Spread+: Scalable Model Aggregation in Federated Learning With Non-IID Data

Huanghuang Liang, Xin Yang, Xiaoming Han, Boan Liu, Chuang Hu, Dan Wang, *Senior Member, IEEE*, Xiaobo Zhou, *Senior Member, IEEE*, and Dazhao Cheng, *Senior Member, IEEE*

***Abstract*—Federated learning (FL) addresses privacy concerns by training models without sharing raw data, overcoming the limitations of traditional machine learning paradigms. However, the rise of smart applications has accentuated the heterogeneity in data and devices, which presents significant challenges for FL. In particular, data skewness among participants can compromise model accuracy, while diverse device capabilities lead to aggregation bottlenecks, causing severe model congestion. In this article, we introduce Spread+, a hierarchical system that enhances FL by organizing clients into clusters and delegating model aggregation to edge devices, thus mitigating these challenges. Spread+ leverages hedonic coalition formation game to optimize customer organization and adaptive algorithms to regulate aggregation intervals within and across clusters. Moreover, it refines the aggregation algorithm to boost model accuracy. Our experiments demonstrate that Spread+ significantly alleviates the central aggregation bottleneck and surpasses mainstream benchmarks, achieving performance improvements of 49.58% over FAVG and 22.78% over Ring-allreduce.**

***Index Terms*—Federated learning, edge computing, communication efficiency, model aggregation, scalability, tiering.**

I. Introduction

THE rapid advancement in communication technologies and the proliferation of intelligent devices are generating unprecedented volumes of data at the network edge. Traditional machine learning frameworks, which typically centralize data for processing, are increasingly inadequate in managing this influx efficiently. This centralization not only burdens network capacity, increases communication costs, and prolongs model training times, but also poses significant challenges for latency-sensitive applications such as autonomous driving and real-time healthcare. Furthermore, consolidating data processing in one location increases privacy risks, increasing the potential for unauthorized information disclosure and complicating the adherence to stringent data security regulations [2]. Addressing these challenges is crucial for maximizing the potential of edge computing while safeguarding user privacy and ensuring real-time responsiveness in critical applications.

The rise of edge computing and increasing data privacy concerns have spurred the adoption of the FL paradigm, which facilitates collaborative data-oriented tasks without the need to upload raw data. In FL, the parameter server (PS) architecture is widely utilized, where distributed servers train datasets locally and exchange updates—typically gradients—with centralized parameter servers. This architecture allows for the distribution of model training computation across various workers, thereby reducing overall training time. However, while PS servers effectively aggregate and centralize models, they are less impacted by congestion compared to model training computation, often due to worker shortages in data centers. In scenarios involving numerous edge devices, the operational nodes significantly outnumber the data center capacities. Each device handles relatively small data quantities, leading to delays in model aggregation that can extend beyond the computation time for model training. This aggregation congestion can subsequently delay the overall model training process. To address these issues, researchers have proposed communication scheduling strategies to streamline model updates and alleviate aggregation congestion [3].

Several studies have explored gossip learning (GL) [4], a decentralized method where workers share models only with their neighbors. Research indicates that inadequate model synchronization per iteration significantly degrades training quality in GL, particularly as the number of workers increases [5]. This is due to the ad hoc nature of GL, which can diminish model accuracy and synchronization across the network [6]. In contrast, the hierarchical architecture effectively enhances synchronization and reduces aggregation congestion [7].

However, challenges persist, particularly with Non-IID data distribution among clients, which is common in practical scenarios. Inappropriate clustering techniques can exacerbate data

Received 4 August 2024; revised 17 January 2025; accepted 4 February 2025. Date of publication 18 February 2025; date of current version 3 March 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62302348 and Grant 62341410, in part by National Key Research and Development Program of China under Grant 2023YFE0205700, in part by Science and Technology Development Fund, Macao S.A.R (FDCT) project under Grant 0078/2023/AMJ, and in part by UM projects under Grant 6SKL-IOTSC-2024-2026 and Grant CPG2024-00022-IOTSC. An earlier version of this paper was presented at the ACMICPP ACM International Conference on Parallel Processing [DOI: 10.1145/3545008.3545030]. Recommended for acceptance by S. Wang. *(Corresponding authors: Chuang Hu; Dazhao Cheng.)*

Huanghuang Liang is with the School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China, and also with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Taipa 999078, Macau (e-mail: hhliang@whu.edu.cn).

