Hierarchical Federated Learning Architectures for the Metaverse

GU Cheng, LI Baochun
(Department of Electrical and Computer Engineering,University of Toronto, Toronto M5S2E8, Canada)

DOI: 10.12142/ZTECOM.202402006
https://kns.cnki.net/kcms/detail/34.1294.TN.20240624.1615.002.html, published online June 24, 2024
Manuscript received: 2024-05-31

Abstract: In the context of edge computing environments in general and the metaverse in particular, federated learning (FL) has emerged as a distributed machine learning paradigm that allows multiple users to collaborate on training a shared machine learning model locally, eliminating the need for uploading raw data to a central server. It is perhaps the only training paradigm that preserves the privacy of user data, which is essential for computing environments as personal as the metaverse. However, the original FL architecture proposed is not scalable to a large number of user devices in the metaverse community. To mitigate this problem, hierarchical federated learning (HFL) has been introduced as a general distributed learning paradigm, inspiring a number of research works. In this paper, we present several types of HFL architectures, with a special focus on the three-layer client-edge-cloud HFL architecture, which is most pertinent to the metaverse due to its delay-sensitive nature. We also examine works that take advantage of the natural layered organization of three-layer client-edge-cloud HFL to tackle some of the most challenging problems in FL within the metaverse. Finally, we outline some future research directions of HFL in the metaverse.

Keywords: federated learning; hierarchical federated learning; metaverse

Citation (Format 1): GU C, LI B C. Hierarchical federated learning architectures for the metaverse [J]. ZTE Communications, 2024, 22 (2): 39 - 48. DOI: 10.12142/ZTECOM.202402006
Citation (Format 2): C. Gu and B. C. Li, "Hierarchical federated learning architectures for the metaverse [J]. *ZTE Communications*, vol. 22, no. 2, pp. 39 - 48, Jun. 2024. doi: 10.12142/ZTECOM.202402006.

1 Introduction

With the advent of spatial computing and head-mounted devices such as the Apple Vision Pro, the metaverse computing environments have quickly become a reality and will become an everyday routine in the future. On the other hand, with the breakneck speed at which both computation power and efficiency in deep learning have advanced in the past decade due to its dramatic success, we almost always need to rely upon the power of deep learning models in any distributed computing environments, and in the metaverse in particular. One major bottleneck in using deep learning in the metaverse today, however, is the availability of raw data that we can use to train a new model or fine-tune a pretrained model.

One of the key contributing factors to the high popularity of deep learning lies in its ability to enable the data-driven paradigm in algorithm design. Unlike the traditional algorithm design paradigm, which relies on human expertise to produce hand-crafted logic and heuristics, the data-driven paradigm instead focuses on training a model, often as a black box, to learn features from a large dataset in order to produce desired outputs. This method excels in tasks where the desired function is too complex to understand or to construct manually; however, at the same time, it often requires large volumes of training datasets to work effectively, especially when in metaverse computing environments. This becomes a significant issue in traditional centralized deep learning which requires a server to access data directly, yet the data may contain highly sensitive or private information that makes it difficult to do so.

Federated learning (FL) is one of the methods that has emerged in recent years that aims to solve this exact problem. The term was first introduced in 2017[1] to describe a distributed machine learning paradigm that typically involves a group of clients and a central server, where each client trains a deep learning model with its local data, and then uploads the model parameters to the central server to aggregate into a single global model. This allows the clients to retain sensitive data locally, while still contributing to the global model by uploading its local model parameters via each round of communication with the server. The intrinsic protection of privacy offered by this architecture has earned

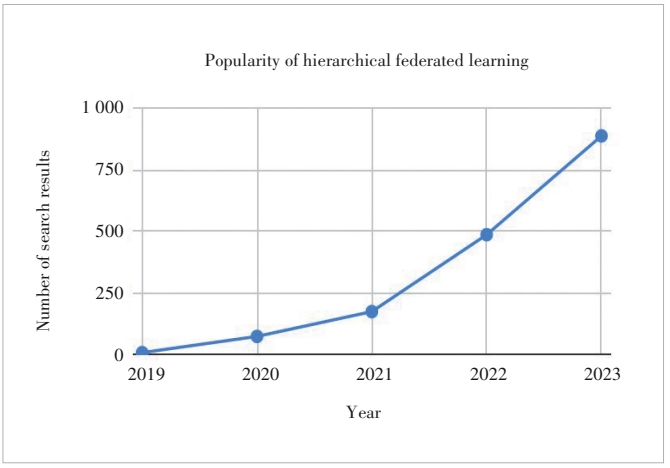
immense popularity in the past few years.

However, the original two-layer client-server architecture for federated learning is hardly scalable in the metaverse, where scalability is needed the most. One of the key reasons why the original architecture is not suitable for the metaverse environments is the high bandwidth demand in each communication round between the server and clients. As the complexity and size of local models increase, the number of data each client needs to send to the server also increases proportionally. Some popular modern models like large language models often have from millions to billions of parameters, making the bandwidth consumption of each communication round extremely high. The most apparent effect of this is an increased communication time between the server and the clients. Sometimes the communication time becomes significant enough to warrant a reduction in total communication rounds by increasing the number of local training rounds. However, this can result in a variety of negative effects, including an increase in the total number of rounds for the global model to converge or a drop in the final model's accuracy. Moreover, these negative effects often get amplified when the data are non-independently and identically distributed (non-IID), which is common in real-world data[2].

