Final Report : CS 525 - Spring 2021
Asynchronous Federated Learning on Hierarchical Clusters

Yijun Wang
University of Illinois
yijunw4@illinois.edu

Tony Mu
University of Illinois
tongm3@illinois.edu

Albert Yeh
University of Illinois
ayeh2@illinois.edu

Abstract

Federated Learning is a rapidly growing area of research and with various benefits and industry applications. Typical federated patterns have some intrinsic issues such as heavy server traffic, long periods of convergence, and unreliable accuracy. In this paper, we address these issues by proposing asynchronous hierarchical federated learning, in which the central server uses either the network topology or some clustering algorithm to assign clusters for workers (i.e., client devices). In each cluster, a special aggregator device is selected to enable hierarchical learning, leads to efficient communication between server and workers, so that the burden of the server can be significantly reduced. In addition, asynchronous federated learning schema is used to tolerate heterogeneity of the system and achieve fast convergence, i.e., the server aggregates the gradients from the workers weighted by a staleness parameter to update the global model, and regularized stochastic gradient descent is performed in workers, so that the instability of asynchronous learning can be alleviated. We evaluate the proposed algorithm on CIFAR-10 image classification task, the experimental results demonstrate the effectiveness of asynchronous hierarchical federated learning.

1. Introduction

Federated learning is a machine learning methodology where the model updates happen in many clients (mobile devices, silos, etc.), as opposed to a more traditional setting, where training happens on a centralized server. This approach has many benefits, such as avoiding the upkeep of a centralized server, minimizing the network traffic between many clients and servers, as well as keeping sensitive user data anonymous and private.

In 2016, McMahan et al.[2] defined the term Federated Learning: "We term our approach Federated Learning, since the learning task is solved by a loose federation of participating devices (which we refer to as clients) which are coordinated by a central server." For cross-device Federated Learning, the most common architecture pattern is a centralized topology. This topology has been a popular area of research, and has seen much success when applied to commercial products. However, the centralized topology introduces some challenges.

We are learning from the tradeoffs presented by the comparison of certain topologies and help us focus on which tradeoffs we want to measure specifically. In a centralized network topology, a central server is where all the training happens. However, in some scenarios, a central server may not always be desirable; or the central may not be powerful enough [29]. The central server could also become a bottleneck for large amount of network traffic, and large number of connections. This bottleneck is exacerbated when the federated networks are composed of a massive number of devices [17]. Furthermore, current Federated Learning algorithms, such as Federated Averaging, can only efficiently utilize hundreds of devices in each training round, but many more are available [1]

One approach that attempted to address these challenges is decentralized training [2, 17, 27, 29]. However, in large scale FL systems, a fully decentralized FL topology is inefficient, since the convergence time could be long, and the traffic between devices could be too intensive. Well-connected or denser networks encourage faster consensus and give better theoretical convergence rates, which depend on the spectral gap of the network graph. However, when data is IID, sparser topologies do not necessarily hurt the convergence in practice. Denser networks typically incur communication delays which increase with the node degrees [14]. We take a hard look at the gossip learning, one of these state-of-the-art decentralized machine learning protocols, Because this paper identifies the conditions in which gossip learning can and cannot be applied, we take lessons learned in our evaluation of other topologies including gossip learning.

We look into segmented network capacity and focus on utilizing the bandwidth given the restraints of mobile networks. In this paper, we examine how clustered network

topologies could have an effect on several key measures compared to fully decentralized federated learning topology. Some key measures we look at include: client-server network traffic, convergence time, and model accuracy. In the clustered network topology, each client will belong to a cluster. Each cluster will also have a leader, so there is still a notion of a central server. However, the responsibilities of the central server are limited to initiating a new round of training, and talking to leaders to collect updated models. In this topology, clients do not communicate directly with the central server. This should significantly reduce the amount of network ingress incurred at the central server. We are also able to observe some interesting results on the convergence time after adopting the clustered topology.

There has been some existing work [3, 9, 15, 23] on clustered network topology. However, they are mainly focused on model performance and accuracy under heterogeneous data distribution. In this paper, we examine the effect of clustered network topology on the system performance of federated learning systems. Specifically, we dive into the effect on client-server network bandwidth, model convergence time, and model accuracy. We also explore different architectural patterns for clustered federated learning and their effects on the key measurements mentioned above.

