No One Idles: Efficient Heterogeneous Federated Learning with Parallel Edge and Server Computation

Feilong Zhang [1] **Xianming Liu** [1 2] **Shiyi Lin** [1] **Gang Wu** [1] **Xiong Zhou** [1] **Junjun Jiang** [1 2] **Xiangyang Ji** [3]

Abstract

Federated learning suffers from a latency bottleneck induced by network stragglers, which hampers the training efficiency significantly. In addition, due to the heterogeneous data distribution and security requirements, simple and fast averaging aggregation is not feasible anymore. Instead, complicated aggregation operations, such as knowledge distillation, are required. The time cost for complicated aggregation becomes a new bottleneck that limits the computational efficiency of FL. In this work, we claim that the root cause of training latency actually lies in the aggregation-then-broadcasting workflow of the server. By swapping the computational order of aggregation and broadcasting, we propose a novel and efficient parallel federated learning (PFL) framework that unlocks the edge nodes during global computation and the central server during local computation. This fully asynchronous and parallel pipeline enables handling complex aggregation and network stragglers, allowing flexible device participation as well as achieving scalability in computation. We theoretically prove that synchronous and asynchronous PFL can achieve a similar convergence rate as vanilla FL. Extensive experiments empirically show that our framework brings up to $5.56\times$ speedup compared with traditional FL. Code is available at: *https://github.com/Hypervoyager/PFL*.

1. Introduction

Federated Learning (FL) is an emerging distributed machine learning framework that enables numerous devices to collaboratively train a shared model without exposing their private data (Shokri & Shmatikov, 2015; Konečnỳ et al., 2016; Kairouz et al., 2019; Li et al., 2020). The main advantage of FL is the decoupling of model training from the necessity of directly accessing to the training data. Thus, in many data-sensitive scenarios, such as applications in biomedicine (Xu et al., 2021b; Courtiol et al., 2019; Xu et al., 2021b) and finance (Long et al., 2021; Pfitzner et al., 2021), FL can significantly reduce the concern of privacy and security.

Following the pioneer FedAvg (McMahan et al., 2017), most existing FL schemes work in synchronous manner (SyncFL), in which each round includes the following steps (Zhang et al., 2023): 1) the central server broadcasts the latest global model to edge devices; 2) each edge device then updates its local model with the private data, and uploads the local model update to the central server; 3) once *all* local updates are received, the central server conducts the aggregation operation to produce the next global model. In the scenario of device heterogeneity, which widely exists in practical applications, two issues heavily hinder the training efficiency of SyncFL:

– **Network Stragglers.** Device heterogeneity inevitably introduces network stragglers, which take much longer time to complete local training than common ones. According to the rule that aggregation is conducted after *all* local updates are received, due to either the slowest training speed or dropping out mid-round, the slowest-responding client becomes the bottleneck of training efficiency of heterogeneous FL.

– **Complicated Global Aggregation.** The weighted average manner is widely applied for global aggregation, which is simple and fast. As a result, in most existing FL schemes, the aggregation time spent in the central server is ignored. However, for the scenario of FL over heterogeneous devices, directly averaging local models is not feasible anymore, since local models may have diverse structures and sizes. Recently, some methods attempt to address the model heterogeneity by performing knowledge distillation in the server side (Zhu et al., 2021b; Lin et al., 2020), leading to complicated aggregation operation. When the time cost for aggregation is significant, as shown in Fig.1-(a), devices are blocked to wait for the central server, which limits the

[1]Faculty of Computing, Harbin Institute pf Technology, Harbin, China [2]Peng Cheng Laboratory, Shenzhen, China [3]Department of Automation, Tsinghua University, Beijing, China. Correspondence to: Xianming Liu <csxm@hit.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

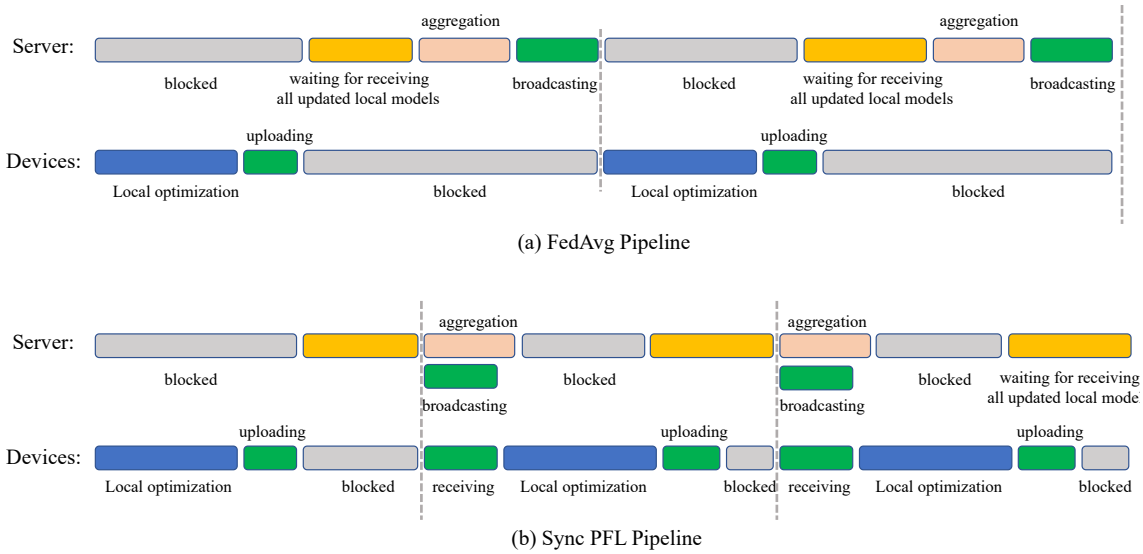

Figure 1. The pipeline of FedAvg and Sync PFL. (a) FedAvg works in a synchronous manner, in which local and global computation are conducted serially. (b) Sync PFL exchanges the execution order of global aggregation and broadcasting.

utilization (*i.e.*, the fraction of processors actively computing at any time) of the system. Accordingly, complicated global aggregation becomes another key factor influencing the training efficiency of heterogeneous FL.

The network latency induced by network straggler has been well studied, while the training latency induced by complicated aggregation is overlooked by the community. The state-of-the-art synchronous and asynchronous FL training strategies attempt to remedy the network latency issue from the edge side. For instance, Delayed gradient averaging (DGA) (Zhu et al., 2021a) is proposed to conduct extra local optimization during the process of communication, which can save the overall time cost to some extent. However, as a synchronous approach, DGA still suffers from the high network latency induced by network stragglers. An alternative strategy is to allow clients to update the global model asynchronously (Xie et al., 2019; Xu et al., 2021a; Dun et al., 2022). For instance, Avdyukhin *et al.* (Avdiukhin & Kasiviswanathan, 2021b) propose an asynchronous local SGD model, where all the client iterates evolve in synchrony with respect to the global clock, but communicate with the server in asynchrony at arbitrary time intervals.