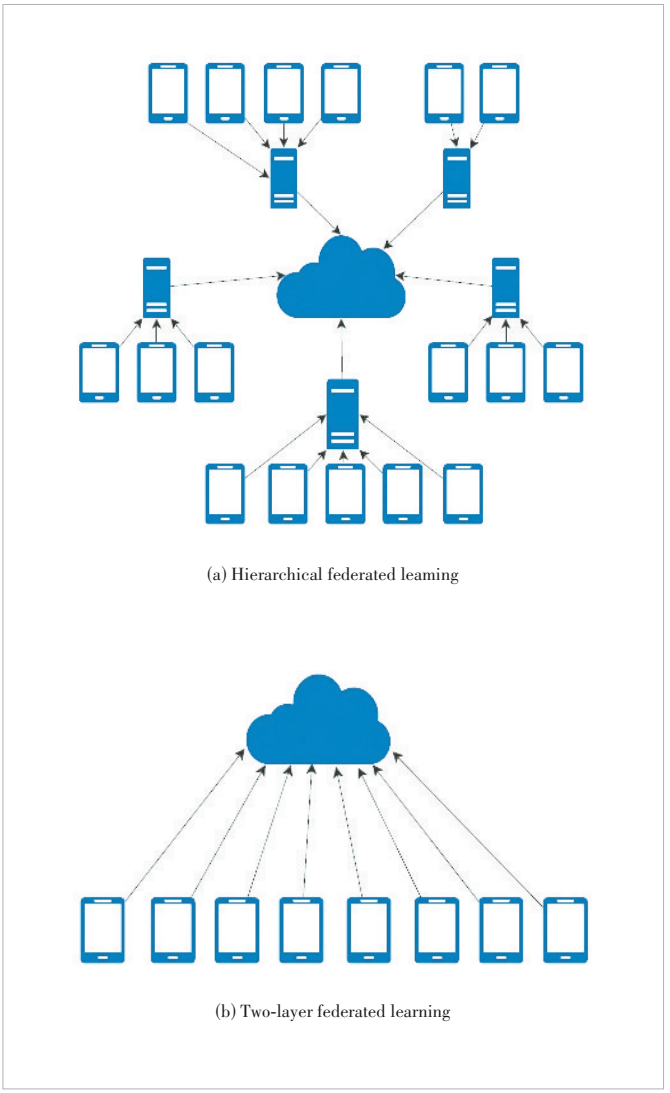
A common approach to increasing scalability in distributed systems in general and the metaverse in particular is to add edge servers, especially when the central server needs to serve a large number of clients. This type of hierarchical organization pattern can be very frequently observed in distributed computing architecture design, such as edge computing systems[3] and software-defined networks (SDN)[4]. We argue that it should be the preferred paradigm of distributed learning in the metaverse computing environment as well.

Driven by this intuition, hierarchical federated learning (HFL) was introduced around late 2019 to early 2020[5 - 6], along with a modified version of a federated averaging (FedAvg) algorithm, called the hierarchical stochastic gradient descent (HSGD) algorithm. Since then, HFL has gained significant popularity, as shown in Fig. 1. In a three-layer HFL, each edge server is assigned to a group of clients local to its service area. The clients only communicate with their assigned edge server, which aggregates the client parameter updates to an edge-local model. Each edge server then sends the aggregated edge-local model to the central cloud server, which further aggregates them to a single global model. Fig. 2 shows the architecture of the three-layer HFL versus traditional two-layer federated learning. Since the original introduction, a number of research works[6 - 13] have emerged to further build on the broad concept of HFL.

However, not all works agree on a three-layer hierarchical structure. Refs. [14 - 16] argue that using a cloud server exposes the system to a single point of failure; instead, they



▲ **Figure 1. Approximate numbers of publications with "hierarchical federated learning" in the title, with results obtained through Google Scholar Advanced Search API and may count duplicated publications**



▲ **Figure 2. Architecture of three-layer hierarchical federated learning versus two-layer traditional federated learning**

propose a two-layer system with a group of edge servers communicating with each other. Refs.[17 – 18] argue that three layers might not be enough in FL and instead propose a multi-layered HFL architecture. In this paper, we examine all of those variations and propose that the three-layer architecture is typically the best choice in most scenarios, including the metaverse, compared with other alternatives. In addition, besides scalability, we will also examine some other intrinsic advantages of HFL over traditional federated learning in the metaverse. Ref. [10] uses knowledge distillation and meta-learning to take advantage of the naturally clustered clients in HFL to improve the training performance on non-IID training data problems. Other researchers propose to capitalize on edge computing power to apply data pruning and quantization mechanisms[7].

The overarching objective of this paper is to provide a general overview of the current landscape of HFL in edge computing environments in general and the metaverse computing environments in particular, and to examine the different architectures for achieving better scalability with HFL. We will also provide some insights on why HFL has the potential to become the mainstream framework for next-generation federated learning systems in the metaverse and outline potential research directions.

2 Background

2.1 Federated Averaging Algorithm

The term federated learning was first introduced by MCMAHAN et al. in 2017[1], along with the Federated Averaging (FedAvg) algorithm which would become the foundation for the majority of federated learning algorithms in the years that ensue. The intrinsic privacy protection offered by FL's nature of not requiring end devices to directly upload data has incentivized heavy research investments and has inspired a large number of works, including the HFL. We begin by examining the FedAvg algorithm to build a foundation for later discussions on the hierarchical stochastic gradient descent algorithm.

The FedAvg algorithm can be defined as the following. Let us assume that we have a training dataset $D = \left\{x_i\right\}_{i=1}^{|D|}$ that is divided among K clients, each owning a subset of the training data $D_k \in D$. Each client k also owns a local model denoted as f_k, which is fully parameterized as a set of weights w_k. We can calculate the loss value of the local model with a loss function l_k.