The rest of this paper is organized as follows. Section 2 discusses several recently developed federated learning work that are related to ours. In Section 3, we illustrate our proposed algorithm in details, theoretical analysis is shown as well. We conduct experiments and show the evaluation results in Section 4. Finally, we conclude our findings and discuss future work directions in Section 5.

2. Related Work

The most widely used and straightforward algorithm to aggregate the local models is to take the average, proposed in [20] and known as Federated Averaging (*FedAvg*). In FL, the communication cost often dominates the computation cost [20], thus is one of the key issues we need to resolve for implementing FL system at scale. In particular, the state-of-the-art deep learning models are designed to achieve higher prediction performance at the cost of increasing model complexity with millions or even billions of parameters [5, 8]. On the other hand, FL requires frequent communication of the models between the server and workers. As such, *FedAvg* [20] encourages each worker to perform more iterations of local updates before communicating during global aggregation, this results in significantly less communication rounds, and also increases the accuracy eventually as model averaging produces regularization effect. Another way to decrease the communication cost is to reduce the size of model information that needs to be sent, either through model compression techniques such as sparsification [25] and quantization [6], or only select a small portion of important gradients to be communicated [28] based on the observation that most of deep learning model parameters are closed to zero [26]. However, these methods may result in deterioration of model accuracy, or incur high computation cost [18]. Alternatively, [19] proposed client-edge-cloud hierarchical federated learning (*HierFAVG*), an edge computing paradigm in which the edge servers play the roles of intermediate parameter aggregators. The hierarchical FL algorithm leverages on the proximity of edge servers, significantly relieves the burden of the central server on remote cloud.

[31] introduces a hierarchical federated learning protocol through LAN-WAN orchestration, which involves a hierarchical aggregation mechanism in the local-area network (LAN) due to its abundant bandwidth and almost negligible monetary cost than WAN, and incorporates cloud-device aggregation architecture, intra-LAN peer-to-peer (p2p) topology generation, inter-LAN bandwidth capacity heterogeneity. While the hierarchical learning pattern is promising to reduce communication, it is not applicable to all networks, as the physical hierarchy may not exist or be known *a priori* [16].

[22] designs and implements Clustered Federated Learning (CFL) using a cosine-similarity-based clustering method that creates a bi-partitioning to group client devices with the same data generating distribution into the same cluster. Client devices are clustered into different groups according to their properties. It has better performance for the non-IID-severe client network, without accessing the local data. [4] implemented a hierarchical clustering step (FL+HC) to separate clusters of clients by the similarity of their local updates to the global joint model. Once separated, the clusters are trained independently and in parallel on specialised models. In [21], a self-organizing hierarchical structured FL mechanism is implemented based on democratized learning, agglomerative clustering, and hierarchical generalization.

[11] contributes to the field of FL with a comparison of the efficiency of decentralized and centralized solutions that are based on keeping the data local, with an assumption that gossip learning will hurt performance. They recognize many areas of tradeoffs to measure, and that many algorithms can outperform each other depending on what metric they are measuring. However in their testing, they observed that federated learning converges faster, since it can mix information more efficiently and is clearly competitive with centralized federated learning. They believe future work could be improved via applying even more sophisticated peer sampling methods that are optimized based on other tradeoffs. Their experimental scenarios include a real churn trace collected over phones, both continuous and bursty communication patterns, different network sizes and different distributions of the training data over the devices.