Instead of working on the edge side, in this paper, we claim that the bottleneck of training latency actually lies in the workflow of the server. This research line has not been explored yet. We propose a novel and efficient scheme called *parallel federated learning* (PFL) to achieve training efficiency over heterogeneous devices, which can solve the above two issues simultaneously. Specifically, the popular FedAvg conducts serial computation between edge nodes and the server: uploading → global aggregation → broadcasting → local optimization. In this pipeline, as shown in Fig.1-(a), the server would be blocked to wait for receiving *all local updates* to perform the global aggregation; the edge nodes would be blocked to wait for receiving the broadcasted global model to start the next local optimization. By carefully examining the above process, it can be found that the root cause of training inefficiency actually lies in the dependency of server on all local updates uploading and the dependency of edge on global aggregation. An intuitive idea is to decouple these dependencies such that the edge and server can run in parallel.

Without redesigning the whole FL framework, we propose a very simple operation—exchange the execution order of global aggregation and broadcasting. The new workflow of the server becomes: uploading (server receives the currently completed local updates) → broadcasting (server broadcasts the stale global model in the buffer) → global aggregation (server averages the currently received local updates to get the next global model). Under this setup, for an edge device, once the local optimization is completed, it uploads the local update to the server; right after the server receives the new local update, it immediately broadcasts the stale global model in the buffer to the edge devices; the edge devices no longer need to wait for the aggregation result but start the next local optimization upon the received global model; the global aggregation on new updates is then carried out. In this way, the global aggregation is decoupled from the communication and can be performed in parallel with local training. PFL includes both synchronous and asynchronous versions, as shown in Fig. 1-(b) and Fig. 2. We theoretically prove that PFL can achieve the similar convergence rate as FedAvg, and empirically show that our framework can

tolerate both network stragglers and complicated aggregation operation, which brings $1.77\times$ to $5.56\times$ speedup. In summary, PFL enjoys the following merits:

– Handling Device Heterogeneity and Network Stragglers. Edge clients and the server conduct model updates in a fully asynchronous fashion, which is favorable in handling device heterogeneity. Moreover, the server no longer needs to wait for the straggling devices, which upload the local updates whenever they are available to the server.

– Flexible Device Participation. In cross-device FL, it is expected that each client would only participate in an arbitrary number of update rounds. Our PFL can achieve flexible device participation naturally. Once a local client gets disconnected, it can download the global model into the buffer, upon which it can conduct a local model update and rejoin the collaborative training circle.

– Scalability in Computation. In PFL, the global aggregation is executed in parallel with communication and local updates, thus it can be generalized to conduct more complicated aggregation operations beyond simple averaging. For instance, we can employ knowledge distillation to deal with the model heterogeneity or use homomorphic encryption to boost security.

2. Methodology

We consider federated learning with N edge devices, each of which owns a private dataset $\mathcal{S}_i = \{(\mathbf{x}_j^i, y_j^i)\}_{j=1}^{m_i}$ including m_i training samples from an unknown and fixed distribution $\mathcal{P}_i$ over $\mathcal{X} \times \mathcal{Y}$. The sample number m_i and the data distribution $\mathcal{P}_i$ can be diverse across edge devices. The overall goal is to train a global model $f(\omega)$ using samples from all clients, where ω is denoted as the model parameters to be optimized. Formally, for each device i, an objective function $F^{(i)}(\omega)$ is defined for optimization over $\mathcal{S}_i$:

$$F^{(i)}(\omega) = \mathbb{E}_{(\mathbf{x},y)\in\mathcal{S}_i}[L(f(\mathbf{x},\omega), y)], \tag{1}$$

2.1. Parallel Federated Learning

The proposed PFL can be conducted in both synchronous and asynchronous modes. For the synchronous mode, in each round, the server needs to wait until receiving all updates of edge devices prior to conducting the global aggregation; while for the asynchronous mode, the server allows asynchronous updates. In the following, we introduce them in detail.

2.1.1. Synchronous PFL

Due to the limitation of synchronized aggregation, our synchronous PFL (SPFL) cannot handle the bottleneck induced by network stragglers, which instead is tailored to alleviate the training latency induced by complicated aggregation.

Algorithm 1 Asynchronous Parallel Federated Learning

Input: N edge devices with private datasets $\{\mathcal{S}_i\}_{i=1}^N$, the maximum global clock number T.
Output: The final global model $\omega_T^{(g)}$
Initialization: $\omega_0^{(g)}$;
for global clock $t = 1, ...T$ **in parallel do**
  for edge device i corresponding to global clock t
  – **Received global models:** get $\omega_{t-1}^{(g)}$
  – **Local Update:** $\omega_t^{(i)} \leftarrow \omega_{t-1}^{(g)} - \gamma G_{t-1}^{(i)}$;
  – **Uploading:** Uploading the gradient updates $\delta^{(i)}$ to the central server;
  Global Computation on Server
  – **Communication:**
  • Receive the gradient update $\delta^{(i)}$ of arriving device i at global clock $(t+n+1)$;
  • Send back the stale global model $\omega_{t+n-r}^{(g)}$ to device i;
  – **Aggregation:**
  • Update the global model:
  $\omega_{t+n+1}^{(g)} \leftarrow \omega_{t+n+1-p}^{(g)} + \frac{1}{N}\sum_{i\in\mathcal{C}_t}\delta^{(i)}$,

Specifically, for the t-th round, SPFL works as follows:

– Local Update: Based on the received global model $\omega_{t-1}^{(g)}$, each edge node i updates the local model by stochastic gradient descent using its private data:

$$\omega_t^{(i)} \leftarrow \omega_{t-1}^{(g)} - \gamma G_{t-1}^{(i)}, \quad t = 1, ..., T, \tag{2}$$

where $G_{t-1}^{(i)}$ represents the gradient updates computed by the i-th node based on $\omega_{t-1}^{(g)}$.

– Uploading: Once the edge node i finishes the local training, it uploads the gradient update $\delta^{(i)}$ to the central server.

– Broadcasting: Right after receiving the uploaded signals $\{\delta^{(i)}\}_{i=1}^N$ from all edge devices, the central server sends back to them *the stale global model* in the buffer. Note that it is the aggregation result of the last round but not the up-to-date one. If the aggregation procedure of the last round has not yet completed, the server needs to wait until its completion prior to broadcasting the global model.

– Global Aggregation: Along with broadcasting, the central server performs the aggregation operation based on the received local updates:

$$\omega_{t+1}^{(g)} = \omega_t^{(g)} + \frac{1}{N}\sum\nolimits_{i=1}^{N}\delta^{(i)}, \quad t = 1, ..., T, \tag{3}$$

When the central server is performing aggregation, the edge nodes are performing local training. Accordingly, by exchanging the execution order of broadcasting and aggregation, we decouple local training and global aggregation and achieve parallel speedup.