Training is performed in a total of T rounds. In each round, the FedAvg algorithm can be divided into the local training stage and the global aggregation stage. During round t, each client k performs m local iterations to minimize some average local loss:

$$\min_{w} L_k(w_k^t, D_k),$$
$$L_k(w_k^t, D_k) = \frac{\sum_{i=1}^{D_k} l_k(x_i, w_k^t)}{|D_k|}. \tag{1}$$

Once the local update is completed, the clients send the weights w_k^t to the central server for an average aggregation via the following equation:

$$w^{t+1} = \frac{\sum_{k=1}^{K} |D_k| w_k^t}{|D|}. \tag{2}$$

After aggregation, the global server then broadcasts the model to all clients to be used for the next round of training. The algorithm terminates after T rounds.

2.2 Definition of Hierarchical Federated Learning

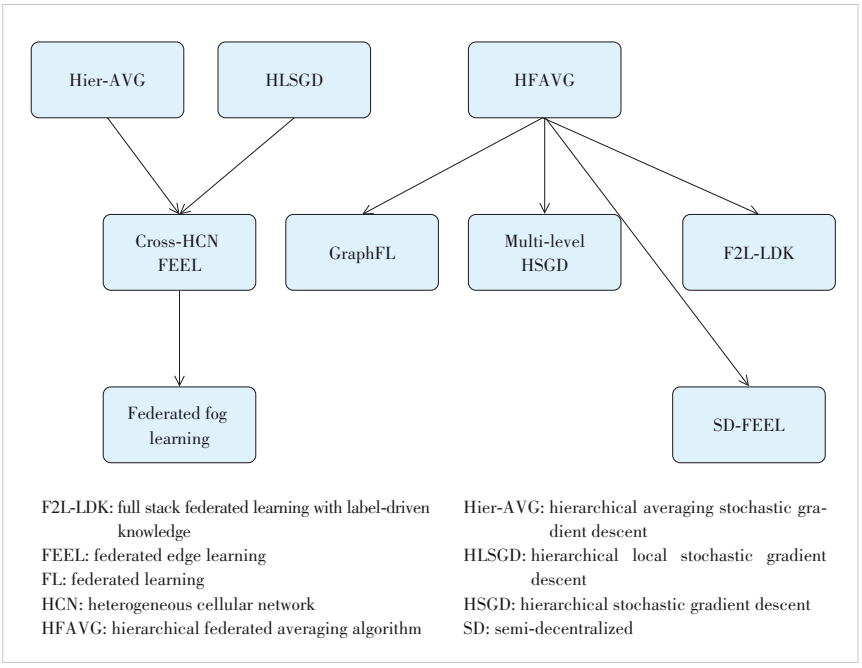
Before we start, it is necessary to define the exact scope of our discussion. To do so, we must first find a suitable definition for what is HFL. However, to our best knowledge, despite the growing popularity of the topic, there is no existing work that adequately defines what HFL is. Fortunately, the nomenclature of the term itself is fairly self-explanatory and provides a good foundation on which we can easily build our own definition. To begin with, as the name indicates, HFL is a subset of FL. Hence, it inherits the same set of definitions. Same as FL, HFL is also a distributed machine learning paradigm that involves multiple data owners and a model owner to train a single model, whereby the training data are kept strictly private to each data owner. Typically, in traditional FL, each data owner trains its own local model and sends the update to a single model owner entity, which aggregates all updates to update the target global model. An example would be the FedAvg algorithm discussed in the previous section. It is worth noting that, from a system organization perspective, this direct communication between data owners and model owners naturally partitions the system into two layers.

Instead of directly aggregating all updates from the data owners into a global model update, HFL employs multiple intermediate model aggregators to first aggregate the client updates into intermediate updates, and then aggregate these intermediate updates into the final global update. From a system organization perspective, these intermediate model aggregators typically form one or more extra layers in the system and introduce a hierarchy to the communication structure, which is responsible for the "hierarchical" part in HFL. An example of this would be the Hierarchical Stochastic Gradient Descent algorithm introduced in Ref. [5].

To summarize, HFL can be broadly defined as a subset of FL that consists of multiple intermediate aggregators between the data owners and the global server.

3 Progenitors of Hierarchical Federated Learning

Although hierarchical federated learning has only recently started to gain popularity, the concept was first introduced between 2018 and 2019. Specifically, Refs. [5 – 6, 19 – 20] contributed to building the foundation for the HFL field, as shown in the dependency graph in Fig. 3. We will discuss these works in this section. A comparison between these works and the rest of the works discussed in this paper can be found in Table 1.



▲ **Figure 3. Citation graph for all the HFL architectures discussed in this paper. An arrow pointing from work A to work B means that A appears in B's citations**

1) Hierarchical local stochastic gradient descent (HLSGD) and hierarchical averaging stochastic gradient descent (Hier-AVG). The very first mentioning of an algorithm that can be classified as an example of HFL can be found in Ref. [19] by LIN et al., which was first available as a preprint in 2018 and subsequently published in 2020. The focus of the paper was comparing the performance of FedAvg (referred to as local-SGD in the paper) against traditional SGD with large mini-batches, and HLSGD was only briefly introduced in the appendix.

A simplified version of the HLSGD algorithm is shown in Algorithm 1, where we can observe that the algorithm follows our definition of HFL well. Each node in a graphic processing unit (GPU) block is a data owner, as local data are not shared between any two nodes horizontally and are not passed to outer loops. On top of the data owners, each "block" serves as an intermediate model aggregator, since each of them hosts a local-SGD or the FedAvg algorithm among the GPU nodes on the block. Finally, a global model is obtained by combining the models from the blocks. This basic anatomy of the algorithm can be observed across almost all HFL implementations.