Xin Yang, Xiaoming Han, Boan Liu, and Dazhao Cheng are with the School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China (e-mail: xingyang@whu.edu.cn; hanxiaoming@whu.edu.cn; boanliu@whu.edu.cn; dcheng@whu.edu.cn).

Chuang Hu and Xiaobo Zhou are with the State Key Laboratory of Internet of Things for Smart City and Faculty of Science and Technology, University of Macau, Taipa 999078, Macau (e-mail: chuanghu@um.edu.mo; waynexzhou@um.edu.mo).

Dan Wang is with the Department of Computing, Hong Kong Polytechnic University, Kowloon 999077, Hong Kong (e-mail: csdwang@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TPDS.2025.3539738

unevenness, leading to decreased accuracy and slower convergence in federated learning models [8]. Recent studies have dissected the problem of forming clusters by dissecting the problem of forming clusters by customers into a kind of hedonic coalition formation (HCF) game [9], [10], but they tend to ignore the communication congestion problem in the formation of clusters in the pursuit of model performance.

To address data heterogeneity, minimize model aggregation congestion, and ensure efficient task execution, we introduce Spread+, a hierarchical framework for distributed model aggregation. Spread+ utilizes a cloud server as the federated learning (FL) coordinator and organizes edge devices into clusters, designating selected devices as cluster heads. This structuring deviates from the parameter server (PS) architecture by delegating model aggregation to specific workers, thereby reducing congestion. Unlike the gossip learning (GL) architecture, Spread+ implements structured model aggregation to maintain high training quality.

Developing such an architecture, however, presents two primary challenges: (1) Constructing clusters to balance accuracy with communication efficiency amidst Non-IID data. This involves grouping clients with similar data distributions to reduce data heterogeneity and selecting cluster heads based on an analysis of communication overhead. (2) Implementing model training within a tiered system to effectively minimize model aggregation congestion. To address this, we have developed an adaptive runtime algorithm that manages model synchronization both within clusters (between cluster heads and edge devices) and across clusters (between the cloud server and cluster heads). This algorithm aims to reduce congestion during aggregation and shorten the overall duration of model training.

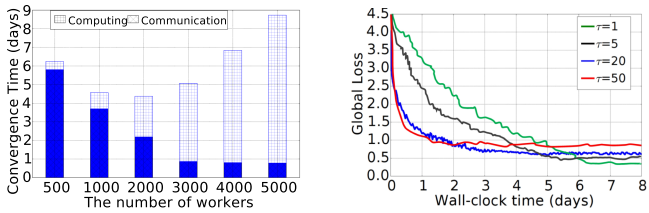
In summary, the contributions of this paper are:

First, We address the scalability issues prevalent in existing ML systems when applied to FL applications. Our study highlights the significant delays caused by congestion when combining models and demonstrates that an inappropriate cluster organization strategy reduces the effectiveness of training (Sections II-A, II-B).

Second, We introduce Spread+, a distributed architecture that decentralizes model aggregation to improve scalability and efficiency. This includes a directed clustering algorithm, which formalizes the clustering problem as a hedonic coalition game. Additionally, it features a modular design for runtime operations and an adaptive algorithm to fine-tune training parameters, with verified convergence results (Sections III, III-B, III-D, and IV-A).

Third, We present a fully functional system developed using Google's FL framework, extensively evaluated in simulated environments and real-world settings using actual smartphone data. Additionally, we provide two case studies showcasing the practical application of Spread+ (Sections V, VI, and VII).

Compared with the previous conference version of our work [1], this journal paper presents three significant enhancements: 1) By formalizing the problem as a hedonic coalition game, we have optimized the cluster organization strategy, thereby enhancing the hierarchical architecture. 2) We expanded the experimental scope by including four additional baselines, providing a more comprehensive evaluation. 3) We illustrate the system's practical applicability through two new case studies: "Smart Object Detection for Self-Driving Cars" and "Fault Detection in Industrial Equipment." These additions demonstrate the feasibility and practical utility of our system.



(a) Scalability problem. (b) Aggregation intervals.

Fig. 1. The impact of model aggregation.

II. Background and Motivation

A. Worker Scaling and Aggregation Congestion

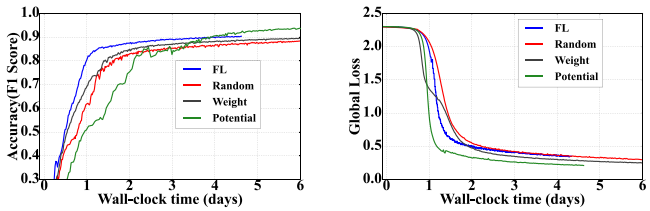
We conducted targeted experiments to assess two critical aspects: 1) congestion-related delays during model aggregation and 2) model accuracy degradation due to insufficient model synchronization. These experiments used 251 Amazon EC2 instances, in which CNNs were trained using the CIFAR-10 dataset.