They also evaluate a number of additional techniques including a compression technique based on sampling, and token account based flow control for gossip learning. After evaluating the average cost of both approaches, they found that the best gossip variants perform comparably to the best federated learning variants overall. [12] initiated their study with a logical hypothesis is that gossip learning without an aggregation server or a central component will be strictly less efficient than federated learning due its reliance on a more basic infrastructure, just message passing and no cloud resources. They state that, "One of the most challenging problem of federated learning is the poor network connection as the workers are geodistributed and connected with slow WAN." They asked the question if workers can even send full model updates (Size of BERT Large can be up to 1360MB). They ask if "is it possible for workers to synchronize the model partially, from/to only a part of the workers, and still achieve good training results?" They explore this area with a decentralized FL solution, called Combo. Knowing that peer-to-peer bandwidth is much smaller than the worker's maximum network capacity, their program could fully utilize the bandwidth by saturating the network with segmented gossip aggregation. Their experiments end up showing that they can reduce the training time significantly with great convergence performance. [13] focus on the the design choices for a sparse model averaging strategy in a decentralized parallel SGD. Their attempt to design an optimal communication topology that is both quick and efficient. They propose a superpeer topology where they form a ring and have some number of regular peers connected to them. The hierarchical two-layer sparse communication topology allows a principled trade-off between convergence speed and communication overhead and is well suited to loosely coupled distributed systems. We demonstrate this using an image classification task and a batch stochastic gradient descent learning (SGD) algorithm that their proposed method shows similar convergence behavior as Allreduce while having lower communication costs. Giaretta and Girdzijauskas [10] examine the conditions in which gossip learning can and cannot be applied. Our team takes a hard look at the extensions that their research has mentioned to mitigate some of the limitations in FL. They present a thorough analysis of the applicability of gossip learning, Their work includes scenarios that range from the effect of certain topologies, and the correlation of communication speed and data distribution. Although tested on Industrial IoT (IioT) setting, Savazzi et al. [24] study a handful of gossip based decentralized machine learning methods in the context of (IioT) apps. Their focus is on when the data distribution is not identical over the nodes (similar to privacy data on cell phones. They do not consider compression techniques or other algorithmic enhancements such as token-based flow control. Their paper proposes a serverless learning approach, where the proposed FL algorithms use device to device cooperation to perform data operations on the network by iterating local computations using consensus-based methods.

Most of the current FL systems are implemented using synchronous update, which is susceptible to the straggler effect. To address this problem, [30] proposed an asynchronous algorithm for federated optimization, *FedAsync*, which solves regularized local problems and then uses a weighted average with respect to the staleness to update the global model. [7] presented *ASO-Fed* for asynchronous online FL, which uses the same surrogate objective for local updates, while the local learning rate is adaptive to the average time cost of past iterations.

3. Approach

3.1. Problem Formulation

We consider the supervised federated learning problem which involves learning a single global statistical model owned by the central server, while each of N devices owns a private dataset and works on training a local model. Let $\mathbf{w}$ parameterize the model, and $\mathcal{D}^i = \{\mathbf{x}_j, y_j\}$ denote the training dataset owned by i-th device, where $i \in \{1, \cdots, N\}$, $\mathbf{x}_j$ is the j-th input sample from $\mathcal{D}^i$, while y_j is the corresponding label. Denote $\ell(\mathbf{x}_j, y_j|\mathbf{w})$ as the loss function presents the prediction error, our overall goal is to minimize the empirical loss $\mathcal{L}(\mathbf{w})$ over all distributed training data $\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}^i$, i.e., we aim at solving the following optimization problem:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{\sum_{i=1}^{n} \sum_{j \in \mathcal{D}^i} \ell(\mathbf{x}_j, y_j|\mathbf{w})}{|\mathcal{D}|}. \tag{1}$$

The problem is often solved by mini-batch stochastic gradient descent (SGD), in which during each step, the model is updated as

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{w}}, \tag{2}$$

where α denotes the learning rate, and the average gradient

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{1}{m} \sum_{j \in B} \frac{\partial \ell_j}{\partial \mathbf{w}} \tag{3}$$

is derived through back-propagation from the mini-batch B of m input samples. In the typical FL setting, each device i performs SGD with data sampled from its own private training dataset $\mathcal{D}^i$ and train a local model

$$\mathbf{w}_i = \arg\min_{\mathbf{w}_i} \mathcal{L}_i(\mathbf{w}_i) = \frac{\sum_{j \in \mathcal{D}^i} \ell(\mathbf{x}_j, y_j|\mathbf{w}_i)}{|\mathcal{D}^i|}, \tag{4}$$

the server aggregates all local models collected from the workers and update the global model which is then sent back to the workers for next iteration.