The authors in Ref. [20] proposed an almost identical algorithm in early 2019, which is Hier-AVG. The paper was theory-focused and provided little consideration for real work application scenarios, but it nonetheless contributed to building the foundation for HFL by offering a formal math-

▼ **Table 1. Comparison of different hierarchical federated learning architectures**

Architecture	Number of Layers	Client-Edge-Cloud	Scalability
HLSGD	3	Unspecified (but can be)	Unlimited
Hier-AVG	3	Unspecified (but can be)	Unlimited
HFAVG	3	Yes	Unlimited
Cross-HCN FEEL	3	No (client-edge-edge)	Limited by the coverage of the macro-cell base station
Graph-FL	2	No (client-edge)	Unlimited, but communication costs between servers exhibit quadratic growth
SD-FEEL	2	No (client-edge)	Unlimited, but may be limited by max network span in spare edge networks
Federated fog learning	$N \geqslant 2$	No (client-edge*N-cloud)	Unlimited
Multi-level HSGD	$N \geqslant 2$	No (client-edge * N-cloud)	Unlimited
F2L-LDK	3	Yes	Unlimited (supports dynamic edge participation)

F2L-LDK: full stack federated learning with label-driven knowledge
FEEL: federated edge learning
FL: federated learning
HCN: heterogeneous cellular network
HFAVG: hierarchical federated averaging algorithm
Hier-AVG: hierarchical averaging stochastic gradient descent
HLSGD: hierarchical local stochastic gradient descent
HSGD: hierarchical stochastic gradient descent
SD: semi-decentralized

ematical convergence proof for the algorithm.

Algorithm 1. The hierarchical local-SGD algorithm

Require: K ←total numbers of nodes
Require: K' ←numbers of nodes per block
Require: T←rounds of global updates
Require: N←rounds of block updates
Require: M←rounds of local updates
Initialize all models to w_0
for all K in parallel **do**:
  for t in 0, 1, 2, 3...T **do**: (Global aggregation)
    for n in 0, 1, 2, 3....N **do**: (Block aggregation)
      for m in 0, 1, 2, 3...M **do**: (local update)
        Sample minibatch of data;
        Compute the gradient;
        update the local model w_l;
      end for
      enter synchronized mode:
      w_b←Aggregate all w_l in the block;
      w_l←w_b
      end synchronized mode
    end for
    enter synchronized mode:
    w_g←Aggregate all block models w_b
    w_g←w_g
    end synchronized mode.
  end for
end for

2) The hierarchical federated averaging algorithm (HFAVG) is introduced by one of the most highly cited papers in the field of HFL, "Client-edge-cloud hierarchical federated learning" by LIU et al.,[5] which was first available as a preprint in late 2019, and was later published in 2020. Although the algorithm itself does not deviate from the HLSGD and Hier-AVG algorithms, it is the first to structure the algorithm explicitly under a three-layer client-edge-cloud setting. The main contribution of the paper is proposing HFL to solve one of the most significant challenges in the field of FL, especifically federated edge learning (FEEL): scalability.

One of the major bottlenecks in federated learning is the communication latency between a server and its clients. A solution to this problem is to use a server that is physically close to the clients. Following this reasoning, FEEL[21] emerged in early 2018. Instead of using a central cloud server to facilitate model weight aggregation, FEEL uses a single server at the edge of the network, for example, a cellular base station (CBS), to reduce latency. However, this naturally limits the scalability of the system as a CBS can only serve clients in its physical vicinity, which reduces the total size of the available client pool.

HFAVG offers an intuitive solution to bypassing this limitation by coordinating multiple edge servers through the cloud. The algorithm describes a process of performing FedAvg with each edge server independently for a set number of rounds, and then aggregating all the edge models in the cloud to obtain a global model. This structure of client-edge-cloud is intuitive and effective, and makes the application of an HFL algorithm a convincing case. This is perhaps the key reason why this paper received more popularity despite being released almost a year later than the previously mentioned papers.

3) Cross heterogeneous cellular network (HCN) FEEL. Following the footsteps of HFAVG, another paper "Hierarchical federated learning across heterogeneous cellular networks"[6] was released as a preprint in late 2019 and published in 2020. The paper also aims to solve the problem of scalability in federated edge learning. However, instead of defining a general framework for using the cloud to orchestrate edge servers, the paper suggests a more specific architecture of using small-cell base stations (SBS) as edge servers, and a dedicated macro-cell base station covering the physical area of the SBS as the central server.

The main advantage of this architecture over the more general client-edge-cloud federated learning is the low communication latency, which is further reflected in the algorithm design. The main difference between the HFL algorithms in cross HCN FEEL compared with the previous work is the lack of the innermost local update loop—each client only performs training on one minibatch of data before synchronizing with the small-cell base stations. This results in a very high rate of synchronization between clients in the same group (referred to as local averaging). Moreover, since the communication latency between the sub-cell and macro-cell base stations is also relatively small compared with a far-away cloud server, the rate at which the global model can be updated per edge model update (referred to as global averaging) can also be quite high. Studies on the convergence rate of HFL algorithms[5–6, 12, 20] have shown that the rate of local averaging for a small number of client groups is the dominating term in the overall convergence rate, calculated as the number of local iterations required. Coupled with the already low communication latency from MBS to the clients, in theory, cross HCN FEEL offers an extremely fast convergence speed compared with client-edge-cloud.

Although this conclusion may seem impressive, it holds limited practical value because convergence speed is usually not a major concern in federated learning. On the other hand, the trade-off is that the scalability of the overall framework is now limited by the coverage of MBS, which can be a major limitation. Moreover, using a cloud central coordinator has the advantage of easy management and deployment, and offers access to vast and flexible computation resources, all of which are important benefits in large-scale machine learning that cannot be easily enjoyed on a macro-cell base station.

4 Exploring the Organization in Hierarchical Federated Learning

Although three-layer HFL is the simplest and most intuitive way of organizing an HFL system, a number of works have also proposed fewer or more layers to construct an HFL system. We study some examples of such architectures and compare them with the three-layer alternatives.