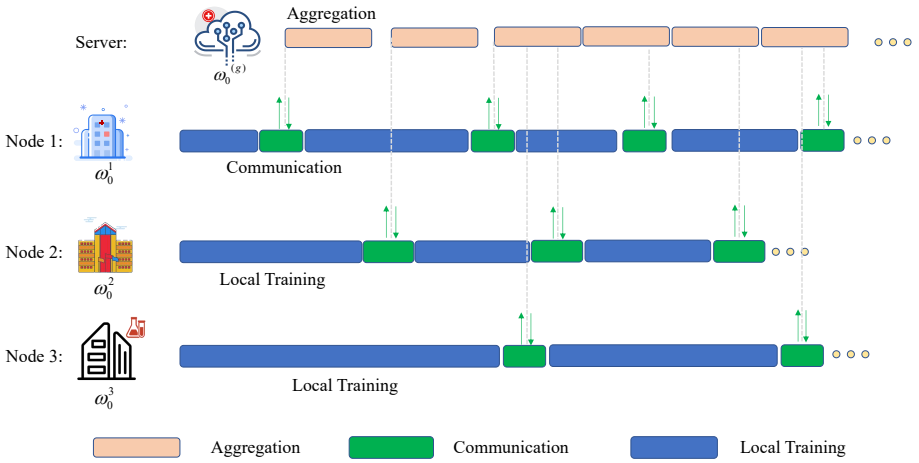

Figure 2. The pipeline of asynchronous PFL. The local training process of the edge device and the aggregation process of the central server can be carried out simultaneously, thus greatly improving the training efficiency of FL.

2.1.2. Asynchronous PFL

Compared with the synchronous mode, the asynchronous mode enjoys more degree of freedom. In the proposed asynchronous PFL (APFL), we attempt to simultaneously remedy the training latency induced by network stragglers on the edge side and complicated aggregation operation on the server side, as shown in Algorithm 1. In APFL, each edge device and the central server have different clocks. To handle this issue, we define the global clock: whenever any device communicates with the central server, the global clock will increase by 1. In other words, each global clock corresponds to only one device. Specifically, for the t-th global clock, the proposed PFL works as follows:

– Local Update: Edge device i completes the communication with the central server, which receives the global model denoted as $\omega_{t-1}^{(g)}$. Based on $\omega_{t-1}^{(g)}$, edge device i updates the local model by stochastic gradient descent using its private data:

$$\omega_t \leftarrow \omega_{t-1}^{(g)} - \gamma G_{t-1}^{(i)}, \quad t = 1, ..., T. \tag{4}$$

– Uploading: After completing the local training, edge device i then uploads the gradient update $\delta^{(i)}$ to the central server. During the local training and the uploading process of device i, there would be n other devices communicating with the central server. Therefore, when the central server receives the local update of the device i, the global clock becomes $t + n + 1$.

– Broadcasting: Right after receiving the uploaded signal $\delta^{(i)}$ from edge device i, the central server sends back to it *the stale global model* in the buffer, which is the most recent aggregation result. It would be $\omega_{t+n}^{(g)}$ if the central server is idle, or $\omega_{t+n-r}^{(g)}$ if the central server is performing aggregation, where r denotes the number of local devices that previously arrived at the server but did not participate in aggregation. The former is a special case of the latter with $r = 0$. For simplicity, we denote the stale global model as $\omega_{t+n-r}^{(g)}$. If the local model uploaded in the previous clock by that device has not completed aggregation, then the device must wait until it has. Note that the local signal $\delta^{(i)}$ received this time does not participate in the current aggregation process. Therefore, the edge device i no longer needs to wait for the global aggregation, which immediately starts the next local training relying on $\omega_{t+n-r}^{(g)}$. The central server then performs the next aggregation in parallel with the broadcasting and the new local training on device i.

– Global Aggregation: The new broadcasting-then-aggregation mechanism, which sends to clients the stale global models but not the latest ones, makes the global aggregation decoupled from the communication. However, this fully asynchronous manner leads to a scenario that, when the global aggregation is being executed, there are some local updates arriving. This situation is easy to handle: the arrivals just wait for the current aggregation to be done and then launch a new aggregation. The averaging operation is thus performed on the updates of the new received edge devices:

$$\omega_{t+n+1}^{(g)} \leftarrow \omega_{t+n+1-p}^{(g)} + \frac{1}{N} \sum\nolimits_{i \in \mathcal{C}_t} \delta^{(i)}, \quad t = 1, ..., T, \tag{5}$$

where $\mathcal{C}_t$ represents the devices that arrived during the last aggregation, p is the number of devices in $\mathcal{C}_t$. Normally, $\mathcal{C}_t$ contains only one device.

2.1.3. Comparison of SPFL and APFL

The main difference between SPFL and APFL is whether the central server needs to wait for all devices. SPFL is subject to the requirement of synchronized aggregation, while

APFL is not. This results in SPFL and APFL having different speed-up ratios compared to standard federated learning. Specifically, we divide the training time of federated learning into four parts: the time cost T_l for local training, the time cost T_c for communication, the time cost T_w for waiting, and the time cost T_g for global aggregation. Standard federated learning works in a serial manner, for which the training time of each round is $T = T_l + T_c + T_w + T_g$. In SPFL, the aggregation of the central server is performed in parallel with the local training of the edge nodes and the communication process, the training time for each round is $T = \max(T_l + T_c + T_w, T_g)$. In APFL, there is no waiting time, we have $T = \max(T_l + T_c, T_g)$.

2.2. Convergence Analysis

In this subsection, we present the convergence analysis to demonstrate in theory that the proposed synchronous/asynchronous PFL both can get a similar convergence as FedAvg. It is worth noting that there is a standard routine for FL convergence analysis, thus we just follow it without the necessity of creating a new one. We take the following standard assumptions (Stich, 2018; Avdiukhin & Kasiviswanathan, 2021a; Nguyen et al., 2022):

- **Smoothness:** All local functions $f^i (i \in [N])$ are L-smooth.
- **Bounded second moment:** There exists a constant $G_{max} > 0$ such that: $\mathbb{E}\left[||\nabla F^{(i)}(x)||^2\right] \leq G_{max}^2, \quad \forall i \in [N], \forall \mathbf{x} \in \mathbb{R}^d$,

  where $\nabla F^{(i)}(x)$ is an unbiased stochastic gradient of $f^{(i)}$ at x.
- **Bounded variance:** There exists a constant $\sigma \geq 0$ such that:

$$\mathbb{E}_{\xi \sim \mathcal{S}_i} \|\nabla F_i(\mathbf{x}) - \nabla f_i(\mathbf{x})\| \leq \sigma^2, \quad \forall i \in [N], \forall \mathbf{x} \in \mathbb{R}^d. \tag{6}$$

– Analysis for Synchronous PFL. In synchronous PFL, we only exchange the execution order of global aggregation and broadcasting, which makes each edge device receive the stale global model. At first glance, it looks a bit like asynchronous FL. The difference is that in asynchronous FL, only part of the devices (stragglers) upload the stale gradient updates. In synchronous PFL, the global model received by the devices is only from the last round. Due to the above similarities, we can analyze the convergence of synchronous PFL with the help of existing studies on asynchronous FL. Here, we follow (Avdiukhin & Kasiviswanathan, 2021a), but give a sharper bound.

Definition 1 (Synchronous PFL Sequence) *Let $\omega_t^{(g)}$ represents the global model of synchronous PFL, $G_t^{(i)}$ represents the uploaded gradient updates computed by the i-th client based on $\omega_t^{(g)}$, the global model $\omega_t^{(g)}$ can be expressed as:*

$$\omega_t^{(g)} = \omega_0^{(g)} - \sum_{\tau=2}^{t} \gamma \operatorname{avg}_i \left(G_{\tau-2}^{(i)}\right), \quad t = 2, ..., T, \tag{7}$$

where $\omega_0^{(g)}$ and $\omega_1^{(g)}$ are two different initialized models.

