Kathmandu University Journal of Science, Engineering and Technology, 2025

Kathmandu University
Journal of Science, Engineering and Technology

Federated Learning Framework for Scalable AI in Heterogeneous HPC and Cloud Environments

Sangam Ghimire[a*], Paribartan Timalsina[a], Nirjal Bhurtel[a], Bishal Neupne[a], Bigyan Byanju Shrestha[a]
Subarna Bhattarai[a], Prajwal Gaire[a], Jessica Thapa[a], Sudan Jha[a]

[a]*Department of Computer Science and Engineering, SoE,, Kathmandu University, Nepal.*

Abstract

As the demand grows for scalable and privacy-aware AI systems, Federated Learning (FL) has emerged as a promising solution, allowing decentralized model training without moving raw data. At the same time, the combination of high-performance computing (HPC) and cloud infrastructure offers vast computing power but introduces new complexities—especially when dealing with heterogeneous hardware, communication limits, and non-uniform data. In this work, we present a federated learning framework built to run efficiently across mixed HPC and cloud environments. Our system addresses key challenges such as system heterogeneity, communication overhead, and resource scheduling, while maintaining model accuracy and data privacy. Through experiments on a hybrid testbed, we demonstrate strong performance in terms of scalability, fault tolerance, and convergence, even under non Independent and Identically Distributed (non-IID) data distributions and varied hardware. These results highlight the potential of federated learning as a practical approach to building scalable Artificial Intelligence (AI) systems in modern, distributed computing settings.

Keywords: Federated Learning (FL), High Performance Computing (HPC), Cloud Computing, Distributed AI, Heterogeneous Systems, Scalability.

1. Introduction

As AI models continue to grow in complexity and size, so does the demand for vast computational resources and access to large-scale distributed datasets. At the same time, growing concerns about data privacy, ownership, and regulatory compliance make it increasingly difficult to centralize data for training. FL has emerged as a promising paradigm for addressing these challenges, enabling the training of collaborative models across multiple data silos without requiring the raw data to leave its source. While FL has gained traction in mobile and edge environments, such as smartphones and IoT devices, its application in large-scale computing platforms like HPC clusters and cloud infrastructure remains underexplored.

Meanwhile, the convergence of HPC and cloud computing is reshaping the landscape of modern data-intensive applications. These hybrid environments combine the raw power and efficiency of HPC with the scalability and flexibility of the cloud, making them well-suited for training large AI models. However, this integration brings new challenges: heterogeneous hardware (e.g., Central Processing Units (CPUs), Graphics Processing Units (GPUs), Tensor Processing Units (TPUs)) , inconsistent network performance, dynamic resource availability, and non-uniform data distributions across clients.

In this context, the deployment of federated learning across such mixed infrastructure is both a timely opportunity and a technical challenge. This paper explores how FL can be adapted and optimized to run efficiently across heterogeneous HPC and cloud environments, with a focus on scalability, system resilience, and performance under non-IID data conditions.

[*]Corresponding author. Email: 1sangamghimire1@gmail.com

2. Literature Review

FL has rapidly grown as a research area since it was introduced by McMahan et al. [1], with the goal of enabling decentralized model training while preserving user privacy. FL removes the need to centralize data by having clients train models locally and only share model updates. This is especially beneficial in settings where privacy, data security, or legal constraints prevent data sharing, such as healthcare, finance, and mobile systems.

Initial research in FL focused on edge computing environments where clients are resource-constrained devices such as smartphones or sensors [2]. In these settings, major challenges include limited computation, high client churn, and extremely non-IID data distributions. Approaches such as Federated averaging (FedAvg) [1], FedProx [3], and Federated Matched Averaging (FedMA) [4] sought to improve convergence and personalization under such constraints. Systems like TensorFlow Federated [5], FedML [6] and Flautim [7] were developed to provide flexible simulation and experimentation tools tailored for these edge use cases.

However, these systems generally assume relatively homogeneous client devices and do not scale effectively to enterprise-grade computing environments, such as data centers, cloud platforms, or HPC clusters. They are also not optimized for high-throughput interconnects, batch schedulers (like SLURM), or mixed-architecture setups (e.g., CPU, GPU, TPU nodes).

