k^{(t,\tau)}$ and we have $\overline{\mathbf{w}}^{(t,\tau+1)} = \overline{\mathbf{w}}^{(t,\tau)} - \mu \sum_{k\in\mathcal{S}^{(t)}} \frac{1}{S} \mathbf{g}_k^{(t,\tau)}$. Let the global model at iteration t be $\mathbf{w}^t = \overline{\mathbf{w}}^{(t-1,\tau_{\max})}$. At the start iteration, we initialize $\mathbf{w}^{(t,0)} = \mathbf{w}^{t-1}$ for $k \in \mathcal{K}_{\text{cache}}^{(t)}$ and $\mathbf{w}^{(t,0)} = \mathbf{w}^{t-1}$ for other k. Thus, we have $\overline{\mathbf{w}^{(t,0)}} = \frac{K_{\text{cache}}^{(t)}}{K} \mathbf{w}^{t-1} + \frac{K - K_{\text{cache}}^{(t)}}{K} \mathbf{w}^t$.

Lemma 4. *For partial client participation, if the client learning rate satisfies $\eta \le \frac{1}{4L}$, then*

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} F(\overline{\mathbf{w}}^{(t,\tau)}) - F(\mathbf{w}^*) | \mathcal{F}^{(t,0)}\right] \\
\le &\frac{1}{2\eta\tau_{max}} \left(W^{(t,0)} - \mathbb{E}\left[W^{(t,\tau_{max})} | \mathcal{F}^{(t,0)}\right]\right) + \eta \frac{\sigma^2}{K} \\
&+ \frac{L}{\tau_{max}} \sum_{\tau=0}^{\tau_{max}-1} \frac{1}{K} \sum_{k\in\mathcal{K}} \mathbb{E}\left[||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2 | \mathcal{F}^{(t,0)}\right]
\end{aligned}$$

where $W^{(t,\tau)} := ||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}^*||^2$ and $\mathcal{F}^{(t,0)}$ is the σ-field representing all historical information up to the start of iteration t.

Proof. The proof follows directly from the proof of Lemma 1, noticing that $\mathbb{E}_{\mathcal{S}^{(t)}} \frac{1}{K} \sum_{k\in\mathcal{S}^{(t)}} \frac{K}{S} \mathbf{g}_k^{(t,\tau)} = \frac{1}{K} \sum_{k\in\mathcal{K}} \mathbf{g}_k^{(t,\tau)}$. □

Lemma 5. *For partial client participation, let $V^{(t,\tau)} = \frac{1}{K} \sum_{k\in\mathcal{K}} ||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)}||^2$, we have*

$$\mathbb{E}\left[V^{(t,\tau)}\right] \le \tau_{max}^2 \eta^2 G^2 + 2\frac{K}{S}\eta^2 G^2 + 2\tau_{max}\eta^2\zeta^2.$$

Proof. Let $\overline{\mathbf{g}}^{(t,\tau)} = \sum_{k\in\mathcal{S}^{(t)}} \frac{1}{S} \mathbf{g}_k^{(t,\tau)}$ and $\mathbf{g}_k^{(t,\tau)} = 0$, for $k \notin \mathcal{S}^{(t)}$, we have

