Towards Adaptive and Scalable Hierarchical Federated Learning

1st Marco Lettiero
Dept. of Science and Technology
University of Naples Parthenope
Naples, Italy
m.lettiero@ieee.org

Abstract—Federated learning enables multiple parties to train a shared model without exchanging raw data, yet it becomes inefficient when thousands of geographically dispersed clients compete for the attention of a single coordinator. Hierarchical federated learning mitigates this bottleneck by introducing an intermediate layer of edge servers, but existing solutions rely on a fixed pool of intermediaries and cannot adapt to changing workloads. We introduce the first auto-scaling hierarchical federated learning framework that automatically creates and removes edge servers in response to real-time demand. In a fully containerized evaluation, on-demand edge servers significantly reduced the volume of traffic processed by the central coordinator compared to that in a flat architecture, while the total end-to-end traffic remained statistically identical. A cost analysis based on public cloud pricing projects annual savings, thereby demonstrating the economic value of elastic scaling with a purely cloud-based approach with respect to other methods.

Index Terms—federated learning, horizontal scaling, hierarchical federated learning, cloud computing

I. INTRODUCTION

An innovating approach to machine learning is Federated Learning (FL), which gives a new collaborative method for model training using decentralized devices without explicit data sharing [1], [2]. This is done by training the local models from client devices that is aggregated to the central node that produce the real global model. FL gives many solid main point aspects like, the communication capability of the clients and central server [2], [3], the manage around statistical heterogeneity of client datasets [4]–[6], the ensuring fairness among all participants [7], [8], improving security against various attacks [9]–[11], and developing more sophisticated aggregation mechanisms [12], [13]. While traditional FL demonstrates significant potential in collaborative model training, it faces critical challenges when scaled to large number of Geographically remote clients. The direct communication of the clients can lead to communication overhead, network congestion, and reduced scalability, particularly in scenarios characterized by high client heterogeneity or unreliable connectivity. Hierarchical Federated Learning (HFL) offers a practical solution by introducing one (or more) intermediate layer of edge servers or aggregators. With HFL, only a smaller group of edge nodes exchange messages with the central node [14], [15]. This layered approach not only reduces communication overhead but also enhances scalability

and robustness by allowing local aggregation among geographically or logically grouped clients [16]. Edge servers can buffer updates, reduce latency, and handle brief network disruptions, improving fault tolerance. Additionally, HFL supports more nuanced personalization strategies by enabling group-specific models or aggregation schemes better suited to client-level heterogeneity [17], [18]. For all these reasons, HFL is still widely researched and explored (e.g., two-tier and multi-tier architectures, more complex aggregation methods designed [19], [20]). As such, HFL is increasingly recognized as a cornerstone for deploying the federated approach in large-scale, real-world applications such as smart cities, IoT networks, and international research collaborations—contexts where data privacy, efficient communication, and resilience are essential [21], [22].

In this work, we aim to demonstrate the concrete performance benefits—both in terms of reduced communication load and economic efficiency—that result from deploying an auto-scaling HFL framework in a cloud environment. By leveraging dynamic resource provisioning and local aggregation, our system alleviates network bottlenecks and scales according to demand. We provide a cost analysis based on public cloud pricing and show that dynamically allocating edge servers can lead to significant annual savings. Our experiments also empirically confirm that the communication burden is successfully offloaded from the central server to the dynamically deployed edge nodes. The remainder of this paper is structured as follows. Section II describes the background foundations of both FL and HFL methodologies, establishing the conceptual basis of our work. Section III reviews the existing literature, highlighting relevant contributions. SectionIV properly defines the problem we aim to address in this work. SectionsV andVI detail the proposed architecture, its implementation, and the experimental results, providing also a comparison with similar approaches as discussed in the related works. SectionVII finally concludes the paper and outlines future research directions and improvements.

II. BACKGROUND

This section is used to provide the theoretical foundations for our study, as the theory of FL, and its typical formulation are discussed. We also introduce deep learning approaches, specifically the Convolutional Neural Networks (CNNs). This

background establishes the conceptual and methodological basis for the HFL framework presented in this paper.