Several efforts have focused on benchmarking and simulating FL workloads. LEAF [8] introduced a suite of datasets and evaluation metrics for real-world federated scenarios, while FedScale [9] expanded on this by providing a large-scale FL benchmarking platform that supports real-world training pipelines and system heterogeneity. Although these platforms help evaluate FL algorithm behavior at scale, they are primarily used for simulation, rather than actual deployment across hybrid infrastructures.

The idea of bringing FL to HPC and cloud environments has only recently begun to gain traction. Haus et al. [10] explored integrating FL with traditional HPC architectures, emphasizing challenges such as batch job scheduling and MPI-based communication. Similarly, the Adaptive Federated Learning framework introduced in [11] incorporates adaptive workload partitioning for federated learning on remote cloud platforms. However, these systems often operate in siloed infrastructure—either HPC-only or cloud-only—and do not address hybrid orchestration across both.

In contrast, recent work on cross-silo FL [12] focuses on collaborative learning across data centers or enterprises. These scenarios involve more powerful clients with higher reliability and are more aligned with the settings of HPC and cloud systems. Yet, most existing implementations assume static infrastructure and overlook the dynamic nature of multi-cloud and hybrid HPC environments.

A major obstacle in deploying FL at scale is dealing with heterogeneity—not only in data, but in compute capabilities, memory, and network bandwidth. Approaches like Federated learning on non-IID data with Reinforcement learning and Quantifying historical contribution (FedRQ) [13] and FedBalancer [14] introduce adaptive client selection and participation strategies to address system imbalance. Similarly, techniques such as gradient compression [15], gradient sparsification [16], and adaptive synchronization [17] reduce communication overhead in bandwidth-constrained scenarios. Yet, incorporating these methods into a unified system that can operate across a wide spectrum of infrastructure types—HPC nodes with Infiniband, cloud instances with variable latency, or edge clients with intermittent connectivity—remains a significant engineering challenge.

In summary, the current FL landscape includes substantial work in algorithmic design, simulation platforms, and edge-device scenarios. However, few systems target practical deployment in environments where high-performance computing meets cloud infrastructure, which are increasingly common in research institutions, enterprises, and national labs. These hybrid setups introduce complex scheduling, heterogeneity, and data locality constraints that are not fully addressed by current FL frameworks. This paper aims to fill that gap by presenting a federated learning system built specifically for heterogeneous HPC-cloud environments, with contributions at both the system and algorithmic level: dynamic client orchestration, resource-aware scheduling, and robust model aggregation under non-IID and non-uniform resource conditions.

3. Proposed System Architecture

To support scalable and privacy-preserving federated learning in heterogeneous computing environments, we propose a modular and adaptable system architecture that integrates cloud-native and HPC resources under a unified framework. The design addresses core challenges such as client heterogeneity, resource variability, fault tolerance, and efficient communication—all while preserving data locality and minimizing orchestration overhead.

The architecture consists of four primary components: **Central Orchestrator**, **Federated Clients (Workers)**, an optimized **Communication Layer**, and a dynamic **Scheduler Adapter**. Each component is designed to function independently yet integrate seamlessly, enabling operation across cloud instances, HPC clusters, or hybrid combinations of both. The proposed system architecture is shown in Figure 1.

3.1. Design Objectives

Our system architecture is guided by the following key objectives:

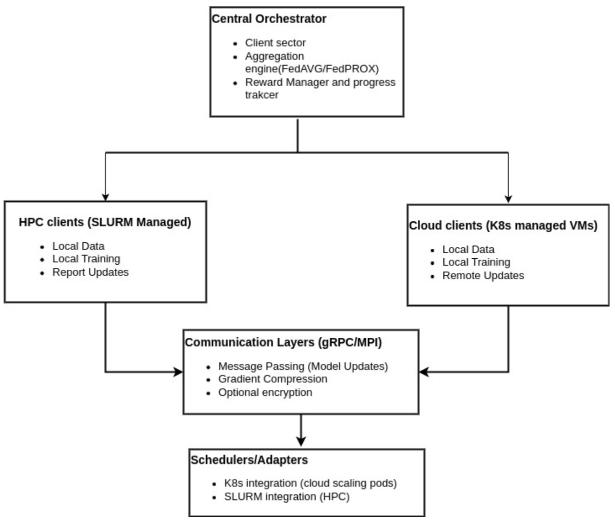

Central Orchestrator
Client sector
Aggregation engine(FedAVG/FedPROX)
Reward Manager and progress trakcer
HPC clients (SLURM Managed)
Local Data
Local Training
Report Updates
Cloud clients (K8s managed VMs)
Local Data
Local Training
Remote Updates
Communication Layers (gRPC/MPI)
Message Passing (Model Updates)
Gradient Compression
Optional encryption
Schedulers/Adapters
K8s integration (cloud scaling pods)
SLURM integration (HPC)

Figure 1: Architecture of the proposed federated learning framework.

- **Scalability:** Efficiently support large-scale federated training across hundreds or thousands of distributed nodes with minimal coordination overhead.
- **Heterogeneity Awareness:** Adapt to variations in hardware configurations, network conditions, and runtime performance across diverse client nodes.
- **Data Privacy:** Ensure that raw data remains local to client nodes, in line with privacy-preserving principles and regulatory compliance.
- **Fault Tolerance:** Maintain robust training performance despite client dropouts, preemptions (e.g., in spot instances), or network disruptions.

3.2. System Components

- **Central Orchestrator:** The central orchestrator is the core coordination unit responsible for managing the federated learning workflow. It is lightweight and stateless, and can be deployed on a cloud controller, an HPC login node, or a dedicated service. It carries out tasks like dynamically selecting a subset of available clients for each round based on compute capacity, bandwidth, and past reliability, aggregation of local model updates using strategies such as FedAvg[1] or FedProx [3] and tracking training progress, managing round deadlines, and handling missing or delayed updates via fault-tolerant coordination logic.

- **Communication Layer:** A flexible and efficient communication layer enables reliable messaging and bandwidth-aware transfer of updates. It supports the protocol like gRPC Remote Procedure Call for cloud-native messaging and Message Passing Interface (MPI) for low-latency interconnects in HPC systems. Similarly for bandwidth it employs gradient compression, sparsification, and federated dropout to reduce communication overhead. To ensure there is secure data transmission and model traning we can extend Transport Layer Security (TLS) encryption or secure aggregation.

- **Scheduler Adapter:** The scheduler adapter provides an abstraction layer between the federated learning system and underlying resource managers. It integrates with SLURM in traditional HPC environments and handles job submission, monitoring, and resource allocation. For efficient management of containerized workloads it supports dynamic pod orchestration and autoscaling through Kubernetes in cloud deployments. In addition, this supports for hybrid coordination capabilities facilitating scheduling across both HPC and cloud resources, supporting elastic, mixed-infrastructure setups.

3.3. Workflow Overview

Each training round begins with the orchestrator selecting clients based on resource profiling and availability. The global model is distributed, and clients train locally on their private data. After local training, clients send model updates back through the communication layer. The orchestrator aggregates these updates to generate a new global model. This process repeats iteratively until the model converges. The algorithm of the proposed workflow is given in Algorithm 1.

The modularity of this architecture enables flexible deployment, adaptive scaling, and robust performance even in the presence of heterogeneous hardware, non-uniform data, and unpredictable resource dynamics.