3.2. Proposed Method

3.2.1 Initialization at Central Server

We consider a hierarchcal FL system which has one central server on the cloud. The central server owns the global model $\mathbf{w}$ and denotes the timestamp t for the model parameters. Therefore, as the learning begins, the central server initializes the global model parameters $\mathbf{w}$, its timestamp $t = 0$, as well as several hyperparameters that are required by learning. In addition, given the network information of all client devices, we allow the central server to be responsible for the knowledge of the hierarchical communication topology. In the case of mobile edge computing, the partition and hierarchy can be naturally formed by the communication edges, as the links between the central server with the edge servers or base stations form a star topology, and so do the links between each edge server with the devices. We extend this architecture to a more general case by allowing the central server to run clustering algorithm to assign which cluster each device belongs to, as well as a special device in each cluster, which we denote as the "*aggregator*", that plays the role of an edge server, that is, provides inter-hierarchy communication including downlink transmission from the central server and client devices and uplink transmission from the clients to the central server, also aggregates information to reduce the necessary communication. In FL, this aggregation work is specific to aggregating of the clients' updated weights/gradients, and sending them to the central server. The downlink transmission is straightforward: the server periodically sends the global model with timestamp as well as the hyperparameters for the learning task to the aggregators, and the aggregator serves as the parent node of the client devices in each cluster, forwards the information it receives from the central server to its children nodes.

3.2.2 Learning on Local Clients

Upon receiving the global model parameters $\mathbf{w}$ with its timestamp (according to the central server clock) from the central server, the worker client performs local update. In order to mitigate the deviations of the local models on an arbitrary device j from that of the central server, following *FedAsync* [30], instead of minimization of the original local loss function ℓ_j, client j locally solves a regularized optimization problem, i.e., performs SGD update for one or multiple iterations on the following surrogate objective:

$$\min_{\mathbf{w}_j} g_j(\mathbf{w}_j) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}^j}\left[\ell_j(\mathbf{w}_j) + \frac{\lambda}{2}||\mathbf{w}_j - \mathbf{w}||^2\right], \quad (5)$$

in which the regularization term $\frac{\lambda}{2}||\mathbf{w}_j - \mathbf{w}||^2$ controls the deviation of the local models. After local learning, the client sends its updated parameters as well as the original model timestamp to its parent node, the corresponding aggregator. The next local learning iteration will based on the newest received global model and the corresponding timestamp.

3.2.3 Learning on Cluster Aggregators

On an aggregator, the rate of receiving updates from the clients may vary caused by several reasons, such as heterogeneity of the computation power among devices, network delay, etc. We propose to perform asynchronous federated learning, that is, the aggregator immediately aggregates the update from the clients and reports to the central server. In real implementation, we can use a thread-safe FIFO queue to store the updates from the clients inside each aggregator, and periodically aggregates the results in the queue without waiting for that from some potential stragglers. This is different from the synchronous FL paradigm, and the uplink communication is then non-blocking. Again, following *FedAsync* [30], we use a function of staleness to mitigate the error caused by obsolete models. Intuitively, more staleness results in larger error. On an aggregator device, assume the latest global model it received was with timestamp t' (according to central server clock) at the moment it is about to aggregate the updates, and the local model from client was with timestamp t, then it must be true that $t' \geq t$. We modify the learning rate to be weighted by the staleness:

$$\alpha_{t'} = \alpha \times \sigma(t' - t), \quad (6)$$

in which $\sigma(z)$ is the staleness function. Different forms of $\sigma(z)$ were defined in [30], such as:

- the polynomial form:

$$\sigma(t' - t) = (t' - t + 1)^{-\beta}, \quad (7)$$

- the hinge form:

$$\sigma(t'-t) = \begin{cases} 1 & \text{if } t' - t \leq b \\ \dfrac{1}{a(t' - t - b) + 1} & \text{otherwise} \end{cases} \quad (8)$$

Note that $\sigma(t' - t) = 1$ if $t' = t$, and monotonically decreases as t' and t deviates more, so that the obsolete update would affect the model less as it shrinks the learning rate. Therefore, upon receiving local update from client device j, the model can then be updated on the cluster aggregator k as:

$$\mathbf{w}_k^{(t')} \leftarrow (1 - \alpha_{t'})\mathbf{w}^{(t')} + \alpha_{t'}\mathbf{w}_j^{(t)}. \quad (9)$$

Or equivalently, if we aggregate the gradients collected from the clients, we have:

$$\mathbf{dw}_k^{(t')} = \sum \alpha\sigma(t' - t)\mathbf{dw}_j^{(t)}, \quad (10)$$

where $\mathbf{dw}_j^{(t)}$ is the gradient collected by device j in k-th cluster.