Definition 2 (Virtual Sequence) *We construct a virtual sequence in which each device at round t uses the $\omega_{t-1}^{(g)}$ to calculate the uploaded gradient updates, this virtual sequence's global model can be expressed as:*

$$\zeta_t = \omega_0^{(g)} - \sum_{\tau=1}^{t} \gamma \operatorname{avg}_i \left(G_{\tau-1}^{(i)}\right), \quad t = 1, ..., T. \tag{8}$$

Similar virtual sequences have been used before in decentralized optimization in various contexts (Lian et al., 2017; Yuan et al., 2016; Nedić et al., 2018; Stich, 2018; Avdiukhin & Kasiviswanathan, 2021a).

Proposition 3 (Distance Bound) *For ζ_t of virtual sequence and $\omega_t^{(g)}, \omega_t^{(i)}$ in synchronous PFL we have:*

$$\max\left(\mathbb{E}\left[\left\|\zeta_t - \omega_t^{(i)}\right\|^2\right], \mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|^2\right]\right) \leq 4\gamma^2 G_{\max}^2 \tag{9}$$

This Proposition shows that ζ_t are close to $\omega_t^{(g)}$ and $\omega_t^{(i)}$ for all devices. Its proof can be seen in Appendix.

Theorem 4 *Let $f_{\max} = f\left(\omega_0^{(g)}\right) - f\left(\omega^\star\right)$*, where $\omega^\star$ is the minimizer for f, $\gamma = \sqrt{N}/\sqrt{T}$ we have:

$$\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] = O\left(\frac{1}{\sqrt{NT}} + \frac{N}{T}\right). \tag{10}$$

Compared to the corresponding result for (Basu et al., 2019) and (Avdiukhin & Kasiviswanathan, 2021a), $O\left(\frac{1}{\sqrt{NT}}\right)$, when N is fixed, our results get a sharper convergence rate. And illustrates a clearer relationship between T and N, i.e. as the number of nodes N increases, the number of communication rounds T should be larger as well.

– Analysis for Asynchronous PFL. Convergence of asynchronous PFL can be demonstrated in a similar way as above, with the difference that in asynchronous PFL, the global model received by the device will be more than one clock staler than the current global model when aggregated. For asynchronous PFL, we can use almost the same method of (Avdiukhin & Kasiviswanathan, 2021a) to prove the convergence of our method. Our contribution does not lie in the novelty of the method of proof, but rather in pointing out that our approach can be fully accommodated in existing proofs of asynchronous FL and achieve greater speed-up

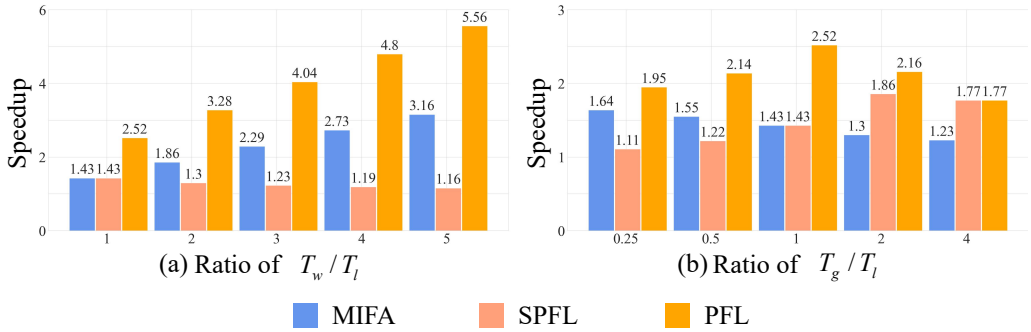

Figure 3. The speedup ratio comparison under different setting. T_w/T_l represents the ratio of the straggler device's dropout time to the local training time. Ratio of T_g/T_l is associated with the number of devices and the computing power of the central server. Experimental results show that PFL yields the highest speedup ratios in both situations.

ratios. For convenience, we use a notion similar to that used in synchronous PFL to describe the global model and the local model. Except for local models, we use w_t to represent the local model, since each global clock only corresponds to one device.

Definition 5 (Asynchronous PFL Sequence) *Let $\omega_t^{(g)}$ represents the global model of asynchronous PFL, $G_t^{(i)}$ represents the uploaded gradient updates from the i-th client based on $\omega_t^{(g)}$, the global model $\omega_t^{(g)}$ can be expressed as:*

$$\omega_t^{(g)} = \omega_0^{(g)} - \sum_{\tau=1}^{t} \gamma G_{\tau-\delta_t}^{(i)}, \quad t = 1, ..., T, \tag{11}$$

where $\tau - \delta_t$ denotes the clock of the global model received by the device participating in the t-th round of aggregation, and δ_t can represent the degree of asynchrony.

Proposition 6 (Distance Bound) *For ζ_t of virtual sequence and $\omega_t^{(g)}, \omega_t$, we have:*

$$\begin{aligned} &\max\left(\mathbb{E}\left[\|\zeta_t - \omega_t\|^2\right], \mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|)^2\right]\right) \\ &\leq 4(\delta-1)^2\gamma^2 G_{\max}^2, \end{aligned} \tag{12}$$

Theorem 7 *Let $f_{\max} = f\left(\omega_0^{(g)}\right) - f\left(\omega^\star\right)$,* where $\omega^\star$ is the minimizer for f, if $T > N^3$ and $\delta \leq T^{1/4}/N^{3/4}$ we have:

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] = O\left(\frac{Lf_{\max}}{\sqrt{NT}} + \frac{\mathbf{G}_{\max}^2}{\sqrt{NT}} + \frac{L\sigma^2}{\sqrt{NT}}\right), \tag{13}$$

where $\delta = \max(\delta_t)$.

3. Experiments

In this section, we provide a thorough evaluation of the proposed PFL over heterogeneous devices. We first compare the convergence speed and test accuracy of PFL under average-based aggregation. Next, to investigate the speedup ratio of PFL under different scenarios, experiments are carried out with different T_w/T_l and T_g/T_l, where T_w represents the reconnection time of the straggler devices, T_l represents the local training time, and T_g represents the aggregation time. We then show that our method is also feasible for distillation-based aggregation. The convergence curve of our method is also provided. Experimental results show that our method is resilient to stragglers and can achieve a parallel pipeline to improve computational efficiency without sacrificing test accuracy largely.

3.1. Experimental Setup

Straggler Devices and Sample Rate. Following the previous study (So et al., 2022), we randomly select $p \times N$ devices where p is the dropout rate. To constrain the asynchronous level between devices, we set $T_w/T_l = 3$, which represents the ratio of waiting time to local training time. We consider the worst-case scenario (Bonawitz et al., 2017), where the straggler devices artificially drop after receiving the global model. According to the realistic FL system (Bonawitz et al., 2019), we set $p = 0.2$.

Datasets and Models. We conduct experiments on three datasets, including MNIST, CIFAR-10, and Tiny-Imagenet (100,000 images with 200 classes). For all three datasets, we use the same ResNet-18 architecture for a fair and clear comparison.