$$\begin{aligned}
V^{(t,\tau+1)} &= \sum_{k\in\mathcal{K}} \frac{1}{K} ||\mathbf{w}_k^{(t,\tau)} - \overline{\mathbf{w}}^{(t,\tau)} - \eta \frac{K}{S} \mathbf{g}_k^{(t,\tau)} + \eta \overline{\mathbf{g}}^{(t,\tau)}||^2 \\
&= V^{(t,\tau)} + 2\eta \sum_{k\in\mathcal{K}} \frac{1}{K} \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \frac{K}{S} \mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)} \rangle \\
&\quad + \eta^2 \sum_{k\in\mathcal{K}} \frac{1}{K} ||\frac{K}{S} \mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)}||^2 \\
&= V^{(t,\tau)} + 2\eta \sum_{k\in\mathcal{K}} \frac{1}{K} \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \frac{K}{S} \mathbf{g}_k^{(t,\tau)} \rangle \\
&\quad + \eta^2 \sum_{k\in\mathcal{K}} \frac{1}{K} ||\frac{K}{S} \mathbf{g}_k^{(t,\tau)} - \overline{\mathbf{g}}^{(t,\tau)}||^2 \\
&\le V^{(t,\tau)} + 2\eta \sum_{k\in\mathcal{K}} \frac{1}{K} \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \frac{K}{S} \mathbf{g}_k^{(t,\tau)} \rangle \\
&\quad + \eta^2 \sum_{k\in\mathcal{S}^{(t)}} \frac{K}{S^2} ||\mathbf{g}_k^{(t,\tau)}||^2,
\end{aligned}$$

where the inequality is derived using variance decomposition. Take the conditional expectation on both sides and consider $\mathbf{g}_k^{(t,\tau)}$ with its original definition for all k,

$$\begin{aligned}
\mathbb{E}[V^{(t,\tau+1)} | \mathcal{F}^{(t,\tau)}] \le &V^{(t,\tau)} + \eta^2 \frac{K}{S} G^2 \\
&+ 2\eta \sum_{k\in\mathcal{K}} \frac{1}{K} \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) \rangle, \qquad (35)
\end{aligned}$$

The last term in (35) can be further bounded as

$$\begin{aligned}
&\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) \rangle \\
= &\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) - \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) \rangle \\
&+ \sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) \rangle
\end{aligned}$$

where the first term is non-positive by convexity, and the second term equals to $\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) - \nabla F(\overline{\mathbf{w}}^{(t,\tau)}) \rangle$ as $\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F(\overline{\mathbf{w}}^{(t,\tau)}) \rangle = 0$. Thus, we have

$$\begin{aligned}
&\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\mathbf{w}_k^{(t,\tau)}) \rangle \\
\le &\sum_{k\in\mathcal{K}} d_k \langle \overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}, \nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) - \nabla F(\overline{\mathbf{w}}^{(t,\tau)}) \rangle \\
\le &\sum_{k\in\mathcal{K}} d_k \Big(\frac{1}{2\eta\tau_{\max}} ||\overline{\mathbf{w}}^{(t,\tau)} - \mathbf{w}_k^{(t,\tau)}||^2 \\
&+ \frac{\eta\tau_{\max}}{2} ||\nabla F_k(\overline{\mathbf{w}}^{(t,\tau)}) - \nabla F(\overline{\mathbf{w}}^{(t,\tau)})||^2\Big) \\
\le &\frac{1}{2\eta\tau_{\max}} V^{(t,\tau)} + \frac{\eta\tau_{\max}}{2} \zeta^2
\end{aligned}$$

where the second inequality is by Young's inequality, and the last one is by (15). Plugging the above equation to (35) yields

$$\mathbb{E}[V^{(t,\tau+1)} | \mathcal{F}^{(t,\tau)}] \le (1 + \frac{1}{\tau_{\max}}) V^{(t,\tau)} + \eta^2 \tau_{\max} \zeta^2 + \eta^2 \frac{K}{S} G^2.$$

Take full expectation on both sides, by (14), we obtain

$$\mathbb{E}[V^{(t,\tau+1)}] \le (1 + \frac{1}{\tau_{\max}}) \mathbb{E}[V^{(t,\tau)}] + \eta^2 \frac{K}{S} G^2 + \tau_{\max} \eta^2 \zeta^2 \qquad (36)$$

By telescoping,

$$\mathbb{E}[V^{(t,\tau)}] \le (1 + \frac{1}{\tau_{\max}})^\tau \mathbb{E}[V^{(t,0)}] + \frac{(1 + \frac{1}{\tau_{\max}})^\tau - 1}{\frac{1}{\tau_{\max}}} (\eta^2 \frac{K}{S} G^2 + \tau_{\max} \eta^2 \zeta^2)$$