A. Federated Learning

FL is a decentralized machine learning approach with which multiple entities, such as IoT devices [23], can collaboratively train a shared model. The main advantages behind this approach are given by the absence of data exchanges among the involved nodes and their decentralization [24]. This approach addresses significant privacy concerns and ensures compliance with data protection regulations (which may differ depending on the actual geographical localization of the data themselves) as the training procedure occurs within distinct trust boundaries, before centrally aggregating the resulting updates for model improvement.

This approach offers several key advantages described as follows. *1) Data privacy and access.* It significantly enhances data privacy and security by keeping raw data localized, thereby reducing the risk of breaches and unauthorized access to sensitive information. This design ensures compliance with stringent data protection regulations (e.g., GDPR, HIPAA), and enables organizations to derive insights without directly exposing personal identifiable information. FL also grants access to diverse and larger datasets that would otherwise remain unavailable due to privacy concerns, data ownership, or regulatory restrictions, fostering the development of more robust and generalized models. *2) Lower overhead and computational demands.* Within the FL framework, communication overhead is highly reduced because only compact model updates, rather than bulky raw data, are transmitted. The structure of FL also facilitates on-device learning and edge AI, enabling continuous model adaptation directly at the data source, which can reduce latency and improve real-time performance. *3) Robustness.* By decentralizing the training process, the risk of a single point of failure is mitigated, thereby enhancing overall system robustness.

These advantages are rooted in the fundamental formulation of FL as a decentralized optimization problem, where clients independently optimize local objectives and a central server periodically aggregates the updates. Let K denote the number of clients, where each client k has its own local dataset D_k and performs local training to optimize the model parameters w_k. The objective for each client can typically be represented as:

$$\min_{w_k} \mathcal{L}_k(w_k) = \frac{1}{|D_k|} \sum_{(x,y) \in D_k} \mathcal{L}(f(x; w_k), y)$$

where $\mathcal{L}$ is the loss function, $f(x; w_k)$ is the prediction of the model given inputs x and parameters w_k, and $|D_k|$ is the number of samples in the dataset of the client k. After local training, the clients send their updates to a central server, which aggregates the updates often using techniques such as Federated Averaging (FedAvg) [25] to form a global model w:

$$w = \frac{1}{K} \sum_{k=1}^{K} w_k. \tag{1}$$
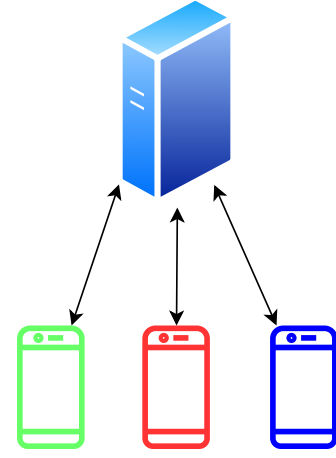
(a) Federated Learning.
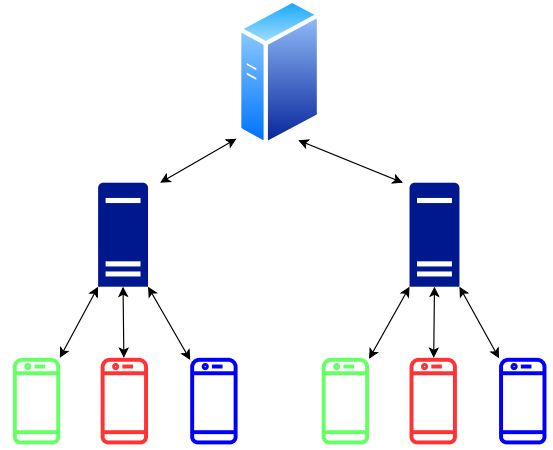
(b) Hierarchical Federated Learning.

Fig. 1. Simplified schemas of message passing in Federated Learning and Hierarchical Federated Learning. Fig. 1a illustrates the communication between federated clients (e.g., mobile devices) and the central server. Fig. 1b depicts how clients instead interact with edge servers, while the final aggregation is performed between the edge servers and the central server.

This global model is then sent back to the clients for further training and the process iterates until convergence. The key difference between the mathematical formulation of federated learning and that of traditional approaches lies in the decentralization of the training process, as summarized in Fig.1a.