3.2.4 Learning on Central Server

The central server aggregates the results from the cluster aggregators to update the global model. Similar to the learning procedure on the cluster aggregators, in the central server, we can use a queue to store the updates from the aggregators. As asynchronous learning, the numbers of updates gathered from each of the aggregators can be imbalanced. As such, we let the aggregator k send the number of updates n_k along with the aggregated results (i.e., $\mathbf{w}_k^{(t')}$ and timestamp t') to the central server. We assume the newest global model was updated at timestamp t'' according to the central server clock, and that it must hold that $t'' > t'$. Combine the update counts information and the staleness schema, by collecting the update from aggregator k, the learning rate is weighted and modified as

$$\alpha_{t''} = \frac{n_k}{N}\sigma(t'' - t')\alpha,$$

where N is the total number of devices in the FL system which is known to the central server. Note that this weighting mechanism makes sense if the data are i.i.d. over the clients. However, the bias could be severe if non-i.i.d. data are involved, as the mechanism favors learning for faster computed and communicated devices, in which case we need to carefully tune and design a more complicated weighting mechanism. The central server updates the global model as follows:

$$\mathbf{w}^{(t''+1)} \leftarrow (1 - \alpha_{t''})\mathbf{w}^{(t'')} + \alpha_{t''}\mathbf{w}_k^{(t')}, \qquad (11)$$

The detailed algorithm is illustrated in Algorithm 1.

3.3. Convergence Analysis

Definition 1 (Lipschitz Smoothness)**.** A function f is Lipschitz smooth with constant $L > 0$ if its derivatives are Lipschitz continuous with L, i.e., $\forall x, y$,

$$||\nabla f(y) - \nabla f(x)|| \leq L||y - x||,$$

in other words, we have

$$f(y) - f(x) \leq \langle \nabla f(x), y - x\rangle + \frac{L}{2}||y - x||^2.$$

Definition 2 (Strong Convexity)**.** A differentiable function f is μ-strongly convex with constant $\mu > 0$ if $\forall x, y$,

$$f(y) - f(x) \geq \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}||y - x||^2,$$

which is equivalent to the function $g(x) = f(x) - \frac{\mu}{2}||x||^2$ is convex for all x, while the latter is further equivalent to $\nabla^2 g \succeq 0$, which is $\nabla^2 f \succeq \mu I$.

Assumption 1. The global loss function $\mathcal{L}$ is L-smooth, μ-strongly convex, and bounded from below.

Algorithm 1 Asynchronous Hierarchical Federated Learning (FedAH)

Central Server:
Assign clusters and aggregators according to network topology or by running clustering algorithm.
Initialize global model $\mathbf{w}$ and time clock t.
for $t = 0, \cdots, T-1$ **until** end of learning **do**
  Broadcast $(\mathbf{w}, t)$ to its direct children aggregators.
  Receive triples $\left(\mathbf{dw}_k^{(t')}, t', n_k\right)$ from any direct child aggregator k.
  Update global model $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \alpha_t \mathbf{dw}_k^{(t')}$ where $\alpha_t = \alpha\sigma(t - t')n_k/N$.
end for
Middle Layer Aggregator:
Receive $\left(\mathbf{w}, t''\right)$ from its parent, broadcast to its direct children.
Receive $\left(\mathbf{w}_j^{(t')}, t'\right)$ from any of its direct child j.
Aggregate the collected gradients: $\mathbf{dw}_k^{(t')} = \sum \alpha\sigma(t'' - t')\mathbf{dw}_j^{(t)}$.
Send triples $\left(\mathbf{dw}_k^{(t')}, t', n_k\right)$ to its parent.
Bottom Layer Client Device:
Receive $\left(\mathbf{w}, t''\right)$ from its parent.
Define $g_j(\mathbf{w}_j) = \ell_j(\mathbf{w}_j) + \frac{\lambda}{2}||\mathbf{w}_j - \mathbf{w}||^2$
for local iteration **do**
  Randomly sample $(\mathbf{x}, y) \sim \mathcal{D}^i$
  Local update $\mathbf{w}_j \leftarrow \mathbf{w}_j - \alpha\nabla g_j$
end for
Send updated model $\left(\mathbf{w}_j, t''\right)$ to its parent.