4.1 Two-Layer Hierarchical Federated Learning

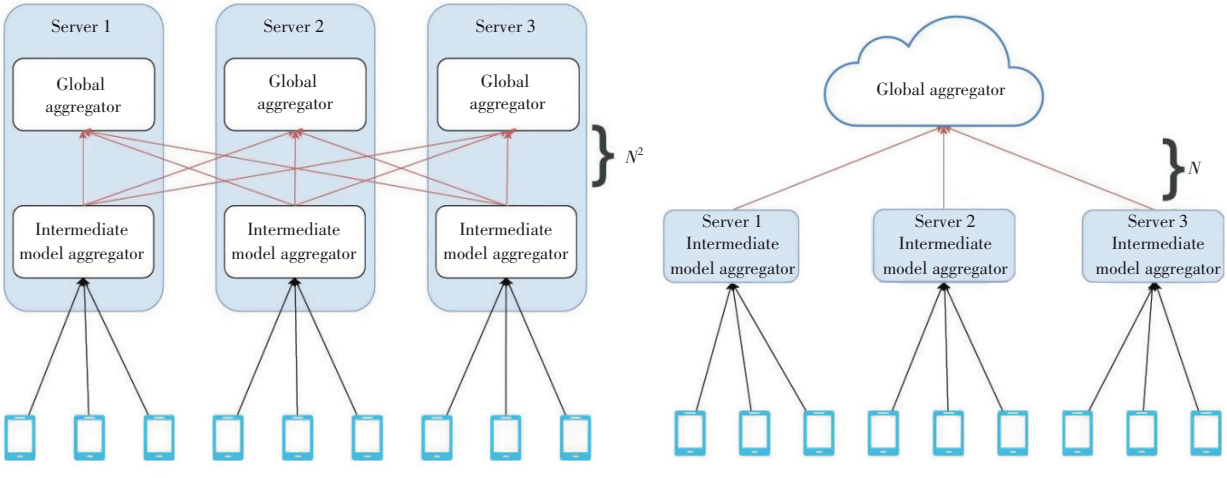
Sometimes there may not exist a dedicated global model aggregator in an HFL architecture that effectively renders the corresponding system organization into two layers. In this section, we discuss two such architectures and offer an argument that such designs can generally be replaced by a standard three-layer architecture, which would likely improves the system performance.

Introduced by RIZK et al. [14], GraphFL is a special privacy-focused HFL architecture. The main goal of GraphFL is to minimize the differential privacy risks between groups of clients. To achieve this, the clients are partitioned into small groups, each group connected to a dedicated intermediate model aggregator server running the FedAvg algorithm. So far, this is identical to standard HFL. However, instead of using a global server, the intermediate model aggregators run consensus algorithms among themselves to output a set of global models collectively, such that each server obtains its own version of the "global model". The consensus algorithm involves each server sending all the other servers the weights of its model along with a small noise, and then performing an average aggregation with all the received models.

Discussion regarding differential privacy is out of the scope of this survey. However, GraphFL nonetheless offers a new perspective: how an HFL architecture can be structured. The differential privacy setting does provide a motivation for this graph-like organization. Nevertheless, it is debatable whether such an architecture is necessary, as we can easily employ a centralized cloud server to collect all the models from the intermediate model aggregator server, perform the consensus algorithm, and then send the result models back. Not only will this architecture allow the benefit of powerful cloud computing resources, but it can also potentially reduce communication latency as client-server communication typically enjoys higher bandwidth compared with peer-to-peer communication. Moreover, as we illustrate in Fig. 4, it can also eliminate the communication step between servers to exchange models, which requires N^2 operations between N servers, as opposed to N operations by sending all the models to a central server.

Another similar implementation of the two-level HFL algorithm is semi-decentralized federated edge learning (SD-FEEL) [15] by SUN et al. Unlike GraphFL, SD-FEEL is designed as a general-purpose federated learning algorithm. In this approach, multiple edge servers coordinate clusters of client nodes to perform local model updates and intra-cluster model aggregation. The edge servers then periodically share their updated models with neighboring (one-hop) edge servers for inter-cluster model aggregation.

The paper claims that this semi-decentralized training protocol leverages the low communication latency between edge servers to facilitate efficient model exchanges, and allows for a large number of client nodes to collaborate with minimal



(a) Architecture of graph FL

(b) Architecture of three-layer client-edge-cloud FL

FL: federated learning

▲**Figure 4. Architecture of graph federated learning versus that of standard three-layer hierarchical federated learning. As we can observe, graph federated learning requires N^2 communications between the servers in the upper layer, while three-layer hierarchical federated learning only requires N**

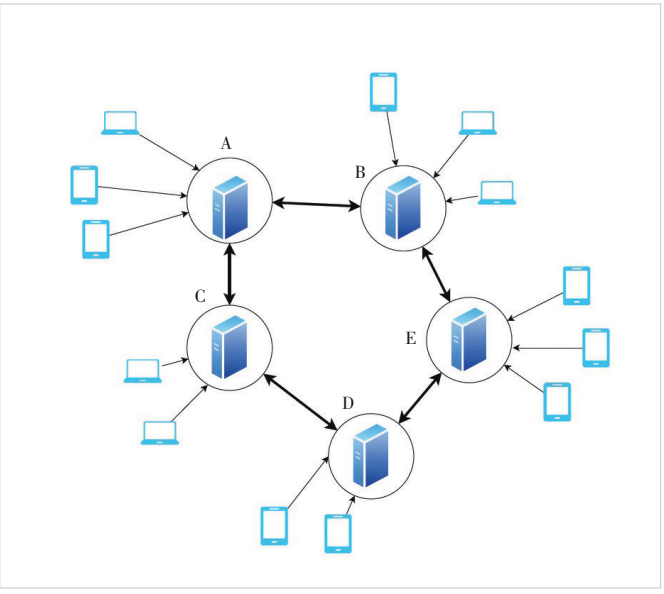
communication cost. However, there are two flaws in this claim. First, the low communication costs between edge servers are only true when the edge servers are close to each other or densely connected by a dedicated network. Second, it is very obvious that this algorithm will encounter issues when the number of edge servers grows, or when the network connecting the edge servers is sparse due to the information propagation delay. For a network where the maximum hop between two servers i, j is N, it would take at least N edge server aggregation rounds before i receives j' s update from N rounds before, and vice versa, while at that time the update may already become stale. An example is shown in Fig. 5. There is no clear solution to this problem, which makes it questionable if the slightly lower communication cost offered by this method is worth trading off the reliability and scalability benefits of using a simple three-layer client-edge-cloud architecture.