In Fig. 1(a), we divide the convergence time of the model into computation time and communication time. Training with 1000 workers took 4.57 days, consisting of 3.7 days for computation and 0.87 days for aggregation. When the number of workers increased to 5000, the computation time decreased to 0.79 days, but the aggregation time rose dramatically to 7.94 days. This illustrates that while the addition of more workers reduces computation time, it leads to longer durations due to server aggregation congestion. Fig. 1(b) analyzes training quality with different worker counts and aggregation intervals. For example, with 3000 workers and a fully synchronized federated learning (FL) model at an aggregation interval of one, training lasted 7.12 days and achieved an accuracy of 92% (global loss of 0.4). However, increasing the interval to 50 reduced the training time to 3.76 days, but also decreased the accuracy to 80% (global loss of 0.92). Due to less frequent model synchronization, longer aggregation intervals minimize congestion but lower training accuracy.

B. Potential Approach and Problems

A potential solution to the communication congestion problem is to organize workers into clusters, creating a tiered system structure. We continued our experiments around cluster organization strategies.

In Fig. 2, *Random* refers to clusters formed randomly among clients with equal probability, while *Weight* refers to clusters based on data possession. In addition, we tried to manually organize the formation of the groups to maximize the similarity of the data among the clients. After several rounds of attempts, the curve that yields the most optimal outcomes is the *Potential*



(a) Accuracy. (b) Global loss.

Fig. 2. The impact of cluster organization strategy.

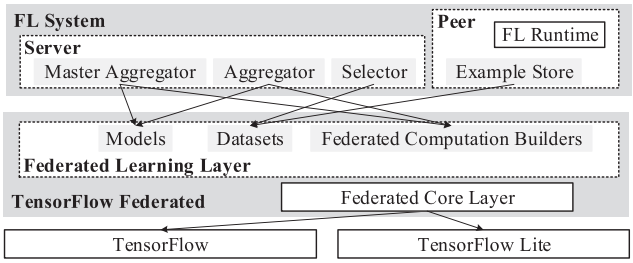


Fig. 3. The implementation of FL platform.

curve. Fig. 2(a) shows that the *Random* and *Weight* methods have become less accurate than FL, indicating a worsening of the non-IID problem. Instead, potential indicates an increase in precision, suggesting that cluster arrangement can improve training results. Fig. 2(b) shows the challenges of maximizing training efficiency. Although *Potential* has improved accuracy, it takes longer to train than FL, and communication congestion reduces efficiency. Thus, a clustering algorithm that solves Non-IID problems and improves communication is essential. These measurements provide further clarification for our design concepts described in Section III.

C. The Current FL System

Google has established a functional FL platform using an open source TensorFlow Federated (TFF). Also, an example FL system was presented that performs model training by coordinating edge devices [11], see Fig. 3.

TFF exploits the well-developed ML systems TensorFlow and TensorFlow Lite. TFF consists of two layers: 1) The Federated Core Layer: TensorFlow and TensorFlow Lite are tailored for FL applications by this layer. Robustly typed functional programming is combined with distributed communication operators, TensorFlow, and TensorFlow Lite. Federated algorithm customization is made easier by integrating fundamental interfaces. 2) High-level interfaces for the integration of the ML model into the TFF ecosystem are provided by the FL layer. There are three primary classes in it: a) Models: These interfaces enable update management, including aggregation, and model incorporation. b) Datasets: Training data is processed effectively by these interfaces. c) Building Federated Computation constructs federated computations related to training. Through the FL layer's interfaces, FL application developers can effectively carry out fundamental FL tasks without diving into FL algorithms.

A FL system consists of worker nodes and a server. The *Aggregator* arranges worker execution, the *Selector* oversees server-worker connections, and the *Master Aggregator* controls the aggregation process in FL. The *FL Runtime* handles training operations, and the *Example Store* handles training data management on the worker side. Utilizing TFF's FL layer, as shown in Fig. 3, simplifies FL system setup. "Clusters" and "cluster heads" are not included in FL. The learning step and aggregation interval are examples of static training parameters. Spread+ enhances these and related algorithms.