Algorithm 1 Federated Learning Training Procedure

```
Require: Initial global model $M_0$, total rounds $R$, convergence threshold $\varepsilon$
Ensure: Final global model $M$
 1: Initialize round counter $r \leftarrow 0$
 2: while $r < R$ do
 3:     Announce start of Round $r$
 4:     Central orchestrator selects a subset of clients $\mathcal{C}_r$
 5:     Send current global model $M_r$ to all clients in $\mathcal{C}_r$
 6:     for all client $c \in \mathcal{C}_r$ in parallel do
 7:         Client $c$ trains $M_r$ on local private data
 8:         Client computes local model update $\Delta M_c$
 9:         Client sends $\Delta M_c$ to the central orchestrator
10:     end for
11:     Server aggregates updates: $\Delta M \leftarrow \text{Aggregate}(\{\Delta M_c \mid c \in \mathcal{C}_r\})$
12:     Update global model: $M_{r+1} \leftarrow M_r + \Delta M$
13:     if Converged($M_r, M_{r+1}, \varepsilon$) then
14:         return $M_{r+1}$
15:     end if
16:     $r \leftarrow r + 1$
17: end while
18: return $M_r$
```

4. Heterogeneity-Aware Optimization

Heterogeneity in FL arises not only from data (non-IID distributions) but also from system-level differences such as compute power, memory, network bandwidth, and availability across participating nodes. These disparities are especially pronounced in mixed HPC and cloud environments, where client capabilities can vary widely. In this section, we describe the optimization techniques incorporated into our system to address these challenges and ensure robust and efficient training.

4.1. Adaptive Client Selection

Selecting the right subset of clients for each training round is critical for maintaining both fairness and efficiency in heterogeneous settings. Our system implements an **adaptive client selection mechanism** that takes into account several factors like resource profiling, performance history and load balancing. Through resource profiling clients are periodically benchmarked for available CPU/GPU capacity, memory, and network latency. For performance history the orchestrator maintains a record of successful participation, update quality, and completion time for each client and through load balancing underperforming or slower nodes may be temporarily excluded to avoid slowing down the round or causing stale updates.By dynamically adjusting the client pool per round, the system avoids bottlenecks and maintains training momentum, even when participating nodes vary significantly in capability.

4.2. Straggler Mitigation

In federated learning, *stragglers*—clients that take disproportionately long to complete local training—can delay or degrade the effectiveness of global aggregation. To handle this, we employ two complementary techniques:

- **Deadline-Based Cutoff:** Each round has a configurable time budget. Clients that fail to report updates within the deadline are skipped for that round, reducing the impact of delays [18].
- **Partial Aggregation:** Rather than waiting for all selected clients, the orchestrator aggregates updates from the fastest k clients (where k is tunable), ensuring timely global updates without compromising overall convergence [19].

These approaches help mitigate the impact of slow or overloaded nodes, particularly useful in environments with spot instances or shared HPC queues.

4.3. Communication-Efficient Updates

Communication is often a limiting factor in federated learning, especially across geographically dispersed cloud regions or bandwidth-constrained HPC interconnects. Our framework incorporates several techniques to reduce communication overhead. One of them is gradient quantization that reduces the precision of model updates before transmission, minimizing bandwidth without significantly affecting accuracy. Similarly, using update sparsification method clients transmit only a subset of the most important gradients (e.g., top-k by magnitude), further reducing data size. Furthermore, we employ federated dropout where clients train and communicate only partial models by randomly dropping out neurons or layers, decreasing both compute and communication loads. These techniques are especially valuable when federated clients operate over variable or costly network links, such as cloud egress bandwidth.

4.4. Robust Aggregation under Non-IID Data

Our system supports multiple aggregation methods tailored for non-IID client data distributions. In addition to the standard FedAvg [1], we implement FedProx that adds a proximal term to local objectives to prevent client models from drifting too far from the global model, improving stability in heterogeneous data conditions [3] and weighted aggregation that assigns dynamic weights to each client update based on local data size, training loss, or gradient variance, improving fairness and convergence. This combination of algorithmic robustness and system-level optimizations ensures that training remains effective even under significant data and system heterogeneity.

5. Experimental Evaluation

To validate the effectiveness, scalability, and robustness of our proposed federated learning framework, we conducted a series of experiments in a hybrid computing environment comprising both HPC and cloud resources. This section describes our experimental setup, datasets, evaluation metrics, and key findings.