4.2 Multi-Layer Hierarchical Federated Learning

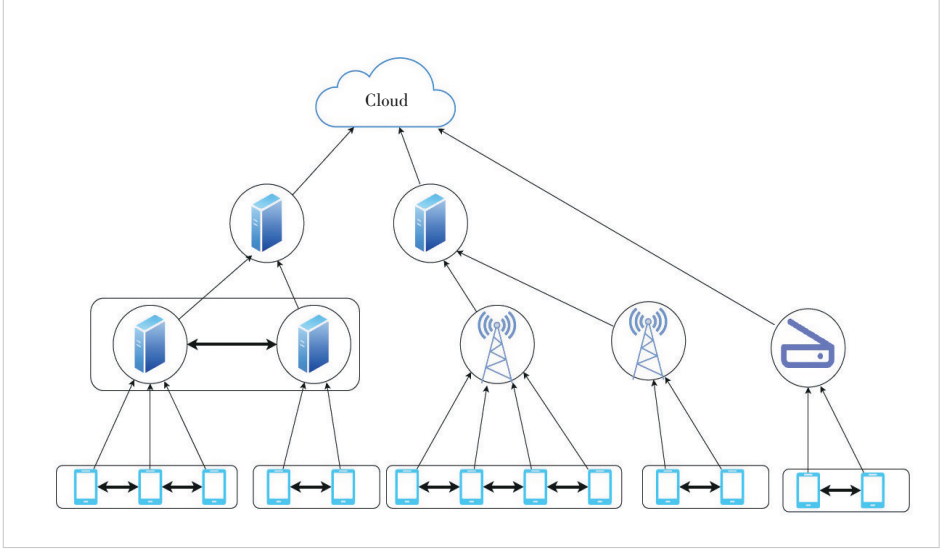
On the other end of the spectrum, we have HFL systems with multiple layers. They are typically constructed by using more than one layer of intermediate aggregators. At first glance, this kind of architecture may be intuitive, since oftentimes networks are multi-layered. However, we present two examples and argue that, typically, a three-layer HFL system is enough as an alternative to multi-layer HFL, if not better, from both a system design perspective and a theoretical perspective.

1) Federated fog learning. An example of an HFL framework that involves more than three layers is federated fog learning, introduced in Ref. [18]. Fog computing, also known as fog networking or fogging, is a distributed computing paradigm that brings computing and storage resources closer to the edge of the network in order to address the limitations of cloud computing in certain scenarios. It involves the deployment of small and low-power devices at the edge of the network, which are capable of performing local computing and storage tasks and communicating with each other and with the cloud. Due to this nature, applying HFL in fog scenarios would likely involve a multi-layered architecture design naturally. Indeed, federated fog learning introduces a generalized and multi-layered HFL architecture with the support of device-to-device collaborative training. An example is shown in Fig. 6. The local models trained on the bottom-layer IoT devices are passed through multiple layers of intermediate model aggregator devices before reaching the cloud for global aggregation. At each layer, models from the same groups in the lower layer are aggregated into a model that is reduced in dimensions (sizes) and then passed upwards, saving communication costs. Sometimes, depending on the use case, devices in the same layer may exchange models to perform a horizontal aggregation before sending the models to the upper layer.

The benefit of this architecture cannot counteract the complexity it brings. The paper does not offer any concrete algo-



▲ **Figure 5. An example of a semi-distributed federated edge learning system in a ring shaped edge network. The maximum number of hops in this network between any two servers is two, which means that, for example, it would take at least two rounds of local averaging before A receives the model from E, and vice versa**



▲ **Figure 6. Architecture of federated fog learning. The nodes enclosed in horizontal boxes perform peer-to-peer horizontal aggregation between each other before uploading the model to the upper layer**

rithms, nor does it provide proof of convergence for this architecture, rendering it difficult to properly evaluate this new class of multi-layered HFL. However, it is well-known that one of the key motivations for FL is data privacy requirements, that is, training data cannot leave the devices of the data owners. However, it is debatable if such a requirement applies to small and low-power devices such as IoT sensors in a fog computing setting. Most often, a large group of IoT devices belong to a single silo (e.g. a factory or a warehouse), where the data generated can be collected and processed in a single silo server.

Generally, each silo can be seen as one data owner, hence there is usually little motivation to keep data private to each IoT device within the silo. It is much simpler and easier to train a silo model instead and perform cross-silo federated learning between different silos. On the other hand, in situations where keeping data private for IoT devices is a hard requirement, this federated fog learning system may become useful. However, one may also argue that instead of performing intermediate aggregation steps, it can be simpler (and often faster) to just relay the end device models to the closest edge server instead.

2) Multi-level HSGD. A more detailed example of multi-layer HFL is Multi-level HSGD introduced in Ref. [12] by WANG et al. This theory-focused paper extends the existing HSGD algorithm (the same algorithm as HLSGD, HFAVG, etc.) for three-layer HFL to multiple layers, and provides a convergence-bound analysis. The multi-level HSGD algorithm introduced is the same as the standard three-layer HFL algorithms such as HLSGD, except that instead of one single layer of intermediate model aggregators, there can now be multiple layers.