III. Design

A. Overview

Spread+ features a hierarchical design, illustrated in Fig. 4(a), comprising a cloud server, multiple cluster heads, and many workers. Each worker is equipped with local data and an individual model known as the edge model. Similarly, each cluster head possesses a model called the *cluster model*. Before training, the system employs a directed cluster organization algorithm to merge workers with comparable data distributions. The training proceeds iteratively, where workers update their models using local data and the edge model through an optimization function. Within Spread+, cluster heads periodically perform intra-cluster aggregation. The period between successive intra-cluster aggregations is termed the *intra-cluster aggregation interval*, denoted as k_1; The server performs inter-cluster aggregation at regular intervals, termed the *inter-cluster aggregation interval*, denoted as k_2.

The key steps in the training process are listed below 1) The Server initiates cluster construction; 2) The server calculates both intra/inter-cluster aggregation intervals k_1 and k_2, subsequently sending k_1, k_2, and the global model to the cluster heads; 3) Cluster heads align their cluster models with the global model and dispatch k_1 and the cluster model to their subordinate workers; 4) Workers adjust their edge models to match the received cluster model, then perform k_1 iterations for edge model updates before sending the edge model back to their cluster head; 5) After receiving edge models from all subordinate workers, a cluster head conducts intra-cluster aggregation to combine the edge models of its workers and then sends the revised cluster model back to them; 6) Following k_2 intra-cluster aggregation cycles, cluster heads forward the cluster model to the server, which then performs inter-cluster aggregation to refresh the global model. These steps are reiterated until the global model reaches convergence.

B. Hedonic Coalition Formation Game

1) Problem Formulation: In an HCF game with player(Also known as a worker) set $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$, $C \subseteq \mathcal{N} \neq \emptyset$ is called a coalition on $\mathcal{N}$. An HCF structure of $\mathcal{N}$ is a collection $\Gamma = \{C^1, C^2, \ldots, C^M\}$ such that $\bigcup_{i=1}^{M} C^i = \mathcal{N}$. And for any $n_i, n_j, C^i \cap C^j = \emptyset$. These players are then organized as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{L}$ is the set of links. A link $(n_i, n_j) \in \mathcal{L}$ represents the connection of worker n_i and n_j. The degree sequence of the graph $\mathcal{G}$ is denoted as $\{d_i\}_{i=1}^{N}$.

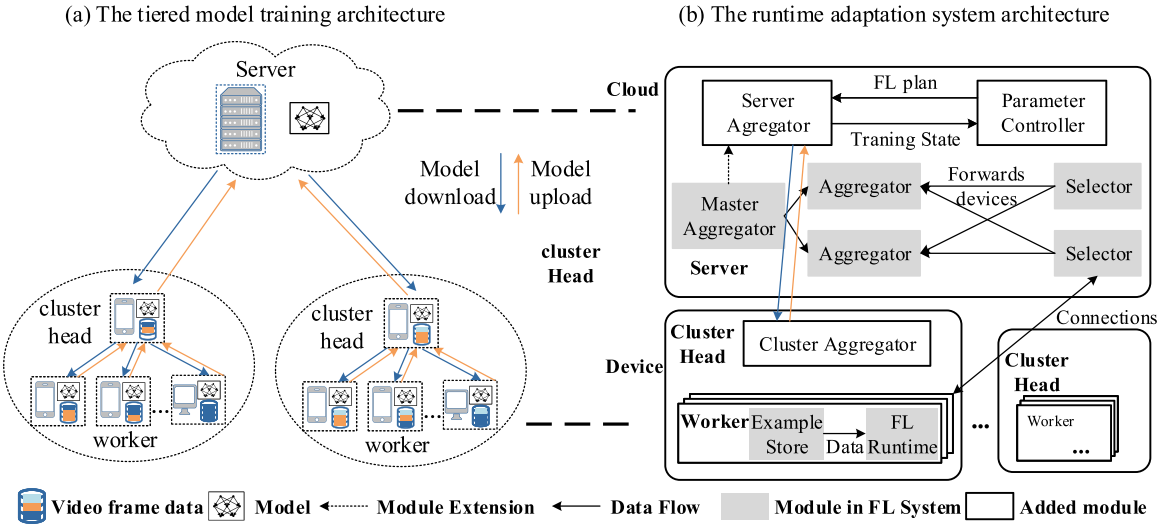


Fig. 4. The overview of training architecture.