Note that $V^{(t,0)}$ may not be zero with the cached model (different from that in classic FedAvg). Define $d_{\text{cache}}^{(t)} = \sum_{k\in\mathcal{K}_{\text{cache}}\cap\mathcal{S}^{(t)}} \frac{1}{K}$ and $\overline{d}_{\text{cache}} = \mathbb{E}[d_{\text{cache}}^{(t)}] = \frac{K_{\text{cache}} S}{K^2}$, where $|\mathcal{K}_{\text{cache}}| = K_{\text{cache}}$, we have

$$\begin{aligned}
\mathbb{E}[V^{(t,0)}] = &\mathbb{E}[d_{\text{cache}}^{(t)} ||\mathbf{w}^{t-1} - (d_{\text{cache}}^{(t)} \mathbf{w}^{t-1} + (1 - d_{\text{cache}}^{(t)}) \mathbf{w}^t)||^2 \\
&+ (1 - d_{\text{cache}}^{(t)}) ||\mathbf{w}^t - (d_{\text{cache}}^{(t)} \mathbf{w}^{t-1} + (1 - d_{\text{cache}}^{(t)}) \mathbf{w}^t)||^2] \\
\le &\mathbb{E}[d_{\text{cache}}^{(t)} (1 - d_{\text{cache}}^{(t)}) ||\mathbf{w}^t - \mathbf{w}^{t-1}||^2] \\
\le &\overline{d}_{\text{cache}} (1 - \overline{d}_{\text{cache}}) \mathbb{E}[||\mathbf{w}^t - \mathbf{w}^{t-1}||^2]
\end{aligned}$$

With partial client participation:

$$
\begin{aligned}
&||\mathbf{w}^t - \mathbf{w}^{t-1}||^2 \\
=&||d_{\text{cache}}^{(t)}(\mathbf{w}^{t-2} - \mathbf{w}^t - 1) - \sum_{k\in\mathcal{S}^{(t-1)}} \frac{\eta}{S} \sum_{\tau=0}^{\tau_{\max}-1} \mathbf{g}_k^{(t-1,\tau)}||^2 \\
=&||\sum_{t'=1}^{t}(\prod_{t''=1}^{t-t'} d_{\text{cache}}^{(t-t'')}) \sum_{k\in\mathcal{S}^{(t'-1)}} \frac{\eta}{S} \sum_{\tau=0}^{\tau_{\max}-1} \mathbf{g}_k^{(t'-1,\tau)}||^2 \\
\leq&(\sum_{t'=1}^{t}\prod_{t''=1}^{t-t'} d_{\text{cache}}^{(t-t'')})^2 \tau_{\max}^2\eta^2G^2
\end{aligned}
$$

where we let $\prod_{t''=1}^{0} d_{\text{cache}}^{(t-t'')} = 1$. Taking expectation on both sides,

$$
\begin{aligned}
&\mathbb{E}||\mathbf{w}^t - \mathbf{w}^{t-1}||^2 \\
\leq&\mathbb{E}[(\sum_{t'=1}^{t}\prod_{t''=1}^{t-t'} d_{\text{cache}}^{(t-t'')})^2]\tau_{\max}^2\eta^2G^2 \\
\leq&(\sum_{t'=1}^{t}(\bar{d}_{\text{cache}})^{t-t'})(\sum_{t'=1}^{t}(\frac{S}{K})^{t-t'})\tau_{\max}^2\eta^2G^2 \\
\leq&\frac{1}{1-\bar{d}_{\text{cache}}}\frac{K}{K-S}\tau_{\max}^2\eta^2G^2
\end{aligned}
$$

Thus, we have

$$
\mathbb{E}[V^{(t,\tau)}] \leq 3\frac{K_{\text{cache}}S}{(K-S)K}\tau_{\max}^2\eta^2G^2 + 2\eta^2\frac{K}{S}G^2 + 2\tau_{\max}\eta^2\zeta^2 \quad (37)
$$