The paper presents an interesting analysis result, which shows that the convergence bounds of multi-level HSGD and regular three-layer HSGD are bounded by the same upper and lower bounds. In other words, adding more layers cannot improve the worst-case or the best-case convergence values of the multi-level HSGD algorithm from that of the three-layer version. What adding more layers does allow is more freedom in choosing the hyper parameters controlling the upstream and downstream aggregation rates for each layer, where the upstream aggregation rate refers to the number of updates the server in the current layer performs before sending the models to the server in the layer above, and downstream refers to the number of updates the lower layer performs before sending the update to the current server. However, whether this increased degree of freedom is beneficial is hard to determine, because it is difficult enough to tune the single pair of upstream and downstream aggregation rates in a three-layer system (i.e. the local aggregation and global aggregation rates). This is because these values are typically found empirically, requiring a full federated learning session to run from beginning to convergence repeatedly. It is easy to imagine that increasing the degree of freedom will make it extremely tasking to find an optimal operation point for the system.

Hence, one may argue that the fact multi-level HSGD only adds more degrees of freedom in choosing upstream and downstream aggregation rates without changing the upper and lower bounds is an argument against using multi-level HSGD as opposed to three-level, in the interest of keeping the system simple without significant performance detriments.

5 Unique Applications of Client-Edge-Cloud Hierarchical Federated Learning

Ever since client-edge-cloud three-layer federated learning design rose to popularity, there have been a number of works taking advantage of this architecture and presenting interesting solutions to unique challenges in federated learning[8-10]. In this section, we briefly discuss one of these works as an example to showcase the advantage of a client-edge-cloud federated learning architecture.

Full stack federated learning with label-driven knowledge distillation (F2L-LDK) is a federated learning method introduced in Ref. [10]. The method consists of two parts: a scalable HFL framework, dubbed full-stack federated learning, and a knowledge distillation-based model training scheme aimed at solving the non-indentically and independently-distributed (non-IID) data problem, which is a major challenge in federated learning.

The HFL algorithm of F2L-LDK is almost the same as the standard HLSGD or HFAVG algorithm with the exception that instead of running another round of FedAvg at the global server for all the edge models. It performs a multi-teacher knowledge distillation instead, where the "teachers" are the edge models and the "student" is the output global model. On top of offering good performance against non-IID data, this design makes the system very flexible to the number of edge servers participating in each round, even allowing adding or removing edge servers halfway through the training process without significant detriments to the training efficiency and final converged accuracy.

F2L-LDK is a prime example of the advantages of a client-edge-cloud architecture, as it leverages the flexibility and the powerful computing resources available in the cloud to perform knowledge distillation, while preserving great scalability thanks to the edge-based intermediate-model aggregators.

6 Open Challenges and Future Research

Despite its recent rise in popularity, HFL is still a young field, especially compared with traditional non-hierarchical federated learning. Hence, many research directions mature in non-hierarchical settings have yet to be studied under the hierarchical scenario. One example is asynchronous feder-

ated learning, which is yet to be extended to a multi-layered federated learning setting. Another example is quantization and model pruning, which are great methods for reducing communication costs in federated learning but have not been applied to HFL at the time of writing this paper. Besides the two examples, there are many more exciting research topics in the field of federated learning that can be further explored with HFL architectures.

7 Related Work

7.1 Cross-Silo Federated Learning

There exists a line of work called cross-silo federated learning[22], which may bear some similarities to HFL. On the surface level, cross-silo learning also tends to follow a three-layer organization structure, where the clients are divided into groups, each group assigned a server, often called a silo or an institution. However, there is a key distinction between cross-silo FL and HFL. In cross-silo FL, it is assumed that the silos or institutions are capable of accessing client data. Moreover, each silo or institution is autonomous, often independent of the cloud service provider. In other words, each silo or institution can be viewed as a single client in the sense of traditional two-layer FL. Hence, in this work, we do not consider cross-silo FL as HFL.

7.2 Federated Edge Learning

Similar to traditional centralized federated learning, FEEL is an approach to machine learning that allows multiple devices to collaborate and learn a shared model. The two are different in the location of the model training and the data being used.

In a typical centralized federated learning setting, the global model aggregator is located in the cloud. In each round of communication, clients must directly communicate with the cloud, which may result in high communication strains. On the other hand, federated edge learning moves the global aggregator to the edge server, greatly reducing the latency and bandwidth constraints for communication between the clients and the server. However, in doing so, federated edge learning trades off scalability, since an edge server is limited to serving clients within its service range. In some situations, it may also lose the benefits of flexibility and access to an abundance of computing resources, which are both features enjoyed by cloud-based FL.

8 Conclusions

In this paper, we provided an original definition for HFL in edge computing environments in general, and the metaverse in particular. We then presented four pieces of early prototypes of HFL architectures that initialized this field of study, and compared client-edge-edge with client-edge-cloud architectures from both communication and scalability perspectives. The latter architecture would be the most fitting alternative for the metaverse. We then explored different types of HFL based on the number of layers and also presented an argument that these architectures could generally be replaced by three-layer client-edge-cloud for better performance and simplicity. Next, we demonstrated the advantages of the client-edge-cloud architecture in the metaverse, showing one example work that studied the utilization of multi-teacher knowledge distillation in FL. Finally, we outlined some potential future research directions in the field of HFL, based on existing research in traditional federated learning.