Communication bandwidth. In our experiments, we are particularly focused on mobile devices in FL, the real measured bandwidth of a mobile phone is 4MB/s. The cost of time to upload and broadcast updated local and global models, denoted as T_c.

Baselines. We analyze and compare the performance of PFL with three baseline schemes: FedAvg (McMahan et al., 2017), DGA (Zhu et al., 2021a), and MIFA (Gu et al., 2021). DGA allows extra local optimization during the communication process to implicitly reduce wasted computing power

Table 1. Accuracy (%) comparison of synchronous FedAvg and our PFL's on 3 datasets with both i.i.d and non-i.i.d data partitions. Speed-up is measured for each global clock, normalized by the run-time of FedAvg. The best result is **boldfaced**.

Method	MNIST		CIFAR-10		Tiny-ImageNet		Time for per global clock
	i.i.d	non-i.i.d	i.i.d	non-i.i.d	i.i.d	non-i.i.d	
FedAvg (McMahan et al., 2017)	**93.21**	**85.31**	85.41	75.26	59.52	48.14	$T = T_l + T_c + T_w + T_g$
	$1\times$		$1\times$		$1\times$		
DGA (Zhu et al., 2021a)	92.12	83.72	83,72	73.24	58.11	46.26	$T = T_l + (T_c + T_w + T_g)$ $= T_l + T_E$
	$1\times$		$1\times$		$1\times$		
MIFA (Gu et al., 2021)	92.88	84.73	85.14	74.78	59.24	47.12	$T = T_l + T_c + T_g$
	$1.57\times$		$1.86\times$		$1.98\times$		
Our SPFL	93.14	85.26	**85.57**	75.31	**60.21**	**48.31**	$T = \max(T_l + T_c, T_g) + T_w$
	$1.22\times$		$1.30\times$		$1.33\times$		
Our APFL	93.08	85.13	85.34	**75.46**	60.18	48.07	$T = \max(T_l + T_c, T_g)$
	$\mathbf{2.20}\times$		$\mathbf{3.28}\times$		$\mathbf{3.91}\times$		

during communication. MIFA is an asynchronous communication FL method to solve straggler problems. These two methods address the challenge of training efficiency in FL from two different perspectives.

Data Partition: Following (Avdiukhin & Kasiviswanathan, 2021a; Lin et al., 2023), for IID setup, we randomly split the data equally into N sub-datasets; for Non-IID setup, each device has a corresponding class, where μ fraction of local data is from the corresponding class, while the rest are randomly selected from other classes. In our experiments, $\mu = 0.3$.

3.2. Performance Evaluation

To fully explore the communication efficiency of FL, we simulate all the three scenarios mentioned in the above section, concretely, 1) devices will randomly drop out, 2) the upload and download time of the models is measured by bandwidth 4Mb/s, and 3) the central server needs to perform additional complex computations such as knowledge distillation or toxic device identification.

Average-based Aggregation. Although various novel aggregation methods have been proposed, FedAvg is still the most widely used method, so we first investigate the performance of our method under the average-based approach. To simulate complex operations on the central server in the real environment without introducing additional effects, we perform additional complex tasks on the central server to demonstrate the superiority of the training efficiency of our approach. Specifically, an additional stage of toxic device identification is added before the central server performs the aggregation. The local models uploaded from each device need to be validated by the public dataset on the central server before participating in aggregation. This is a very efficient but time-consuming detection method and, therefore, is rarely used in real-world scenarios. As can be seen in Table 1, our method can achieve almost the same accuracy as FedAvg, while achieving a speed increase of approximately $3.91\times$. Speed-up is measured in run time per global clock between device and central server, normalized by the run time of FedAvg. DGA does not reduce the computation time for each communication but rather speeds up convergence by allowing the nodes to perform local training while communicating. Therefore, it has an acceleration ratio of only $1\times$. MIFA can get good speedups in the presence of straggler, e.g. $1.86\times$ faster on the CIFAR-10 dataset. Our approach can improve the computational efficiency of federated learning in three dimensions simultaneously, the time for per global clock is $T = \max(T_l + T_c, T_g)$, for example, when on the CIFAR-10 dataset, our method can obtain a $3.28\times$ speedup.

Impact of T_w/T_l on speedup ratio. T_w/T_l represents the ratio of reconnection time of straggler devices to local training time. The longer the dropout time, the greater its impact on the overall computational efficiency of SyncFL and the greater its impact on the accuracy of AsyncFL. Therefore, it is necessary to investigate the effect of different T_w/T_l on computational efficiency, here we set $T_w/T_l = 1$ and $T_w/Tl = 1, 2, 3, 4, 5$, and the experiment is carried out on the CIFAR-10. As can be seen in Fig. 3, as T_w/T_l increases, the computational speed-up ratio between our method and MIFA's method gets higher and higher, and when $T_w/T_l = 5$, the computational efficiency of our PFL can be improved by up to $5.56\times$, while MIFA can only improve the speed-up ratio by $3.16\times$.

Impact of T_g/T_l on speedup ratio. T_g/T_l represents the ratio of the central server's computation time to the local device's training time, which is related to the number of devices and the computation capacity of the central server, and is further complicated when the computation power of

Table 2. Training efficiency (%) comparison of FedAvg and PFL on three datasets with different local epochs under distillation-based aggregation. The communication time T_c and the stragglers are ignored, the times in the table are the average times for each training between edge device and central server.

Dataset	Local epoch	T_l (s)	T_g (s)	Fedavg		PFL		Speedup
				Time(s)	Acc.	Time(s)	Acc.	
MNIST	5	4	5	9	99.31	5	99.18	1.80 ×
	10	7	5	12	98.17	7	98.09	1.71 ×
	15	11	5	16	96.13	11	96.24	1.45 ×
CIFAR-10	5	19	20	39	89.22	20	89.13	1.95 ×
	10	38	20	58	87.81	38	87.76	1.53 ×
	15	60	20	80	86.48	60	86.37	1.33 ×
Tiny-ImageNet	5	185	180	365	62.88	185	62.79	1.97 ×
	10	375	180	555	60.13	375	60.06	1.48 ×
	15	563	180	743	57.25	563	57.22	1.32 ×

the edge devices is different from each other. To simplify the computational model, we assume that the local training time is the same for each edge device. Here, we set $T_g/T_l = 1$ and $T_w/Tl = 0.25, 0.5, 1, 2, 4$, and the experiment is carried out on the CIFAR-10. As can be seen in Fig. 3, the speed-up ratio of our method reaches its maximum $2.52\times$ when $T_g/T_l = 1$.

Distillation-based Aggregation. To verify the effectiveness of our method for complex aggregations, we have chosen the first distillation-based aggregation method (Lin et al., 2020). The distillation-based approach makes better use of information from edge devices, but introduces serious computational reductions in efficiency that make it difficult to apply to real-world applications. Following (Lin et al., 2020), we added additional public datasets to the central server and then fine-tuned the global model by distillation after coarse aggregation was completed by averaging operations. In this experiment, the straggler node is ignored. As can be seen from Table 2, our method is still feasible under distillation-based aggregation and can improve the computational efficiency without sacrificing accuracy, achieving $2\times$ speedup when the training time for edge devices is the same as the aggregation time for the central server.