B. Deep Learning and Convolutional Neural Networks

Deep learning is a subfield of machine learning that focuses on using multi-layered artificial neural networks to learn complex patterns in large datasets. These networks have revolutionized a wide range of domains, including computer vision, speech recognition, and natural language processing, thanks to their capacity to automatically extract high-level abstractions from raw data [26]–[30]. In many of these applications, CNNs have become the predominant architecture due to their ability to process grid-like data structures, such as images, by learning spatial hierarchies of features through

convolutional operations [31], [32]. CNNs achieve superior performance in visual tasks while maintaining computational efficiency, making them especially well-suited for deployment in edge environments and resource-constrained devices.

A convolutional layer in a CNN applies a set of filters (kernels) to the input data, capturing local features through the convolution operation. Let $x \in \mathbb{R}^{H \times W \times C}$ denote an input image with height H, width W, and C channels. A convolutional filter $f \in \mathbb{R}^{k_h \times k_w \times C}$ produces a feature map y via:

$$y_{i,j} = \sum_{m=1}^{k_h} \sum_{n=1}^{k_w} \sum_{c=1}^{C} f_{m,n,c} \cdot x_{i+m-1,j+n-1,c} + b,$$

where b is a bias term and (i, j) index the spatial location in the output feature map. This operation extracts local patterns, which are further refined by activation functions and pooling layers to build progressively more abstract representations.

Modern deep learning frameworks have played a pivotal role in advancing these capabilities by simplifying model development, training, and deployment. Among these, PyTorch[1] has emerged as a particularly popular and versatile framework within the research community. Its dynamic computation graph, constructed at runtime, offers flexibility for debugging and experimentation, distinguishing it from static graph frameworks such as early versions of TensorFlow[2] [33]. PyTorch is deeply embedded in the Python ecosystem, providing native integration with NumPy, seamless GPU acceleration via CUDA, and support for automatic differentiation. These features make it well-suited for both rapid prototyping and production-level deployment.

The integration of deep learning models, particularly CNNs, with federated learning frameworks has become a cornerstone of many modern distributed learning systems. CNNs, with their ability to extract compact yet informative representations from visual data, are particularly compatible with federated learning scenarios. They help mitigate communication overhead by transmitting only essential model updates rather than raw data, which is crucial for preserving privacy and ensuring data sovereignty. This synergy between deep learning and federated learning thus underpins our choice of CNN architectures for the HFL framework presented in this work.

III. RELATED WORKS

Several studies have extended analysis around HFL as a promising approach to address scalability and communication bottlenecks in large-scale federated learning deployments (Fig. 1b). A common architectural feature in these works is the presence of fixed edge servers or aggregators, which act as intermediaries between the clients and the central server. These studies have explored the benefits of using static edge servers to improve communication efficiency and overall system scalability. Namely, Lim et al. [14] proposed a hierarchical model for mobile edge computing environments,

where edge servers perform local aggregation of client updates before forwarding them to the central server. Similarly, Wang et al. [15] provided a comprehensive survey on hierarchical architectures, highlighting the advantages of fixed edge servers in reducing communication costs and latency. Vogels et al. [16] investigated the role of fixed edge aggregators in mitigating the effects of unreliable connectivity and enhancing system resilience. These works typically assume that edge servers are pre-deployed and possess sufficient computational resources to manage the aggregation tasks within their assigned client groups. While these fixed-edge approaches have demonstrated significant advantages in terms of communication reduction and system robustness, they rely on a static infrastructure that may not be adaptable to dynamic workloads or heterogeneous deployment scenarios.

To overcome the limitations of fixed-edge HFL architectures, recent research has explored the integration of orchestration frameworks and virtualization technologies to enable more flexible and adaptive federated learning deployments. These approaches aim to dynamically manage computational resources and communication pathways, thereby enhancing scalability and responsiveness to varying workloads. Liberti et al. [34] proposed a federated learning framework that leverages Kubernetes for orchestrating containerized FL tasks across heterogeneous edge environments. This setup allows for dynamic scaling and efficient resource allocation, accommodating the diverse capabilities of IoT devices and edge servers. Similarly, Lackinger et al. [35] introduced an inference load-aware orchestration scheme for HFL, which optimizes the placement of aggregator nodes and the association of devices based on real-time inference workloads and processing capacities. Further advancements include the work by Mei et al. [36], who developed an intelligent hierarchical FL system employing semi-asynchronous and scheduled synchronous control strategies within a cloud-edge-client structure. This system utilizes virtual machines to simulate client nodes and implements adaptive control mechanisms to handle the dynamic nature of satellite networks, thereby improving training efficiency and model accuracy.