References

[1] MCMAHAN H B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data [EB/OL]. (2016-02-17) [2024-04-16]. http://arxiv.org/abs/1602.05629
[2] ZHAO Y, LI M, LAI L Z, et al. Federated learning with non-IID data [EB/OL]. (2018-06-02) [2024-04-16]. http://arxiv.org/abs/1806.00582
[3] MAO Y Y, YOU C S, ZHANG J, et al. A survey on mobile edge computing: The communication perspective [J]. IEEE communications surveys & tutorials, 2017, 19(4): 2322 – 2358. DOI: 10.1109/COMST.2017.2745201
[4] KREUTZ D, RAMOS F M V, ESTEVES VERISSIMO P, et al. Software-defined networking: A comprehensive survey [J]. Proceedings of the IEEE, 2015, 103(1): 14 – 76. DOI: 10.1109/jproc.2014.2371999
[5] LIU L M, ZHANG J, SONG S H, et al. Client-edge-cloud hierarchical federated learning [C]//IEEE International Conference on Communications (ICC). IEEE, 2020: 1 – 6. DOI: 10.1109/ICC40277.2020.9148862
[6] ABAD M S H, OZFATURA E, GUNDUZ D, et al. Hierarchical federated learning ACROSS heterogeneous cellular networks [C]//Proceedings of ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020: 8866 – 8870. DOI: 10.1109/ICASSP40776.2020.9054634
[7] LIU L M, ZHANG J, SONG S H, et al. Hierarchical federated learning with quantization: Convergence analysis and system design [EB/OL]. (2021-03-26) [2024-04-16]. http://arxiv.org/abs/2103.14272
[8] LIU C, CHUA T J, ZHAO J. Time minimization in hierarchical federated learning [EB/OL]. (2022-10-07) [2024-04-16]. http://arxiv. org/abs/2210.04689
[9] LIU X F, WANG Q, SHAO Y F, et al. Sparse federated learning with hierarchical personalized models [EB/OL]. (2023-09-25) [2024-04-16]. http://arxiv.org/abs/2203.13517
[10] NGUYEN M D, PHAM Q V, HOANG D T, et al. Label driven Knowledge Distillation for Federated Learning with non-IID Data [EB/OL]. (2022-09-30) [2024-04-16]. http://arxiv.org/abs/2209.14520
[11] Wang X, Wang Y J. Asynchronous Hierarchical Federated Learning [EB/OL]. (2022-05-31) [2024-04-16]. https://arxiv.org/abs/2206.00054
[12] WANG J Y, WANG S Q, CHEN R R, et al. Demystifying why local aggregation helps: convergence analysis of hierarchical SGD [J]. Proceedings of the AAAI conference on artificial intelligence, 2022, 36(8): 8548 – 8556. DOI: 10.1609/aaai.v36i8.20832
[13] WU W T, HE L G, LIN W W, et al. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems [J]. IEEE transactions on parallel and distributed systems, 2021, 32(7):

1539 – 1551. DOI: 10.1109/TPDS.2020.3040867
[14] RIZK E, SAYED A H. A graph federated architecture with privacy preserving learning [C]//The 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, 2021: 131 – 135. DOI: 10.1109/SPAWC51858.2021.9593148
[15] SUN Y C, SHAO J W, MAO Y Y, et al. Semi-decentralized federated edge learning for fast convergence on non-IID data [C]//Proceedings of IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2022: 1898 – 1903. DOI: 10.1109/WCNC51071.2022.9771904
[16] ZHONG Z C, ZHOU Y P, WU D, et al. P-FedAvg: Parallelizing federated learning with theoretical guarantees [C]//IEEE Conference on Computer Communications. IEEE, 2021: 1 – 10. DOI: 10.1109/INFOCOM42981.2021.9488877
[17] DAS A, PATTERSON S. Multi-tier federated learning for vertically partitioned data [C]//IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021: 3100 – 3104. DOI: 10.1109/ICASSP39728.2021.9415026
[18] HOSSEINALIPOUR S, BRINTON C G, AGGARWAL V, et al. From federated to fog learning: distributed machine learning over heterogeneous wireless networks [J]. IEEE communications magazine, 58(12): 41 – 47, 2020. DOI: 10.1109/MCOM.001.2000410
[19] LIN T, STICH S U, PATEL K K, et al. Don't use large mini-batches, use local SGD [EB/OL]. (2018-08-22) [2024-04-16]. http://arxiv.org/abs/1808.07217
[20] ZHOU F, CONG G J. A distributed hierarchical SGD algorithm with sparse global reduction [EB/OL]. (2022-02-17) [2024-04-16]. http://arxiv.org/abs/1903.05133
[21] ZHU G, LIU D, DU Y, et al. Toward an intelligent edge: wireless communication meets machine learning [J]. IEEE communications magazine, 58(1): 19 – 25, 2020. DOI: 10.1109/MCOM.001.1900103
[22] KAIROUZ P, MCMAHAN H B, AVENT B, et al. Advances and Open Problems in Federated Learning [J]. Foundations and trends® in machine learning, 14(1 – 2): 1 – 210, 2021

Biographies

GU Cheng received his BASc Degree of Honours in Computer Engineering Cooperative Program with Distinction in 2022, and his MASc degree from the Department of Electrical and Computer Engineering in May 2024, both from University of Waterloo, Canada. His research interests focus on building next-generation AI assisted distributed systems.

LI Baochun (bli@ece.toronto.edu) received his BE degree from Tsinghua University, China in 1995 and his MS and PhD degrees from the University of Illinois at Urbana-Champaign, USA in 1997 and 2000, respectively. Since 2000, he has been with the Department of Electrical and Computer Engineering, the University of Toronto, Canada, where he is currently a Professor. Since August 2005, he has been holding the Bell Canada Endowed Chair in computer engineering. He was the recipient of IEEE Communications Society Leonard G. Abraham Award in the field of communications systems in 2000, the Multimedia Communications Best Paper Award from the IEEE Communications Society in 2009, the University of Toronto McLean Award in 2009, the Best Paper Award from IEEE INFOCOM in 2023, and the IEEE INFOCOM Achievement Award in 2024. He is a Fellow of the Canadian Academy of Engineering, the Engineering Institute of Canada, and IEEE. His current research interests include cloud computing, security and privacy, distributed machine learning, federated learning, and networking.