Convergence Analysis. In our scheme, the key to achieving parallel computing is to send the stale global model to the edge devices. It would be interesting to investigate the influence of the stale global model on the final performance. In this part, we conduct a convergence analysis through empirical experiments with respect to both accuracy and communication rounds on the CIFAR-10 dataset with 20 edge devices. The convergence curve is provided in Fig. 4. It can be found that, in the early training stage, the proposed PFL suffers from lower accuracy and stronger oscillations compared with FedAvg; while with the communication round increases, the PFL finally achieves similar performance to FedAvg. This is consistent with the intuition

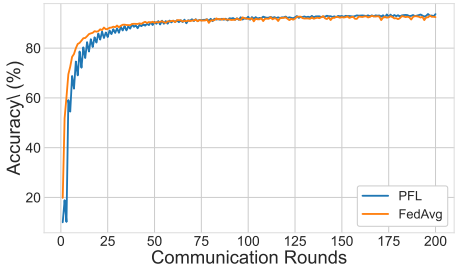

Figure 4. Convergence analysis of PFL and FedAvg on CIFAR-10.

that, when the algorithm converges, there is no significant difference between using the stale global model or using the up-to-date global model.

4. Conclusion

The traditional FL works in a serial computation manner, in which communication is deemed a major bottleneck, leading to poor training efficiency. In this paper, we presented a simple yet efficient scheme called parallel federated learning (PFL). Instead of remedying the efficiency issue from the edge side as done in existing methods, we claim that the bottleneck of computational efficiency actually lies in the workflow of the server. Through exchanging the execution order of global aggregation and broadcasting, global aggregation is decoupled from the communication and can be performed in parallel with local training. We theoretically prove that PFL can achieve the same convergence rate as FedAvg. The proposed scheme has the potential to serve as a new baseline FL framework enabling a wide range of applications.

Acknowledgements

This work was supported by National Natural Science Foundation of China under Grants 92270116 and 62071155.

References

Avdiukhin, D. and Kasiviswanathan, S. Federated learning under arbitrary communication patterns. In *International Conference on Machine Learning*, pp. 425–435. PMLR, 2021a.

Avdiukhin, D. and Kasiviswanathan, S. Federated learning under arbitrary communication patterns. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 425–435. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/avdiukhin21a.html.

Basu, D., Data, D., Karakus, C., and Diggavi, S. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. *Advances in Neural Information Processing Systems*, 32, 2019.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

Courtiol, P., Maussion, C., Moarii, M., Pronier, E., Pilcer, S., Sefta, M., Manceron, P., Toldo, S., Zaslavskiy, M., Le Stang, N., et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine*, 25(10):1519–1525, 2019.

Dun, C., Garcia, M. D. C. H., Jermaine, C., Dimitriadis, D., and Kyrillidis, A. Efficient and light-weight federated learning via asynchronous distributed dropout. In *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022. URL https://openreview.net/forum?id=DNzb6IdvoUX.

Gu, X., Huang, K., Zhang, J., and Huang, L. Fast federated learning in the presence of arbitrary device unavailability. *Advances in Neural Information Processing Systems*, 34, 2021.

Kairouz, P., McMahan, H. B., and et al. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019. URL http://arxiv.org/abs/1912.04977.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*, 2017.

Lin, S., Zhai, D., Zhang, F., Jiang, J., Liu, X., and Ji, X. Overhead-free noise-tolerant federated learning: A new baseline. *Machine Intelligence Research*, 2023.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2351–2363. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf.

Long, G., Tan, Y., Jiang, J., and Zhang, C. Federated learning for open banking. *CoRR*, abs/2108.10749, 2021. URL https://arxiv.org/abs/2108.10749.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Nedić, A., Olshevsky, A., and Rabbat, M. G. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106 (5):953–976, 2018.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., and Huba, D. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pp. 3581–3607. PMLR, 2022.

Pfitzner, B., Steckhan, N., and Arnrich, B. Federated learning in a medical context: A systematic literature review. *ACM Trans. Internet Technol.*, 21(2), jun 2021. ISSN 1533-5399. doi: 10.1145/3412357. URL https://doi.org/10.1145/3412357.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, 2015.

So, J., Nolet, C. J., Yang, C.-S., Li, S., Yu, Q., E Ali, R., Guler, B., and Avestimehr, S. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.

Stich, S. U. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Xie, C., Koyejo, S., and Gupta, I. Asynchronous federated optimization. *CoRR*, abs/1903.03934, 2019. URL http://arxiv.org/abs/1903.03934.

Xu, C., Qu, Y., Xiang, Y., and Gao, L. Asynchronous federated learning on heterogeneous devices: A survey. *CoRR*, abs/2109.04269, 2021a. URL https://arxiv.org/abs/2109.04269.

Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., and Wang, F. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, 2021b.

Yuan, K., Ling, Q., and Yin, W. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Zhang, F., Li, Y., Lin, S., Jiang, J., Liu, X., et al. Large sparse kernels for federated learning. In *International Conference on Learning Representations, Tiny Papers*, 2023.

Zhu, L., Lin, H., Lu, Y., Lin, Y., and Han, S. Delayed gradient averaging: Tolerate the communication latency for federated learning. *Advances in Neural Information Processing Systems*, 34, 2021a.

Zhu, Z., Hong, J., and Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12878–12889. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/zhu21b.html.

A. Convergence analysis of PFL

Proof for Synchronous PFL.

Proposition 3 (Distance Bound) *For ζ_t of full synchronous FL and $\omega_t^{(g)}, \omega_t^{(i)}$ of Algorithm 1, we have:*

$$\max\left(\mathbb{E}\left[\left\|\zeta_t - \omega_t^{(i)}\right\|^2\right], \mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|^2\right]\right) \leq 4\gamma^2 G_{\max}^2. \tag{14}$$

Proof:

$$\begin{aligned}
\mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|^2\right] &= \mathbb{E}\left[\left\|\sum_{\tau=2}^{t}\gamma\,\mathrm{avg}_i\left(G_{\tau-2}^{(i)}\right) - \sum_{\tau=1}^{t}\gamma\,\mathrm{avg}_i\left(G_{\tau-1}^{(i)}\right)\right\|^2\right] \\
&\leq \gamma^2\mathbb{E}\left[\left\|\sum_{\tau=2}^{t}\mathrm{avg}_i\left(\mathbf{G}_{\tau-2}^{(i)} - \mathbf{G}_{\tau-1}^{(i)}\right) - \gamma\,\mathrm{avg}_i\,\mathbf{G}_0^{(i)}\right\|^2\right] \\
&\leq \gamma^2\mathbf{G}_{\max}^2.
\end{aligned} \tag{15}$$

Similarly, $\mathbb{E}\left[\left\|\omega_t^{(g)} - \omega_t^{(i)}\right\|^2\right] \leq \gamma^2\mathbf{G}_{\max}^2$. Combining these bounds, we have the following.

$$\begin{aligned}
\mathbb{E}\left[\left\|\zeta_t - \omega_t^{(i)}\right\|^2\right] &= \mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)} + \omega_t^{(g)} - \omega_t^{(i)}\right\|^2\right] \\
&\leq 2\left(\mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|^2\right] + \mathbb{E}\left[\left\|\omega_t^{(g)} - \omega_t^{(i)}\right\|^2\right]\right) \\
&\leq 4\gamma^2\mathbf{G}_{\max}^2.
\end{aligned} \tag{16}$$

Theorem 4 *Let $f_{\max} = f\left(\omega_0^{(g)}\right) - f\left(\omega^\star\right)$*, where $\omega^\star$ is the minimizer for f, we have:

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] = O\left(\frac{1}{\sqrt{NT}} + \frac{N}{T}\right). \tag{17}$$

Proof: Similar to (Avdiukhin & Kasiviswanathan, 2021a) Theorem 2.4.