1)Player Payoff: In an HCF game, each player participating in a coalition would expect to receive a positive payoff. The payoff of player n_i in coalition C^j is the difference between the error obtained from its local model and the error obtained from the coalition mode, which is formalized as

$$R_i(C^j) = err_i(\omega_i) - err_i(\omega^j) \tag{1}$$

2)Individual Rationality: Individual rationality states that an individual always chooses the action that maximizes its utility. We assume that each player always tends to join the coalition with the highest payoff in each round, i.e., the coalition chosen by player n_i satisfies

$$R_i(C^j) \geq R_i(C^k), \forall C^k \in \Gamma, j \neq k \tag{2}$$

3)Joining Rule: Considering the overall payoff, players cannot join their preferred coalition at will. Player n_i can be incorporated into the coalition C^j only if the worker gains a positive benefit from joining and the utility of existing participants in the coalition does not decrease. This condition can be formally expressed as:

$$\begin{cases} R_i(C^j \cup i) > R_i(\{i\}) \\ R_\alpha(C^j \cup i) \geq R_\alpha(C^j), \forall \alpha \in C^j \end{cases} \tag{3}$$

4)Communications Condition: In practice, due to the differences in network conditions, there exist some workers that cannot communicate with each other, in other words, graph $\mathcal{G}$ is not a complete graph.

2) Directed Cluster Formation Algorithm: In the case where $\mathcal{G}$ is not a complete graph, organizing workers into clusters prioritizes whether the cluster head can communicate properly with the workers in the cluster. Therefore, in order to construct a hierarchical federated learning architecture based on HCF game, it is necessary to ensure that at least one central node in the divided subgraph can be directly connected to other nodes, i.e., the subgraph is a star-shaped cluster graph.

This star-shaped cluster partitioning problem is NP-hard, considering similarity in cluster partitioning, connectivity in cluster head selection and combinatorial optimization problem to minimize communication overhead. It can be proved by transforming the problem into the vertex-cover problem [12]. A similar problem has been studied by several works [13], [14]. Due to the impracticality of identifying the global optimal solution within polynomial time, we present a heuristic solution called directed cluster organization algorithm.

At the beginning of each iteration, we choose the point with the highest degree as the potential cluster head according to the ranking of degree sequence. Nodes with high degree can better cover the surrounding leaf nodes, thus speeding up cluster construction. The node and its neighbor nodes (or clusters) are selected into the candidate set $\mathcal{N}^*$. Upon the formation of the star-shaped cluster, multiple central nodes may exist, necessitating the selection of the most appropriate node to function as the cluster head. The end-to-end bandwidth from worker n_i and worker n_j is denoted as b_{ij}. To simplify the problem, we assume that $b_{ij} = b_{ji}$. When workers n_i and n_j cannot communicate with each other, b_{ij} equals 0. Model aggregation is a procedure that entails a swift averaging operation. The duration of the averaging operation executed by workers can be regarded as consistent for a specific model and its corresponding model aggregation algorithm. Let v_j represents the time taken by the cluster head h_j for model averaging.

We proceed to evaluate the additional costs associated with model aggregation at cluster heads. Let Q represent the size of the model. For each intra-cluster aggregation, the cluster head h_j, gathers the edge models from its workers, calculates their average to update the cluster model, and subsequently distributes this updated cluster model back to the workers. The computation and communication overhead of the cluster head h_j of C^j is:

$$l_j(C^j) = \sum_{k \in C^j} \frac{2Q}{b_{jk}} + v_j \tag{4}$$

The pseudo code provided in Algorithm 1 presents a clear and concise outline of our method for creating directed clusters.

Algorithm 1: HCF() Pseudo Code.

Input: graph $\mathcal{G}$, degree sequence $\{d_i\}_{i=1}^{N}$
Output: structure Γ
1 Initialize $\Gamma \leftarrow \{\{i\}\}_{i=1}^{N}$; $flag_i \leftarrow 0$; $d_i' \leftarrow d_i$;
2 **while** $\Gamma^t \neq \Gamma^{t-1}$ **do**
3 **for** $C^j \in \Gamma$ **do**
4 Sort and select the minimum $l_i(C^j)$ for cluster head;
5 FL training to generate model ω^j;
6 Broadcast Γ^{t-1};
7 $i \leftarrow \arg\max d_i'$;
8 $\mathcal{N}^* = \{k | k \in \mathcal{N}, flag_k == 0, b_{ik} \neq 0\}$;
9 **for** *client* $i \in \mathcal{N}^*$ *in parallel* **do**
10 $\Gamma_{\setminus i} = \{C^j | i \notin C^j, b_{ij} \neq 0\}$;
11 $C^\alpha \leftarrow \min_{C^\alpha \in \Gamma_{\setminus i}} \sum_{i \in D_i} f(\omega^\alpha)$;
12 **if** *Eq. (3) is satisfied* **then**
13 client n_i join cluster C^α ;
14 $flag_i \leftarrow 1$;
15 Update d_i';
16 Update Γ^t;
17 return Γ;