Assumption 2. We assume bounded delay of communication and bounded processing time in the system.

Lemma 1 (Asymptotic optimality)**.** Suppose Assumption 1 and 2 hold, after T global updates on the server, Algorithm 1 converges to a critical point.

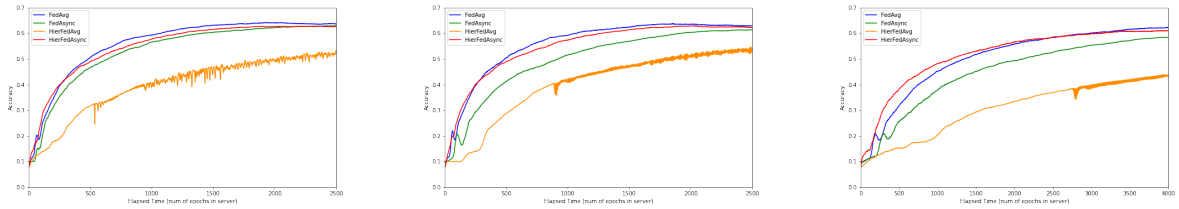
Lemma 2 (Convergence rate)**.**

$$\frac{\mathcal{L}(\mathbf{w}^{(t+1)}) - \mathcal{L}(\mathbf{w}^*)}{\mathcal{L}(\mathbf{w}^{(t)}) - \mathcal{L}(\mathbf{w}^*)} \leq (1 - \delta)^c$$

We will derive the values of c and δ as well as the proofs later.

4. Experiments and Results

In our experiments, we consider an asynchronous hierarchical FL system with N_c client devices, N_a cluster aggregators, and a single central server. The non-server machines are grouped into N_a clusters using hierarchical agglomerate cluster based on their IP addresses, while in reality the computational power and network conditions can be integrated as the clustering features as well. For simplicity purpose, the aggregators in each cluster are randomly selected. We conduct our initial experiments on a standard image classification task, the famous CIFAR-10 dataset were used. We set up the model as a convolutional neural network (CNN) with 3 convolutional blocks, which has 5852170 parameters and achieves 90% test accuracy in centralized training.



(a) 10 client devices, with 2 cluster aggregators for hierarchical learning.

(b) 20 client devices, with 4 cluster aggregators for hierarchical learning.

(c) 50 client devices, with 5 cluster aggregators for hierarchical learning.

Figure 1: Comparison of test accuracy w.r.t. training clock in central server.

For our FL system, we randomly partition the CIFAR-10 dataset among the N_c local learning devices, so that each of the 10 class labels are kinds of balanced distributed over all clients. For local training, SGD optimizer are employed with a batch size of 128 and an initial learning rate of 0.001.

Our models and learning procedures are implemented using PyTorch, and we compare the performance of several different algorithms with our proposed approach. Figure 1 shows the comparison of the global models' accuracy evaluated on central servers' validation dataset verses the training time. In *FedAvg* and *FedAsync*, the worker devices directly communicate to the central server without hierarchical structure, while *FedAsync* allows the server and workers to update the models at any time without synchronization. Our hierarchical learning involves the simplest Client-Aggregator-Server 3-layer hierarchical structure, where *HierFedAvg* perform synchronous learning, while *HierFedAsync* follows the learning schema we described in Section 3.2.

In each setting, we let the learning last for 2500 learning epochs with a clock in the central server. Specifically, each learning epoch is synchronized among all devices. We simulate asynchronous learning system by assuming the fault (e.g. device down, communication loss, straggler effect, etc.) uniformly distributed with probability 0.1 among all non-server devices for each learning epoch. This probability distribution can be further investigated by tuning the parameters and looking at empirical studies.

According to Figure 1, with hierarchical settings, the complexity of learning system is greatly increased, as a result, the *HierFedAvg* algorithm not only converges the slowest, the learning is also not stable. Conversely, it is obvious that our designed *HierFedAsync* algorithm overcomes the issues brought by the hierarchical setting. Although *FedAvg* seems to perform the best when the number of client devices is small (e.g., when there are 10 clients in the system), we would like to emphasize that we did not count the fault device nor the stragglers' effect in the synchronous settings for our experiments, while those effects are included in our asynchronous settings. Even so, our *HierFedAsync* algorithm performs close to the best in all cases, and when the system gets larger, the advantages of *HierFedAsync* gets more obvious, not only shows faster convergence especially early on, also leads to higher test accuracy. And we expect further that as the number of devices gets larger, the advantage gets bigger.