Table 1: Summary of Heterogeneity-Aware Optimization Techniques

Challenge	Technique	Description
System Heterogeneity	Adaptive Client Selection	Clients selected based on compute capability, availability, and historical performance.
Straggler Clients	Deadline-Based Cutoff	Late clients are excluded from aggregation if they miss the round deadline.
	Partial Aggregation	Aggregation proceeds after receiving updates from the fastest k clients.
Communication Overhead	Gradient Quantization	Model updates are compressed by reducing numerical precision (e.g., 16-bit, 8-bit).
	Update Sparsification	Only top-k most significant gradients are sent, reducing payload size.
	Federated Dropout	Clients train and transmit reduced model subsets to lower communication cost.
Data Heterogeneity (Non-IID)	FedProx	Adds a regularization term to stabilize training with divergent local updates.
	Weighted Aggregation	Updates weighted by factors like local dataset size or training loss.

5.1. Experimental Setup

Our testbed consists of a hybrid cluster with 30 virtual machines (VMs) on Amazon Web Services Elastic Compute Cloud (AWS EC2), including both GPU-enabled (p3.2xlarge) and CPU-only (t3.large) instances as cloud nodes and 30 compute nodes managed by SLURM, equipped with NVIDIA Quadro RTX 6000 GPUs with 24 GB VRAM and 4608 CUDA cores, as well as CPU-only nodes as HPC nodes.

.

For the configuration of framework we have a communication backend that utilizes gRPC for cloud environments and MPI for HPC setups. Algorithms such as FedAvg and FedProx is supported and 20 clients are randomly selected in each training rounds while there are 100 training rounds for entire training process. Each clients perform 5 local epochs of training before sending their updates for aggregation.

5.2. Datasets

We evaluated our framework on the following benchmark datasets, each partitioned to simulate non-IID data distributions across clients:

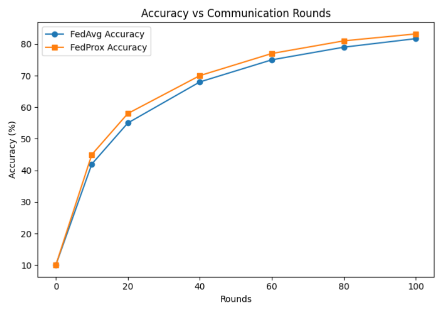
- **CIFAR-10:** Image classification with 10 classes. Each client receives samples from only 2–3 classes.
- **Shakespeare (LEAF):** A character-level language modeling task on dialogue data.
- **MedMNIST:** Medical image classification task simulating privacy-sensitive healthcare settings.

5.3. Evaluation Metrics

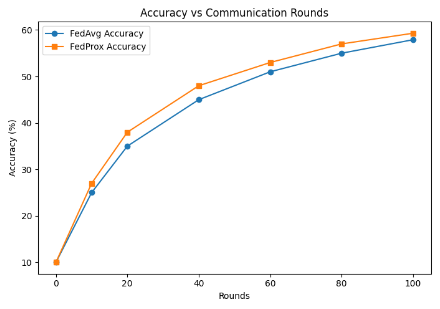
We used the following metrics to evaluate system performance:
Model Accuracy: Test accuracy on a centralized evaluation dataset.
Convergence Speed: Number of rounds to reach a defined accuracy threshold.
Training Time: End-to-end wall-clock time per round and total training time.
Communication Overhead: Average data transmitted per client per round.
Fault Tolerance: Model accuracy under simulated client dropouts or preemptions.

5.4. Results Summary

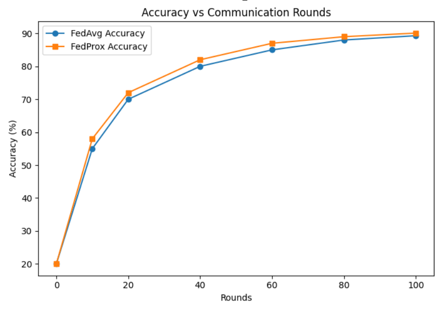
Accuracy and Convergence: Our system achieved competitive model accuracy across all datasets. Under non-IID conditions, FedProx showed improved convergence stability over FedAvg. The dataset-specific results graphs are shown in Figure 2 and numbers are given in table 2.