From smoothness Lipschitz condition on the gradients:

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right) - \nabla f\left(\zeta_t\right)\right\|^2\right] \leq L^2\mathbb{E}\left[\left\|\omega_t^{(g)} - \zeta_t\right\|^2\right] \leq L^2\gamma^2\mathbf{G}_{\max}^2, \text{ and} \\
&\mathbb{E}\left[\left\|\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right\|^2\right] \leq L^2\mathbb{E}\left[\left\|\omega_t^{(i)} - \zeta_t\right\|^2\right] \leq 4L^2\gamma^2\mathbf{G}_{\max}^2.
\end{aligned} \tag{18}$$

First, we bound ζ_t, for ζ_t, we have:

$$\zeta_{t+1} = \zeta_t - \gamma\,\mathrm{avg}_i\left(G_\tau^{(i)}\right). \tag{19}$$

By the smoothness property:

$$\mathbb{E}\left[f\left(\zeta_{t+1}\right)\right] \leq \mathbb{E}\left[f\left(\zeta_t\right)\right] - \mathbb{E}\left[\left\langle\nabla f\left(\zeta_t\right), \gamma\,\mathrm{avg}_i\left(\mathbf{G}_t^{(i)}\right)\right\rangle\right] + \frac{L}{2}\mathbb{E}\left[\left\|\gamma\,\mathrm{avg}_i\left(\mathbf{G}_t^{(i)}\right)\right\|^2\right]. \tag{20}$$

The last term in Eq. 20 can be rewritten as:

$$
\begin{aligned}
&\frac{L}{2}\mathbb{E}\left[\left\|\gamma \operatorname{avg}_i\left(\mathbf{G}_t^{(i)}\right)\right\|^2\right] \\
&= \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\mathbf{G}_t^{(i)} + \nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] \\
&= \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right) + \left(\mathbf{G}_t^{(i)} - \nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right)\right\|^2\right] \\
&= \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] + \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\mathbf{G}_t^{(i)} - \nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] \\
&= \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] + \gamma^2\frac{L\sigma^2}{2}.
\end{aligned}
\tag{21}
$$

Substituting this into the Eq. 20, get:

$$
\begin{aligned}
\mathbb{E}\left[f\left(\zeta_{t+1}\right)\right] \leq& \mathbb{E}\left[f\left(\zeta_t\right)\right] - \mathbb{E}\left[\left\langle \nabla f\left(\zeta_t\right), \gamma \operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\rangle\right] \\
&+ \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] + \gamma^2\frac{L\sigma^2}{2} \\
\leq& \mathbb{E}\left[f\left(\zeta_t\right)\right] - \gamma\mathbb{E}\left[\left\langle \nabla f\left(\zeta_t\right), \operatorname{avg}_i\left(\nabla f^{(i)}\left(\zeta_t\right)\right)\right\rangle\right] \\
&- \gamma\mathbb{E}\left[\left\langle \nabla f\left(\zeta_t\right), \operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right)\right\rangle\right] \\
&+ \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] + \gamma^2\frac{L\sigma^2}{2}.
\end{aligned}
\tag{22}
$$

The second term in Eq. 22 can be simplified by $\operatorname{avg}_i\left(f^{(i)}\left(\zeta_t\right)\right) = f\left(\zeta_t\right)$:

$$
\gamma\mathbb{E}\left[\left\langle \nabla f\left(\zeta_t\right), \operatorname{avg}_i\left(\nabla f^{(i)}\left(\zeta_t\right)\right)\right\rangle\right] = \gamma\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right]. \tag{23}
$$

For the third term in Eq. 22 we have:

$$
\begin{aligned}
&\gamma\mathbb{E}\left[\left\langle \nabla f\left(\zeta_t\right), \operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right)\right\rangle\right] \\
&\leq \frac{\gamma}{2}\left(\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + \mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right)\right\|^2\right]\right) \\
&\leq \frac{\gamma}{2}\left(\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + \operatorname{avg}_i\left(\mathbb{E}\left[\left\|\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right\|^2\right]\right)\right) \\
&\leq \frac{\gamma}{2}\left(\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + 4L^2\gamma^2 G_{\max}^2\right). \quad \text{(According to Eq. 18)}
\end{aligned}
\tag{24}
$$

For the fourth term in Eq. 22 we have:

$$
\begin{aligned}
&\frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right)\right)\right\|^2\right] \\
&= \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\zeta_t\right) + \left(\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right)\right)\right\|^2\right] \\
&\leq \gamma^2 L\left(\mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\zeta_t\right)\right)\right\|^2\right] + \mathbb{E}\left[\left\|\operatorname{avg}_i\left(\nabla f^{(i)}\left(\omega_t^{(i)}\right) - \nabla f^{(i)}\left(\zeta_t\right)\right)\right\|^2\right]\right) \\
&\leq \gamma^2 L\left(\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + 4L^2\gamma^2\mathbf{G}_{\max}^2\right). \quad \text{(According to Eq. 18)}
\end{aligned}
\tag{25}
$$

Substituting Eq. 23, 24, 25 into the Eq. 22, get:

$$\begin{aligned}\mathbb{E}\left[f\left(\zeta_{t+1}\right)\right] &\leq \mathbb{E}\left[f\left(\zeta_t\right)\right] - \gamma\left(1+\frac{1}{2}-\gamma L\right)\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] - 2\gamma^3 L^2 \mathbf{G}_{\max}^2 + 4\gamma^4 L^3 \mathbf{G}_{\max}^2 \\ &\quad + \gamma^2 \frac{L\sigma^2}{2N} \qquad \text{(Assume } \gamma \leq 1/(2L)) \\ &\leq \mathbb{E}\left[f\left(\zeta_t\right)\right] - \gamma \mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + \gamma^2 \frac{L\sigma^2}{2N}.\end{aligned} \tag{26}$$

Move $\mathbb{E}\left[|| \nabla f(\zeta_t) ||^2\right]$ to the left of the inequality;

$$\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] \leq \frac{\left(\mathbb{E}\left[f\left(\zeta_t\right)\right] - \mathbb{E}\left[f\left(\zeta_{t+1}\right)\right]\right)}{\gamma} + \gamma \frac{L\sigma^2}{2N}. \tag{27}$$

Taking the sum over all iterations:

$$\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] \leq \frac{\left(\mathbb{E}\left[f\left(\zeta_0\right)\right] - \mathbb{E}\left[f\left(\zeta_{T+1}\right)\right]\right)}{\gamma T} + \gamma \frac{L\sigma^2}{2N}. \tag{28}$$