Table 1: Comparison of the numbers of gradients sent/received, with 20 client devices in the system and 2500 training epochs in the central server.

	FedAvg	*FedAsync*	*HierFedAvg*	*HierFedAsync*
Central Server	50000	44769	10000	8842
Cluster Aggregators	-	-	60000	52904
Local Clients	50000	45033	50000	44977

Table 1 presents the total numbers of gradients sent or received by each type of devices, from which we can see that the communication burden of the central server would be greatly alleviated in a hierarchical topology, not to mention the potential benefits of more local computation and faster overall convergence. In a large system of network topology, this could also leads to less packet loss, more effective communication and computation.

Our *HierFedAsync* algorithm involves several hyperparameters to tune. We conduct comparative analysis with different values of β in the polynomial form of staleness function as described in Equation (7), and show the effect of staleness on learning convergence in Figure 2. We see that when $\beta = 1$, the learning curve is closed to that without introducing staleness (i.e. $\beta = 0$), but the validation accuracy cannot exceed 55% as training proceeds, moreover, we notice the learning is unstable as the curve oscillates severely. In general, larger staleness alleviates the instability, at the cost of slower convergence. From Figure 2, we can easily see that by using $\beta = 2$ or 3, the performance is significantly improved as the validation accuracy is higher than 60% after convergence, as the convergence rate is also very acceptable. We also note that the learning effect is not sensitive with β as it is between 2 and 3, indicates that β is quite easy to tune. Similar comparative

analysis is conducted for the regularization coefficient λ in Equation (5) on local clients. Although results in Figure shows little effect for change of λ, we would like to note that our current experimental setting does not emphasize on simulation of the stragglers. We expect that the regularization on local clients plays a more important role during training in an asynchronous system as the straggler effect gets more common.

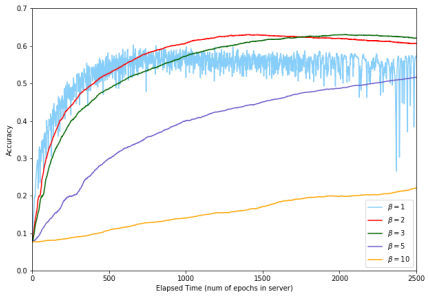


Figure 2: Test accuracy with different β values in *HierFedAsync* with 20 client devices and 2500 training epochs in the central server. The polynomial staleness function $\sigma(t'-t) = (t'-t+1)^{-\beta}$ is used on central server as well as cluster aggregators.

5. Conclusions and Future Work

In this paper, we propose asynchronous hierarchical federated learning. As federated networks are composed of a massive number of devices, communication is a critical challenge in FL. We tackle this problem by exploring different architectural patterns for the design of FL systems. The tradeoff of central and fully decentralized learning on the complexity of the system as well as the learning effectiveness, computational and communication cost is obvious. We deploy a FL system with a central server, but with hierarchical topology. In this paper, we combine asynchronous FL and hierarchical FL into our *FedAH* algorithm. In addition, we blur the concept of network topological edges to form clusters as well as the hierarchical structures. We aim at reducing the communication load between devices and the server in FL system, also improve flexibility and scalability. Our initial experiments demonstrated that combining asynchronous FL and hierarchical FL not only leads to faster convergence, tolerates heterogeneity of the system such as the faulty devices, straggler effect, etc., also significantly alleviates the communication burden on the central server. However, the asynchronous and hierarchical nature greatly increases the complexity of the system, especially on the communication topology, which could lead to unstable learning. We explored the literature and recent research advances, combine them into our proposed method, *FedAH*, which inherits the merits of both asynchronous and hierarchical FL, meanwhile significantly mitigates the instability with the utilization of staleness function and cluster weighting on the central server and edge devices, as well as adding regularization for local updates on client devices. We implemented the system and evaluated on CIFAR-10 image classification task, the results verify the effectiveness of our design and meet our expectation. There are several interesting directions to pursue for the future of this paper. First off, as we mentioned in the paper, our weighting mechanism for aggregation favors learning for faster computed and communicated devices, which works fine if the data are i.i.d. over the clients. We design a more sophisticated weighting mechanism for asynchronous FL if non-i.i.d. data are involved. The second interesting direction in which to take this paper would be modification of the simple L^2-regularization on local clients' learning. We also need to finish the derivation and proof of the theoretical analysis. In addition, more experiments need to be conducted on different datasets and tasks. Furthermore, we need to bring more sophisticated experimental settings for the simulation, for instance, the straggler effect should be emphasized.