These studies demonstrate the potential of combining orchestration tools and virtualization to create more resilient and adaptable HFL systems. However, these approaches are still constrained by the reliance on pre-deployed edge servers and static infrastructure components. While orchestration frameworks like Kubernetes and virtualization techniques improve resource allocation and workload balancing, they typically operate over a fixed set of edge servers or virtual machines that are provisioned in advance. This static nature limits the ability to fully exploit the dynamic scalability of modern cloud computing environments, where the creation and removal of computational resources can be driven by real-time workload demands and network conditions. In contrast, our work investigates an architecture that leverages cloud-native capabilities to dynamically instantiate and decommission edge servers in response to evolving system requirements. By enabling the automatic scaling of intermediate aggregators, our framework

---

[1] https://pytorch.org/

[2] https://www.tensorflow.org/

aims to further reduce communication bottlenecks and enhance overall system flexibility compared to solutions that rely solely on fixed or pre-allocated resources. To the best of our knowledge, no existing studies have explored this dynamic, cloud-native orchestration of HFL architectures, which highlight the novelty and practical significance of our approach.

IV. PROBLEM STATEMENT

The hierarchical structure of HFL alleviates communication bottlenecks, improves scalability, and enhances the robustness of the training process compared to traditional federated learning. In this framework, the single-level FedAvg aggregation in Eq. (1) is expanded to better accommodate the structure and heterogeneity of large-scale edge environment. Within this hierarchical approach, each edge server e begins aggregating the local models of its assigned clients $\mathcal{C}_e$ as

$$w_e = \frac{1}{|\mathcal{C}_e|} \sum_{k \in \mathcal{C}_e} w_k, \tag{2}$$

and, equivalently, the central server aggregates these intermediate edge models to compute the final global model as

$$w = \frac{1}{E} \sum_{e=1}^{E} w_e, \tag{3}$$

where E denotes the total number of edge servers. As highlighted in Sections II-A and III, current HFL frameworks are fundamentally limited by their reliance on a static pool of edge servers. This static allocation poses several challenges in real-world deployments as it cannot accommodate the dynamic nature of edge environments, where the number of active clients, data generation rates, and computational requirements can vary significantly over time. A fixed set of edge servers risks resource underutilization during periods of low activity, leading to economic inefficiency and unnecessary energy consumption. During peak workloads, instead, static resources may become overwhelmed, causing increased latency, dropped updates, and degraded convergence of the global model. Static edge server placement also fails to align with the evolving spatial distribution and data heterogeneity of participating clients, which often exhibit significant variations in data distributions and connectivity patterns. A fixed infrastructure cannot adapt to these variations, leading to suboptimal aggregation topologies that compromise the effectiveness and fairness of the global model.

Fault tolerance is inherently limited in static configurations, which is another crucial aspect to be addressed. The failure of an edge server in a rigid setup can disrupt a large subset of clients, causing interruptions to the federated training process and potentially introducing inconsistencies in the global model updates. This lack of adaptability creates a substantial barrier to maintaining robust and reliable training in dynamic, real-world environments.

Although recent studies have proposed using orchestrators and virtualization tools to improve resource management and placement, these approaches still operate on a fixed or pre-provisioned set of edge servers. They cannot fully leverage
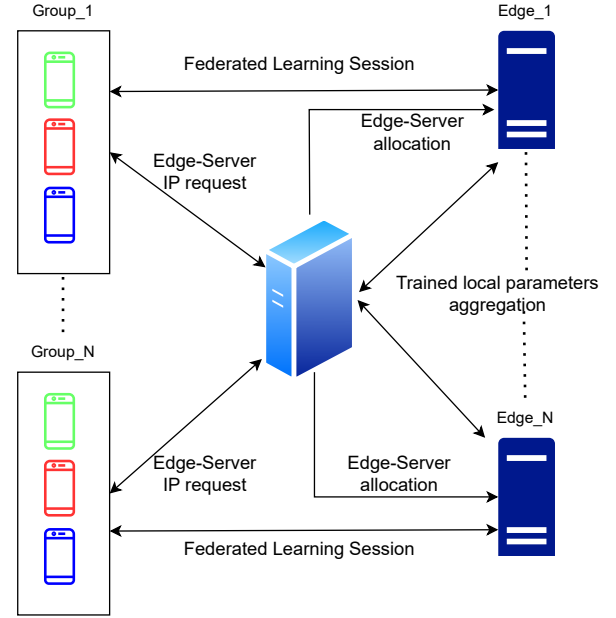