□

Combining Lemma 4 and Lemma 5, we have

$$
\begin{aligned}
&\mathbb{E}\left[\frac{1}{\tau_{\max}}\sum_{\tau=1}^{\tau_{\max}} F(\overline{\mathbf{w}}^{(t,\tau)}) - F(\mathbf{w}^*)|\mathcal{F}^{(t,0)}\right] \\
\leq&\frac{1}{2\eta\tau_{\max}}\left(W^{(t,0)} - \mathbb{E}\left[W^{(t,\tau_{\max})}|\mathcal{F}^{(t,0)}\right]\right) + \eta\frac{\sigma^2}{K} \\
&+ 2\tau_{\max}\eta^2L\zeta^2 + 2\eta^2\frac{K}{S}LG^2 + 3\frac{S}{K-S}\tau_{\max}^2\eta^2LG^2. \quad (38)
\end{aligned}
$$

By convexity, $\mathbb{E}||\overline{\mathbf{w}}^{(t,0)} - \mathbf{w}^*||^2 = \mathbb{E}||d_{\text{cache}}^{(t)}(\overline{\mathbf{w}}^{(t-2,\tau_{\max})} - \mathbf{w}^*) + (1 - d_{\text{cache}}^{(t)})(\overline{\mathbf{w}}^{(t-1,\tau_{\max})} - \mathbf{w}^*)||^2 \leq \frac{K_{\text{cache}}S}{K^2}||\overline{\mathbf{w}}^{(t-2,\tau_{\max})} - \mathbf{w}^*||^2 + (1-\frac{K_{\text{cache}}S}{K^2})||\overline{\mathbf{w}}^{(t-1,\tau_{\max})} - \mathbf{w}^*||^2$, for $t \geq 2$. We also have $\mathbb{E}||\overline{\mathbf{w}}^{(1,0)} - \mathbf{w}^*||^2 = \mathbb{E}||d_{\text{cache}}^{(1)}(\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*) + (1 - d_{\text{server}}^{(1)})(\overline{\mathbf{w}}^{(0,\tau_{\max})} - \mathbf{w}^*)||^2 \leq \frac{K_{\text{cache}}S}{K^2}||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||^2 + (1 - \frac{K_{\text{cache}}S}{K^2})||\overline{\mathbf{w}}^{(0,\tau_{\max})} - \mathbf{w}^*||^2$. Thus, by telescoping, we have

$$
\sum_{t=0}^{T-1}(W^{(t,0)} - \mathbb{E}[W^{(t,\tau_{\max})}]) \leq (1 + \frac{K_{\text{cache}}S}{K^2})||\overline{\mathbf{w}}^{(0,0)} - \mathbf{w}^*||^2.
$$

By above, telescoping (38) with t from 0 to $T-1$ and using the convexity of F, finishes the proof of Theorem 2.

C ADDITIONAL EXPERIMENTAL RESULTS

In this appendix, we present additional experiment results. The main experiment settings are the same as those in Sec. 6.

Effect of different client partitions. In Fig. 7, we further evaluate the effect of client partition on the convergence rate of CacheFL. We consider $\mathcal{K}_{\text{cache}}$ of different sizes and for each size we include the clients with highest per-iteration delay $\mathcal{T}_k$ in $\mathcal{K}_{\text{cache}}$. When $|\mathcal{K}_{\text{cache}}| = 0$, CacheFL reduces to FedAvg. We see that CacheFL has a similar convergence rate in practice when $|\mathcal{K}_{\text{cache}}| = 0, 10, 20, 30,$ and slower convergence rate when $|\mathcal{K}_{\text{cache}}| = 40, 50$, i.e., the majority of the clients use the cache, showing that the effect of using cached models in convergence rate is not significant and the benefit