References

[1] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards Federated Learning at Scale: System Design. *arXiv e-prints*, art. arXiv:1902.01046, Feb. 2019. 1

[2] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv e-prints*, art. arXiv:1602.05629, Feb. 2016. 1

[3] C. Briggs, Z. Fan, and P. Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020. doi: 10.1109/IJCNN48605.2020.9207469. 2

[4] C. Briggs, Z. Fan, and P. Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020. 2

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 2

[6] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018. 2

[7] Y. Chen, Y. Ning, and H. Rangwala. Asynchronous online federated learning for edge devices. *arXiv preprint arXiv:1911.02134*, 2019. 3

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[9] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran. Robust Federated Learning in a Heterogeneous Environment. *arXiv e-prints*, art. arXiv:1906.06629, June 2019. 2

[10] L. Giaretta and Š. Girdzijauskas. Gossip learning: Off the beaten path. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1117–1124. IEEE, 2019. 3

[11] I. Hegedűs, G. Danner, and M. Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021. 2

[12] C. Hu, J. Jiang, and Z. Wang. Decentralized federated learning: a segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019. 3

[13] M. Jameel, J. Grabocka, L. Schmidt-Thieme, et al. Ringstar: A sparse topology for faster model averaging in decentralized parallel sgd. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 333–341. Springer, 2019. 3

[14] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. Nitin Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. Theertha Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and Open Problems in Federated Learning. *arXiv e-prints*, art. arXiv:1912.04977, Dec. 2019. 1

[15] L. U. Khan, M. Alsenwi, Z. Han, and C. S. Hong. Self organizing federated learning over wireless networks: A socially aware clustering approach. In *2020 International Conference on Information Networking (ICOIN)*, pages 453–458, 2020. doi: 10.1109/ICOIN48656.2020.9016505. 2

[16] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020. 2

[17] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. *arXiv e-prints*, art. arXiv:1705.09056, May 2017. 1

[18] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020. 2

[19] L. Liu, J. Zhang, S. Song, and K. B. Letaief. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020. 2

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017. 2

[21] M. N. Nguyen, S. R. Pandey, T. N. Dang, E.-N. Huh, C. S. Hong, N. H. Tran, and W. Saad. Self-organizing democratized learning: Towards large-scale distributed learning systems. *arXiv preprint arXiv:2007.03278*, 2020. 2

[22] F. Sattler, K.-R. Müller, and W. Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 2

[23] F. Sattler, K. R. Müller, and W. Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2020. doi: 10.1109/TNNLS.2020.3015958. 2

[24] S. Savazzi, M. Nicoli, and V. Rampa. Federated learning with cooperating devices: A consensus approach for massive iot networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020. 3

[25] S. U. Stich, J.-B. Cordonnier, and M. Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018. 2

[26] N. Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. 2

[27] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu. D^2: Decentralized Training over Decentralized Data. *arXiv e-prints*, art. arXiv:1803.07068, Mar. 2018. 1

[28] Z. Tao and Q. Li. esgd: Communication efficient distributed deep learning on the edge. In {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018. 2

[29] P. Vanhaesebrouck, A. Bellet, and M. Tommasi. Decentralized Collaborative Learning of Personalized Models over Networks. *arXiv e-prints*, art. arXiv:1610.05202, Oct. 2016. 1

[30] C. Xie, S. Koyejo, and I. Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019. 3, 4

[31] J. Yuan, M. Xu, X. Ma, A. Zhou, X. Liu, and S. Wang. Hierarchical federated learning through lan-wan orchestration. *arXiv preprint arXiv:2010.11612*, 2020. 2