Fig. 2. The figure depicts the auto-scaled hierarchical federated learning framework, where clients connect to dynamically created edge servers for partial training before final aggregation at the central server.

the dynamic elasticity of modern cloud computing platforms, where computational resources can easily and promptly be created or removed on demand to match real-time requirements.

For all these limitations and requirements, there is a need for dynamic auto-scaling mechanisms that can intelligently adjust not only the number of active edge servers but also their deployment locations based on current system conditions. Such mechanisms would help HFL systems adapt to changing workloads and client populations, optimize resource usage, reduce operational costs, and improve overall system resilience and scalability. Addressing this gap forms the foundation of our work, which aims to realize a truly adaptive and efficient HFL framework.

V. PROPOSED METHOD

Our proposed method introduces an HFL architecture incorporating an auto-scaling mechanism for edge servers. This system is designed to efficiently manage communication overhead and dynamically adapt to varying numbers of clients without relying on an external orchestrator like Kubernetes. The core components of our system include clients, edge servers, and a central coordinator.

A. Hierarchical Federated Learning Architecture

The HFL architecture is structured into three main layers. At the base are the clients, which are end-user devices holding local datasets. Each client trains a local model—specifically, an educational version of VGG [37], tinyVGG, in this study, on its data and transmits the resulting model updates (weights) to a designated edge server. To emulate the heterogeneous data distributions typically encountered in real-world federated

learning environments, we applied a random data partitioning strategy across the clients. The number of clients in real-world applications is typically large enough to cover the entire data distribution, effectively replicating the use of the full training dataset in a local environment. In our simulations, however, we deployed only a few clients due to resource constraints. This limitation introduces the risk of using non-IID local datasets and consequently obtaining locally underfitted models.

Edge servers form the intermediate layer and serve as local aggregators. Each edge server collects model updates from a subset of clients and performs an initial aggregation to produce an intermediate model. The number of active edge servers is dynamically controlled by our auto-scaling mechanism, which adjusts resources based on current system demands.

At the top layer is the coordinator, a central server that oversees the federated learning process. It is responsible for assigning newly connected clients to suitable edge servers and for performing global aggregation upon completion of the intermediate federated learning sessions.

B. Auto-Scaling System for Edge Servers

A central innovation of our method is the built-in auto-scaling system for edge servers. This feature enhances resource efficiency and helps manage communication bottlenecks, particularly those that may arise at the coordinator level. In the remainder of this section, the steps of our approach, summarized in Fig. 2, are described with implementation details.

When clients intend to join a FL session, they first query the coordinator to obtain the IP address of an available edge server. This step is handled transparently, so that clients are unaware of the hierarchical topology of the network. The coordinator can leverage these requests to implement a horizontal scaling policy: as the number of requests to the central server grows, new edge servers are dynamically instantiated. The coordinator keeps track of all assignments to ensure system robustness, enabling client reconnections and managing edge server failures. A limitation of the proposed approach is that it does not support multi-layer scaling; only horizontal scaling is addressed.

This auto-scaled system greatly benefits from cloud computing solutions such as Azure VMSS. Once a policy is defined by setting a threshold for the communication overhead, VMSS can instantiate new virtual machines within a few minutes using a pre-defined "golden image". Another important advantage of a cloud-based solution is access to integrated distributed storage services (e.g., Azure Blob Storage[3]), which enable edge servers to automatically recover from crashes or malfunctions by reading the latest partial results from the distributed storage. This approach, however, requires the creation and assignment of multiple public IP addresses and virtual machines—one for each edge server. While this is typically not an issue in production environments, our simulations were limited by the resources available under

the Azure for Students plan[4], which allows only three virtual machines and three public IP addresses. Consequently, we conducted the experiments in a fully simulated environment on a single host.