During the initialization phase, as indicated in line 1, each worker forms an individual alliance and maintains a flag variable to indicate whether it has joined a cluster. Only workers with a flag value of 0 will make an effort to join a cluster. At the start of each iteration, the current cluster selects the cluster head based on the cost ordering of communication and computation, and goes through a process to obtain the cluster model for that iteration. The updated cluster structure information is then distributed to the workers. Traversing the neighboring nodes of the worker is prioritized with the largest degree in order to be able to speed up the cluster formation. On the basis of assurances of communication availability, each worker that has not joined a cluster assesses the unjoined clusters based on their estimated expected error, which is derived from the cluster model. The process is depicted in lines 10 to 11. Subsequently, they evaluate whether affiliating with the alliance that has the lowest joining error will yield superior benefits in comparison to local training. According to lines 12 to 13, a worker will only join the cluster if it results in a favorable outcome and does not decrease the overall satisfaction of the existing members. The cluster configuration is updated after all players have made their final selections. This process may require multiple iterations until the cluster structure achieves a state of stability.

3) The Model Aggregation Algorithm: FedAVG is one of the most widely used FL aggregation algorithms. However, this simple weighted average aggregation algorithm may produce suboptimal results and affect the convergence speed when faced with non-IID data. Therefore, we consider an optimized general framework FedNova [15], which deals with the deviation of the size and distribution of the local training data by correcting the model, in other words, it can have better model accuracy and convergence speed under the influence of Non-IID data. And its generality means this can support optimization methods such as momentum acceleration [16], adding proximal terms [17], etc., and these are also in our future work plan.

Consider ω, ω^j, and ω_i^j as the global model, the cluster model for $\mathcal{C}^j$, and the edge model for worker n_i^j, respectively. Define $\mathcal{D}_j$ and $\mathcal{D}_i^j$ as the dataset in cluster $\mathcal{C}^j$, and the dataset in worker n_i^j. The weight of the worker n_i^j is denoted by $p_i^j = \frac{|\mathcal{D}_i^j|}{|\mathcal{D}^j|}$, while the weight of the cluster $\mathcal{C}^j$ is denoted by $p^j = \frac{|\mathcal{D}^j|}{|\mathcal{D}|}$. The number of local iterations for worker n_i^j is denoted by τ_i^j, while the number of iterations for cluster j is denoted by τ^j. We express the edge loss $F_i^j(\omega) = \sum_{i \in \mathcal{D}_i^j} f_i(\omega)$ in the edge device n_i^j, and the cluster loss is formalized as follows:

$$F^j(\omega) = \left(\sum_{i \in \mathbb{C}_j} p_i^j \tau_i^j \right) \sum_{i \in \mathbb{C}_j} \frac{p_i^j F_i^j(\omega)}{\tau_i^j} \tag{5}$$

The global loss is formalized as follows:

$$F(\omega) = \left(\sum_{j \in \mathbb{C}} p^j \tau^j \right) \sum_{j \in \mathbb{C}} \frac{p^j F^j(\omega)}{\tau^j} \tag{6}$$

If t represents the count of local updates and $\nabla F(\omega)$ symbolizes the gradient, the parameter update rule can be articulated as $\omega(t) = \omega(t-1) - \eta \nabla F(\omega(t-1))$.

C. Runtime Adaptive Model Aggregation

1) The Model Training Problem: Training duration and quality (specifically, accuracy) stand as the primary metrics for assessing a model training system. Spread+, as an ML system, is designed to expedite model training while simultaneously upholding superior training quality. Detailed in Section II-A, it is evident that fully synchronous SGD is capable of attaining the utmost training accuracy. The challenge that Spread+ encounters is minimizing the time required for model training while still attaining an accuracy level comparable to that of fully synchronous SGD.

2) Jointly Analyzing Runtime and Error-Convergence: In this subsection, we jointly analyze the runtime and error-convergence of Spread+.