(a) CIFAR-10



(b) Shakespeare



(c) MedMNIST

Figure 2: Accuracy comparison of FedAvg and FedProx across different datasets under non-IID settings.

Table 2: Accuracy Comparison Between FedAvg and FedProx

Dataset	FedAvg	FedProx
CIFAR-10	81.7%	83.2%
Shakespeare	57.9%	59.3%
MedMNIST	89.3%	90.1%

Scalability: Increasing the number of clients from 10 to 60 showed near-linear improvements in training throughput, achieving a 4.6× speedup in total training time which is tablulated in Table 3 .

Table 3: Scalability Analysis with Varying Number of Clients

Number of Clients	Total Time (min)	Speedup (×)
10	100	1.00
20	58	1.72
30	43	2.32
40	33	3.03
50	27	3.70
60	22	4.55

Straggler Resilience: With 20% simulated client dropouts per round, the final accuracy dropped by less than 1.8%, demonstrating strong fault tolerance.

Communication Efficiency: Gradient quantization and sparsification reduced the average communication volume by 65% without significant accuracy degradation shown in Table 4.

Table 4: Communication Volume Per Round With and Without Compression

Round	No Compression (MB)	With Compression (MB)
1	45	16
2	44	15
3	43	14
4	44	15
5	43	14
6	42	14
7	44	15
8	43	14
9	42	13
10	43	14

5.5. Ablation Studies

To assess the contribution of each optimization, we conducted ablation experiments disabling one component at a time. When the experiment was done without adaptive client selection we saw 12% increase in average round duration and without communication compression there was 70% increase in bandwidth usage. Similarly when experiment was performed without straggler mitigation we saw 15–20% longer training time to reach 80% accuracy. These results confirm that heterogeneity-aware optimizations are essential to maintain both efficiency and reliability in real-world deployments.

6. Discussion, Limitations and Future Works

Our experimental results demonstrate that federated learning can be made both scalable and practical in heterogeneous environments by carefully addressing system-level challenges. The integration of adaptive scheduling, communication-efficient updates, and fault tolerance mechanisms enables reliable training across cloud and HPC infrastructure—even under non-IID data conditions.

The ability to run federated learning over hybrid compute environments has strong implications for industries where data is siloed across institutions or departments (e.g., healthcare, finance, scientific research). Our system design shows that FL is no longer limited to edge devices or simulated environments but can be deployed at enterprise and cluster scale.

Heterogeneity is not a flaw—it's a feature. Rather than treating hardware diversity as a limitation, our system embraces it. By tailoring workloads based on node capability and using adaptive aggregation, slower or lower-power nodes can still contribute without holding back global progress.

While compute is abundant, bandwidth often is not—especially in cloud-based federated setups. Our results show that communication optimizations such as quantization and sparsification significantly reduce training costs without affecting model quality, making FL more feasible in real-world deployments.

Training in distributed environments means expecting failure. Our system showed strong tolerance to client dropouts, but future work must address adversarial behavior and secure aggregation to make FL trustworthy on a scale.

Despite promising results, some challenges remain. The current system assumes relatively static client availability during each round. Highly dynamic workloads (e.g., serverless or elastic computing) would benefit from runtime-aware orchestration. Additionally, our experiments focus on moderate-scale clusters; scaling to hundreds or thousands of nodes introduces new coordination and aggregation bottlenecks that require further study.

Furthermore, We still have several directions that remain open for exploration:

- **Secure aggregation:** Incorporating cryptographic techniques (e.g., differential privacy, homomorphic encryption) to enhance data confidentiality.
- **Elastic orchestration:** Designing a more dynamic scheduling engine that reacts to real-time availability and pricing of cloud and HPC nodes.
- **Federated inference:** Extending the current training framework to support decentralized, real-time inference across the client network.
- **Integration with foundation models:** Adapting the system to fine-tune and serve large-scale pre-trained models (e.g., Large Language Models (LLMs)) in federated scenarios.

7. Conclusion

This paper presented a federated learning framework designed to operate across heterogeneous high-performance computing and cloud environments. By combining adaptive client selection, communication-efficient updates, and fault-tolerant aggregation strategies, our system achieves robust performance even under challenging conditions like system variability and non-IID data.