To formalize the dynamic behavior of our approach, we extend the aggregation expressions to explicitly incorporate the variable number of edge servers and their dynamic client assignments. In the proposed dynamic auto-scaling framework, the number of active edge servers E introduced in Eq. (3) varies over time and is determined by a scaling function ϕ that depends on system conditions (e.g., client activity, communication load). At any given aggregation round t, the number of edge servers is given by

$$E(t) = \phi\big(\lambda(t)\big),$$

where $\lambda(t)$ represents the real-time workload or other system indicators.

The Eq. (2) and Eq. (3) are then expressed, respectively, as

$$w_e^{(t)} = \frac{1}{|\mathcal{C}_e^{(t)}|} \sum_{k \in \mathcal{C}_e^{(t)}} w_k^{(t)}, \quad w^{(t)} = \frac{1}{E(t)} \sum_{e=1}^{E(t)} w_e^{(t)}, \quad (4)$$

where $\mathcal{C}_e^{(t)}$ denotes the dynamically assigned set of clients to edge server e at round t, and $w^{(t)}$ is the global model at that round.

VI. Experimental Setup and Results

Experiments and simulations were conducted on a local host machine equipped with an Intel Core i9-13900K CPU (24 cores, 32 threads) and 32 GB of RAM. This hardware configuration enabled us to circumvent the quota limits imposed by Azure for Students by running several Docker containers, each with four dedicated CPU cores, to host all the required entities (i.e., the coordinator, multiple edge servers, and multiple clients). The deep learning model used in this work, tinyVGG, was trained using CPU only to simulate typical scenarios where clients have limited computational resources. Each client performed a single local training epoch per round, using a batch size of 64 and 10% of the Fashion-MNIST training dataset (i.e., 6,000 samples). In our simulations, the simplicity of tinyVGG allowed the federated training to converge in approximately three rounds. The coordinator applied FedAvg as in Eq. (4) to aggregate the intermediate partial results. To simulate the behaviour of a real network with existing DNS, we also relied on the networking features of Docker. Code is available on https://github.com/Wenress/Hierarchical_FL. Assuming a cost of ~ 0.8 USD/hour for an edge VM with four cores and an additional daily cost of ~ 1 USD for the local storage and static public IP of the VM, the static deployment of two edge servers would incur a daily cost of ~ 40.4 USD. With the proposed dynamic allocation—where edge servers operate for an estimated 8 hours per day—the daily cost decreases to ~ 14.8 USD. This results in an estimated daily savings of ~ 25.6 USD, corresponding to an annual savings

TABLE I
DAILY AND ANNUAL COSTS FOR STATIC VS. DYNAMIC EDGE
DEPLOYMENTS (OURS). ALL VALUES ARE EXPRESSED IN USD.

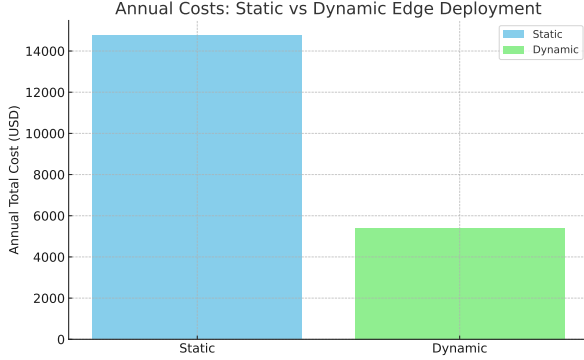| Deployment	VM (day)	Total (day)	Total (year)
Static (24 h/day)	38.4	40.4	14 746
Ours (8 h/day)	12.8	14.8	5 402
Savings	—	**25.6**	**9 344**

Fig. 3. Estimated annual cost comparison between static and dynamic approach (ours).

of approximately ~ 9344 USD. These theoretical results are summarized in Table I.

TABLE II
NETWORK-TRAFFIC DISTRIBUTION: TRADITIONAL FL VS. PROPOSED HFL
(PERCENTAGES REFER TO THE FL BASELINE). VALUES ARE AVERAGED
OVER 10 SIMULATIONS; BASELINE UNCERTAINTY IS ±10%, HFL
UNCERTAINTIES ARE ERROR-PROPAGATED (±14%).