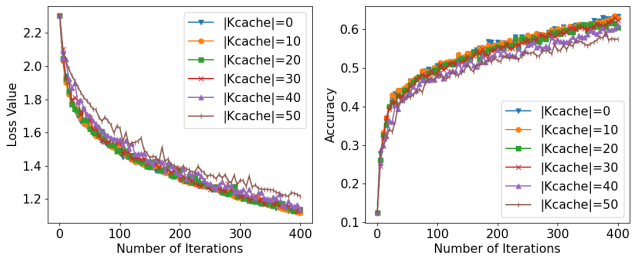


Figure 7: Effect of different number of clients using the cache in convergence rate of CacheFL, with logistic regression on non-i.i.d. CIFAR-10 dataset.

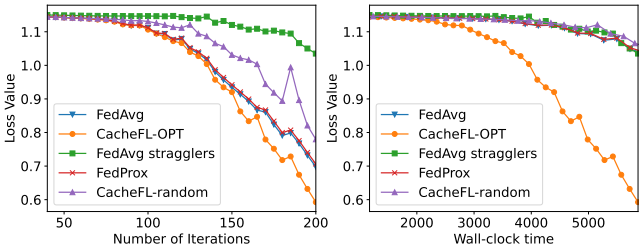


Figure 8: Convergence in loss value with client sampling (partial client participation), with CNN on non-i.i.d. MNIST

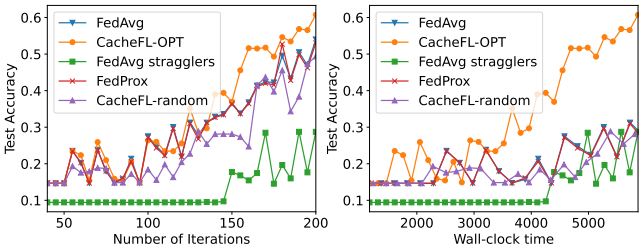


Figure 9: Convergence in accuracy with client sampling (partial client participation), with CNN on non-i.i.d. MNIST

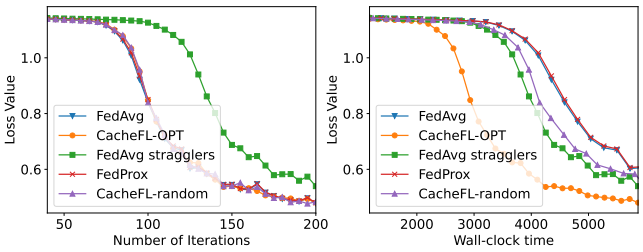


Figure 10: Convergence in loss value with client sampling (partial client participation), with CNN on non-i.i.d. F-MNIST

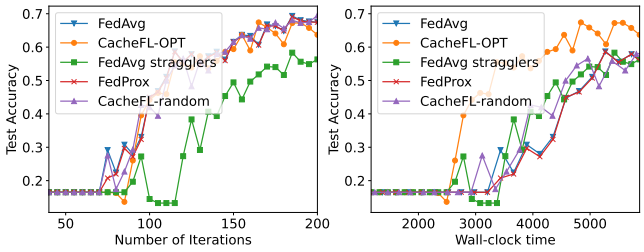


Figure 11: Convergence in accuracy with client sampling (partial client participation), with CNN on non-i.i.d. F-MNIST

in reducing the per-iteration delay plays a more important role if we let $\mathcal{K}_{\text{cache}}$ have a reasonable size.

Experiments with partial client participation. For every compared algorithm, we further consider that in each iteration, 30 out of 50 clients are sampled uniformly at random to participate in the training, i.e., partial client participation. Fig. 8 and 9 compare the convergence speed of the algorithms in training loss and testing accuracy when training CNNs with MNIST data, while Fig. 10 and 11 present the results when training CNNs with FMNIST data. In these experiments, CacheFL-OPT still converges the fastest in wall-clock time. In Fig. 9, we can see that at the wall-clock time when CacheFL-OPT achieves 60% testing accuracy, all other baseline algorithms get accuracies about or below 30%.