Finally, we can bound $|| \nabla f(\omega_t^{(g)}) ||$ in terms of $|| \nabla f(\zeta_t) ||$ as:

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] &\leq 2\left(\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right) - \nabla f\left(\zeta_t\right)\right\|^2\right] + \mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right]\right) \\ &\leq 2\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + 2L^2 \mathbb{E}\left[\left\|\omega_t^{(g)} - \zeta_t\right\|^2\right] \\ &\leq 2\mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right] + \gamma^2 L^2 \mathbf{G}_{\max}^2.\end{aligned} \tag{29}$$

Substituting this into the inequality above on $\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left[\left\|\nabla f\left(\zeta_t\right)\right\|^2\right]$ gives the claimed bound:

$$\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] = O\left(\frac{f_{\max}}{\gamma T} + \gamma^2 L^2 \mathbf{G}_{\max}^2 + \gamma \frac{L\sigma^2}{N}\right). \tag{30}$$

Using the step size $\gamma = \sqrt{N}/\sqrt{T}$, we get:

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] \\ &= O\left(\frac{f_{\max}}{\sqrt{NT}} + \frac{N}{T} L^2 \mathbf{G}_{\max}^2 + \frac{L\sigma^2}{\sqrt{NT}}\right).\end{aligned} \tag{31}$$

Proof for Asynchronous PFL. We use the same notation in synchronous PFL for convenience. In asynchronous PFL, we can use almost the same method of (Avdiukhin & Kasiviswanathan, 2021a) to prove the convergence of our method. Our contribution does not lie in the novelty of the method of proof, but rather in pointing out that our approach can be fully accommodated in existing proofs of asynchronous FL and achieve greater speed-up ratios.

Definition 5 (Asynchronous PFL Sequence) *Let $\omega_t^{(g)}$ represents the global model of asynchronous PFL, $G_t^{(i)}$ represents the uploaded gradient updates computed by the i-th client based on $\omega_t^{(g)}$, the global model $\omega_t^{(g)}$ can be expressed as:*

$$\omega_t^{(g)} = \omega_0^{(g)} - \sum_{\tau=1}^{t} \gamma G_{\tau-\delta}^{(i)}. \tag{32}$$

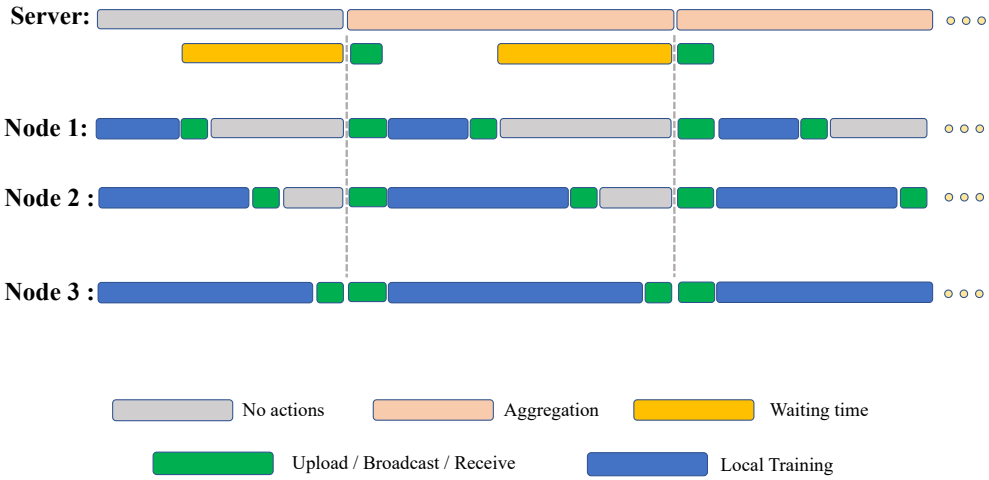

Figure 5. The timing diagram when the aggregation time is $T_g = T_l + T_c + T_w$.

Proposition 6 (Distance Bound) *For ζ_t of virtual sequence and $\omega_t^{(g)}, \omega_t$ of Algorithm 1, we have:*

$$\max\left(\mathbb{E}\left[\|\zeta_t - \omega_t\|^2\right], \mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|)^2\right]\right) \leq 4(\delta-1)^2\gamma^2 G_{\max}^2 \tag{33}$$

Proof:

$$\begin{aligned}
\mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)}\right\|^2\right] &= \mathbb{E}\left[\left\|\sum_{\tau=1}^{t}\gamma\left(\mathbf{G}_{\tau-\delta}^{(j)}\right) - \left(\sum_{\tau=1}^{t}\gamma G_{\tau-1}^{(i)}\right)\right\|^2\right] \\
&\leq \gamma^2\mathbb{E}\left[\left\|\sum_{\tau=1}^{t}\left(\mathbf{G}_{\tau-\delta}^{(i)} - \mathbf{G}_{\tau-1}^{(i)}\right)\right\|^2\right] \\
&\leq \gamma^2(\delta-1)^2\mathbf{G}_{\max}^2.
\end{aligned} \tag{34}$$

Similarly, $\mathbb{E}\left[\left\|\omega_t^{(g)} - \omega_t\right\|^2\right] \leq \gamma^2(\delta-1)^2\mathbf{G}_{\max}^2$. Combining these bounds, we have the following.

$$\begin{aligned}
\mathbb{E}\left[\|\zeta_t - \omega_t\|^2\right] &= \mathbb{E}\left[\left\|\zeta_t - \omega_t^{(g)} + \omega_t^{(g)} - \omega_t\right\|^2\right] \\
&\leq 2\left(\mathbb{E}\left[\left\|\zeta_t - \omega^{(g)}\right\|^2\right] + \mathbb{E}\left[\left\|\omega^{(g)} - \omega_t\right\|^2\right]\right) \\
&\leq 4\gamma^2(\delta-1)^2\mathbf{G}_{\max}^2.
\end{aligned} \tag{35}$$

Theorem 7 *Let $f_{\max} = f\left(\omega_0^{(g)}\right) - f\left(\omega^\star\right)$*, where $\omega^\star$ is the minimizer for f, we have:

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] = O\left(\frac{f_{\max}}{\gamma T} + \gamma^2 L^2 G_{\max}^2\delta^2 + \gamma\frac{L\sigma^2}{N}\right). \tag{36}$$

Proof: Similar to Theorem 4 A.

If $T > N^3$ and $\delta \leq T^{1/4}/N^{3/4}$, we get:

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f\left(\omega_t^{(g)}\right)\right\|^2\right] = O\left(\frac{Lf_{\max}}{\sqrt{NT}} + \frac{\mathbf{G}_{\max}^2}{\sqrt{NT}} + \frac{L\sigma^2}{\sqrt{NT}}\right). \tag{37}$$

B. Special case in synchronous PFL for $T_g = T_l + T_c + T_w$

Here we give a timing diagram for a special case in synchronous PFL when $T_g = T_l + T_c + T_w$. As can be seen in Fig. 5, in this setting, the time spent on each round is T_g.