Entity	FL Baseline (%)	HFL (% of baseline)
Coordinator (central)	100 ± 10	10 ± 1
Edge servers (total)	–	43 ± 6
Clients (total transmitted)	100 ± 10	100 ± 14
Total network traffic	100 ± 10	102 ± 14

In the proposed hierarchical configuration with horizontal auto-scaling, the deployment of two edge servers redistributes network traffic while adding only negligible overhead. Compared with the flat client–coordinator topology adopted as the 100 % baseline, the coordinator now handles merely $10 \pm 1\,\%$ of the traffic, thus off-loading roughly 90 % of the upstream volume. The edge layer absorbs $43 \pm 6\,\%$ of the baseline throughput, whereas clients continue to transmit essentially the same amount of data as in the baseline run ($100 \pm 14\,\%$), so no extra burden is imposed at the user side. A two-sample t-test ($n = 10$ repetitions per scheme) shows that the apparent increase in end-to-end traffic from $100 \pm 10\,\%$ to $102 \pm 14\,\%$ is not statistically significant ($t = 0.37$, $p = 0.72$); the corresponding 95 % confidence intervals, $[93.7, 106.3]$ pp and $[93.1, 110.9]$ pp, overlap extensively. Consequently, the overall traffic remains statistically equivalent to the baseline, while the communication load is cleanly partitioned across the edge
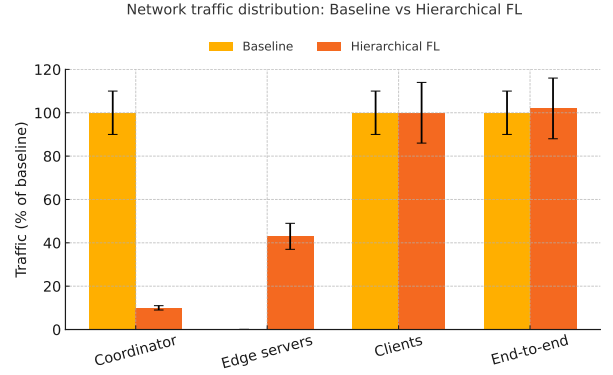

Fig. 4. Comparison of network traffic distribution between the baseline (flat FL) and the hierarchical FL (HFL) configuration. Bars represent mean percentages of traffic relative to the baseline, and error bars indicate standard deviations over 10 repetitions.

tier and the coordinator is virtually unburdened (Table II). Nonetheless, in wide-area deployments the modest overhead observed here may be exacerbated by routing and name-resolution latencies (e.g., DNS or CDN hops), warranting further investigation.

Finally, we note that the choice of the orchestration substrate has non-negligible implications for the economics and the agility of each HFL session. In our approach, the absence of control-plane overhead eliminates fixed costs and favours short-lived, geographically sparse workloads. The coarser scaling granularity of virtual machines, however, lengthens time-to-scale and increases operational complexity when edge sites proliferate (e.g., 2-3 minutes to deploy a new VM in cloud). Conversely, a container-native approach based on Kubernetes introduces a constant, region-specific control-plane charge, but amortises this expense when edge servers remain active for extended periods or when the deployment spans a dense set of regions. The finer-grained pod scheduling offered by Kubernetes reduces recovery times and supports rapid model roll-outs, while its declarative APIs facilitate portability across heterogeneous cloud and on-premise environments—an asset for world-wide HFL where regulatory and latency constraints vary. Hence, the theoretical break-even point between the two paradigms is governed not only by the runtime duty cycle of each edge server but also by the desired elasticity, the frequency of model updates across federated regions.

VII. CONCLUSION

This work has presented the first auto-scaled HFL framework that can elastically instantiate and retire edge servers according to real-time demand. Through a fully containerised simulation, we demonstrated that, with only two dynamically created edge servers, the traffic handled by the central coordinator drops from the baseline $100 \pm 10\,\%$ to merely $10 \pm 1\,\%$, while the clients incur no additional transmission burden ($100 \pm 14\,\%$). The overall end-to-end traffic remains statistically indistinguishable from the flat FL baseline ($102 \pm 14\,\%$, $p = 0.72$), confirming that the communication

overhead introduced by our hierarchical design is negligible. A cost model based on Azure pricing further indicates potential annual savings compared with a static edge deployment, underlining the economic advantage of on-demand scaling. Beyond the empirical evidence, our study contributes (i) a lightweight orchestration mechanism that removes the need for external platforms such as Kubernetes, (ii) an explicit analytical formulation of dynamic edge aggregation, and (iii) an open-source reference implementation that makes HFL readily deployable in research and industry settings.