Through empirical evaluations on hybrid cloud-HPC testbeds, we showed that the proposed approach delivers strong accuracy, scalability, and resilience. The results underline that federated learning is not only viable but advantageous in multi-infrastructure environments where data privacy and system diversity are critical.

Our work represents a step toward scalable, private, and intelligent distributed learning systems that can thrive in the heterogeneous computing ecosystems of the future.

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," 2023. [Online]. Available: https://arxiv.org/abs/1602.05629

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. 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. T. 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," 2021. [Online]. Available: https://arxiv.org/abs/1912.04977

[3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020. [Online]. Available: https://arxiv.org/abs/1812.06127

[4] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020. [Online]. Available: https://arxiv.org/abs/2002.06440

[5] Tensorflow federated. [Online]. Available: https://www.tensorflow.org/federated

[6] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," 2020. [Online]. Available: https://arxiv.org/abs/2007.13518

[7] P. Barros, M. Oliveira, O. Orang, F. Silva, F. Erazo-Costa, A. Bastos, P. Silva, G. Santos, A. Loureiro, M. Ravetti, M. Costa, F. Guimarães, and H. Ramos, "Flautim: A federated learning platform using k8s and flower," in *Anais Estendidos do XXX Simpósio Brasileiro de Sistemas Multimídia e Web*. Porto Alegre, RS, Brasil: SBC, 2024, pp. 87–90. [Online]. Available: https://sol.sbc.org.br/index.php/webmedia_estendido/article/view/30484

[8] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," 2019. [Online]. Available: https://arxiv.org/abs/1812.01097

[9] F. Lai, Y. Dai, S. S. Singapuram, J. Liu, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning at scale," 2022. [Online]. Available: https://arxiv.org/abs/2105.11367

[10] U.-U. Haus, S. Narasimhamurthy, M. S. Perez, D. Pleiter, and A. Wierse, "Federated HPC, Cloud and Data Infrastructures," Hewlett Packard Enterprise and Seagate and Universidad Politécnica de Madrid and KTH Royal Institute of Technology and SICOS BW GmbH, White paper, 2024. [Online]. Available: https://issuu.com/etp4hpc/docs/etp4hpc_wp_federated-hpc-cloud-date-infra_final

[11] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," 2019. [Online]. Available: https://arxiv.org/abs/1804.05271

[12] C. Huang, J. Huang, and X. Liu, "Cross-silo federated learning: Challenges and opportunities," 2022. [Online]. Available: https://arxiv.org/abs/2206.12949

[13] M. Li, J. Zeng, and Y. Xiong, "Federated learning on non-IID data with reinforcement learning and quantifying historical contribution," in *4th International Conference on Internet of Things and Smart City (IoTSC 2024)*, F. Falcone and X. Yao, Eds., vol. 13224, International Society for Optics and Photonics. SPIE, 2024, p. 132241C. [Online]. Available: https://doi.org/10.1117/12.3034982

[14] J. Shin, Y. Li, Y. Liu, and S.-J. Lee, "Fedbalancer: data and pace control for efficient federated learning on heterogeneous clients," in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 436–449. [Online]. Available: https://doi.org/10.1145/3498361.3538917

[15] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2020. [Online]. Available: https://arxiv.org/abs/1712.01887

[16] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," 2020. [Online]. Available: https://arxiv.org/abs/2001.04756

[17] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. R. Cadambe, "Local sgd with periodic averaging: Tighter analysis and adaptive synchronization," 2020. [Online]. Available: https://arxiv.org/abs/1910.13598

[18] A. Goren, N. Lang, N. Shlezinger, and A. Cohen, "Adaptive deadline and batch layered synchronized federated learning," 2025. [Online]. Available: https://arxiv.org/abs/2505.23973

[19] J. Liu, J. H. Wang, C. Rong, Y. Xu, T. Yu, and J. Wang, "Fedpa: An adaptively partial model aggregation strategy in federated learning," *Computer Networks*, vol. 199, p. 108468, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621004199