Runtime analyzing: Define $\zeta_{\tau_1,\tau_2}^{i,j}$ as the computation duration for worker i in Cluster j when computing τ_1-th iteration of τ_2-th inter-cluster aggregation period. The variable $\zeta_{\tau_1,\tau_2}^{i,j}$ are i.i.d. random variables following the probability distribution F_Y. Denote Y as the maximum value in a set of N i.i.d. variables. The delay in communication, a random variable denoted by P, arises during the synchronizing of the model with its headers. The aggregate time required for node i in cluster j to complete τ_2-th k_1 iteration is determined by

$$Z_{\tau_2}^{i,j} = \zeta_{1,\tau_2}^{i,j} + \zeta_{2,\tau_2}^{i,j} + \cdots + \zeta_{k_1,\tau_2}^{i,j} \tag{7}$$

The completion time for the τ_2-th intra-cluster aggregations of cluster j is given by

$$\zeta_{\tau_2}^{j} = \max\{Z_{\tau_2}^{1,j}, Z_{\tau_2}^{2,j}, \ldots, Z_{\tau_2}^{|\mathcal{C}_j|,j}\} + P \tag{8}$$

The completion time for the k_2 intra-cluster aggregations of cluster j is given by

$$Z^j = \zeta_1^j + \zeta_2^j + \cdots + \zeta_{k_2}^j \tag{9}$$

Once every cluster has finalized k_2 intra-cluster aggregations, an inter-cluster aggregation is initiated. Therefore, the overall time needed to complete an inter-cluster aggregation is:

$$T = \max\{Z^1, Z^2, \ldots, Z^M\} + P \tag{10}$$

$$E(T) = Z + \frac{P}{k_2} + \frac{P}{k_1 k_2} \tag{11}$$

Joint analysis with error-convergence: we integrate runtime analysis with error-convergence analysis specifically for Spread+. The essential theoretical assumptions for this are detailed in Section IV-A.

Theorem 1: In the context of Spread+, given Assumption 1, if the learning rate fulfills the condition $\eta\rho + \eta^2\rho^2 k_2(k_2 - 1) \leq 1$, with Y and D as constants, the initial model parameter w_0, then after total T wall-clock time, the minimal expected squared gradient norm within T time interval will be bounded by

$$\frac{2F(x_t)}{\eta T}\left(Y + \frac{P}{k_2} + \frac{P}{k_1 k_2}\right) + \frac{\eta\rho k_1 \sigma^2}{n} + \eta^2\rho^2 k_1 \sigma^2 (k_2 - 1) \tag{12}$$

where ρ represents the Lipschitz constant of the objective function, and σ^2 is indicative of the variance limit of mini-batch stochastic gradients.

The proof of Theorem 1 has similarities to the convergence proofs of local-update SGD algorithms, as discussed in [18]

Theorem 2: For Spread+, following Assumptions 1, the optimization error's upper bound at time T, as expressed in (12), is minimized when the intra-cluster aggregation interval k_1 and the inter-cluster aggregation interval k_2 are aligned according to the subsequent equations:

$$k_1 = \frac{2Pn^2 F(x_t)}{T\eta\sigma^2(1 - n\eta\rho)^2} - 1 \tag{13}$$

$$k_2 = \frac{2PnF(x_t)(1 - n\eta\rho)}{\eta^2\rho\sigma^2 T(1 - n\eta\rho)^2 - 2n\eta^2\rho F(x_t)} \tag{14}$$

The proof is straightforward by setting the derivative of (12) to 0.

3) The Adaptive Aggregation Interval Algorithm: To optimize the true function loss relative to wall-clock time, thereby reducing the model training duration while maintaining accuracy comparable to fully synchronous SGD (proved in [18]). The pivotal question here is the strategy for progressively shortening the aggregation intervals to optimize the function loss with respect to wall-clock time.

According to the analysis in Section III-C2, we can approximate intra-cluster aggregation interval k_1 as follow:

$$k_1 \approx \frac{2Pn^2 F(x_t)}{T\eta\sigma^2(1 - n\eta\rho)^2} \tag{15}$$

We have the intra-cluster aggregation interval k_1^j:

$$k_1^j = \left\lceil k_1^0 \frac{F^j(\omega_j)}{F^j(\omega_0)} \right\rceil \tag{16}$$

where k_1^0 is the initial intra-cluster aggregation interval, F^j and ω_j are the cluster loss function and the real-time cluster model of cluster $\mathcal{C}_j$, respectively.