The next step is to migrate the prototype to VMSS to validate auto-scaling on real cloud infrastructure. We also plan to extend the current three-tier hierarchy to multi-tier topologies, enabling fine-grained aggregation across diverse organisational or geographic clusters. Thus, large-scale, globally distributed experiments to characterise latency, throughput, and energy efficiency at Internet scale will foster a more robust, cost-efficient, and truly planet-wide deployment of federated learning systems.

REFERENCES

[1] H. 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*, pp. 1273–1282, PMLR, 2017.

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

[3] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedsplit: An algorithmic framework for fast federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10376–10387, 2020.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[5] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, pp. 5132–5143, PMLR, 2020.

[6] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[7] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *International Conference on Learning Representations*, 2019.

[8] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*, pp. 4615–4625, PMLR, 2019.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "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.

[10] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *arXiv preprint arXiv:1911.11836*, 2019.

[11] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2020.

[12] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *International Conference on Machine Learning*, pp. 7252–7261, PMLR, 2019.

[13] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *International Conference on Learning Representations*, 2020.

[14] 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*, vol. 22, no. 3, pp. 2031–2063, 2020.

[15] J. Wang, Q. Liu, H. Min, Z. Dou, B. Yang, C. Zhang, and Z. Chen, "A survey on federated learning: The journey from central to hierarchical paradigms," *ACM Computing Surveys (CSUR)*, vol. 55, no. 8, pp. 1–38, 2022.

[16] T. Vogels, S. Jégou, and S. Avestimehr, "Powergossip: Practical algorithm for communication-efficient decentralized deep learning," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 13179–13191, 2021.

[17] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 19588–19598, 2020.

[18] G. Long, T. Shen, M. Xu, and C. Zhang, "Multi-center federated learning: A personalized hierarchical approach," *arXiv preprint arXiv:2009.03728*, 2021.

[19] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2021.

[20] R. Castiglia, S. Scardapane, D. Bacciu, and A. Uncini, "Hierarchical federated learning for multi-center medical image analysis," in *International Conference on Artificial Neural Networks*, pp. 308–319, Springer, 2020.

[21] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep reinforcement learning for federated learning-based resource allocation in iot," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15758–15769, 2021.

[22] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 46–52, 2020.

[23] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[24] B. Kumar, S. Singh, R. Grover, K. R. Isabels, A. Garg, and B. Charudatta Dattatraya, "Analysis of mathematical modelling deterministic and stochastic problems in federated learning," in *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 1700–1704, 2023.

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

[26] Y. Yu, C. Wang, Q. Fu, R. Kou, F. Huang, B. Yang, T. Yang, and M. Gao, "Techniques and challenges of image segmentation: A review," *Electronics*, vol. 12, no. 5, p. 1199, 2023.

[27] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern recognition letters*, vol. 119, pp. 3–11, 2019.

[28] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5731–5780, 2022.

[29] M. Taboada, "Sentiment analysis: An overview from linguistics," *Annual Review of Linguistics*, vol. 2, no. 1, pp. 325–347, 2016.

[30] Y. Liu and J. Zhang, "Deep learning in machine translation," *Deep learning in natural language processing*, pp. 147–183, 2018.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[33] V. Subramanian, *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch*. Packt Publishing Ltd, 2018.

[34] F. Liberti, D. Berardi, and B. Martini, "Federated learning in dynamic and heterogeneous environments: Advantages, performances, and privacy problems," *Applied Sciences*, vol. 14, no. 18, p. 8490, 2024.

[35] A. Lackinger, P. A. Frangoudis, I. Čilić, A. Furutanpey, I. Murturi, I. P. Žarko, and S. Dustdar, "Inference load-aware orchestration for hierarchical federated learning," in *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, pp. 1–9, IEEE, 2024.

[36] Q. Mei, R. Huang, D. Li, J. Li, N. Shi, M. Du, Y. Zhong, and C. Tian, "Intelligent hierarchical federated learning system based on semi-asynchronous and scheduled synchronous control strategies in

satellite network," *Autonomous Intelligent Systems*, vol. 5, no. 1, p. 9, 2025.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.