Algorithm 2: `AAI()` Pseudo Code.

```
1 [k_1^0, F(ω_0), F^1(ω_0), F^2(ω_0), ..., F^M(ω_0), η] ← init();
2 [F^1(ω_1), F^2(ω_2), ..., F^M(ω_M)] ←collect_loss(ω);
3 F(ω) ←global_loss(F^1(ω), F^2(ω), ..., F^M(ω));
4 for l ← 1; l ≤ M; l ← l + 1 do
5     # according to (17);
6     k_1^j ←update_intra(k_1^0, F^j(ω_0), F^j(ω_j));
7     [P, σ] ← monitor_parameters();
8     # according to (14);
9     k_2^j ←update_inter(P, σ, T, η, n, F(ω))
```

Suggested by ADACOMM [18], the following global aggregation interval update rule can maximize the global function loss with respect to wall-clock time: $k = \lceil k^0 \sqrt{\frac{F(\omega)}{F(\omega_0)}} \rceil$, where k_0 is the initial global aggregation interval, F is the global loss function, ω_0 is the initial global model and ω is the real-time global model. Once the global aggregation interval is established, the ensuing task involves striking a balance between intra and inter-cluster aggregation intervals. It's observed that not only does maximizing global function loss per wall-clock time accelerate model training and preserve training quality, but enhancing cluster function loss per wall-clock time within each cluster contributes similarly. Thus, we have the intra-cluster aggregation interval k_1^j of cluster $\mathcal{C}^j$:

$$k_1^j = \left\lceil k_1^0 \sqrt{\frac{F^j(\omega_j)}{F^j(\omega_0)}} \right\rceil \tag{17}$$

where k_1^0 is the initial intra-cluster aggregation interval, F^j and ω_j are the cluster loss function and the real-time cluster model of cluster $\mathcal{C}_j$, respectively.

Based on the formula in (14), the optimal inter-cluster aggregation interval k_2 can be calculated when provided with the global function loss $(F(\omega))$, the communication delay P, and the variance associated with mini-batch stochastic gradients. Variables like $F(\omega)$, $F^j(\omega_j)$ (function loss within individual clusters), stochastic gradients, and the communication delay P are readily measurable during the training process. The remaining task involves determining the initial value of the intra-cluster aggregation interval, denoted as k_1^0. To estimate this parameter, we employ a heuristic approach, conducting a simple grid search. This involves trying out different values for k_1^0, each tested over one or two training epochs.

We introduce an adaptive aggregation interval (AAI) algorithm, as outlined in Algorithm 2, which utilizes the aforementioned adjustment strategy. Following an inter-cluster aggregation by the SA, both the global model and the inter/intra-cluster aggregation intervals are recalculated. Subsequently, the server forwards the global model to the cluster heads and workers and awaits the return of the cluster loss from the cluster heads (line 2). The server then recalculates the global loss using the cluster losses it has received (line 3). In the final step, the server calculates the intra/inter-cluster aggregation intervals for each cluster based on the equations in (17) and (14), detailed in lines 4 through 9.

D. A Modular Design for Runtime Training

The runtime adaption system of Spread+ is based on the FL system and follows a modular design as shown in Fig. 4(b). We explain the key modules here.

Parameter Controller: Spread+ utilizes a *parameter controller* at the server, responsible for adjusting the intra and inter-cluster aggregation intervals based on the current *training state*. This training state is a specialized data structure that logs the overall function loss as well as the cluster-specific function loss for each cluster. Instead of employing static aggregation intervals, Spread+ adopts a dynamic approach. The parameter controller, after each inter-cluster aggregation, collects the training state data and then applies an adaptive aggregation interval algorithm (elaborated in Section III-C). This process is essential for calculating the intra and inter-cluster aggregation intervals. Following this calculation, it formulates an *FL plan*. An FL plan is a data structure that records aggregation intervals.

Server Aggregator (SA): The SA is located in the server and has three main functions. Initially, it focuses on calculating the global function loss by gathering the cluster loss. This data is then used by the SA to determine the global loss, forming the training state that is subsequently relayed to the parameter controller. Second, upon receipt of the cluster models from the cluster heads, the SA employs a model aggregation algorithm to refresh the global model. Lastly, the SA is responsible for dispatching the newly updated global model along with the FL to the cluster heads. This action is crucial for overseeing the execution of the training tasks within each cluster head.

Cluster Aggregator (CA): Spread+ incorporates a CA at each cluster head, which performs three essential functions: 1) It calculates the cluster loss using the edge loss gathered from workers, and then transmits this cluster loss to the SA; 2) Upon receiving the edge models from its workers, the CA executes a model aggregation algorithm to refresh the cluster model; 3) The CA directs the workers to carry out the training tasks by dispatching both the cluster model and the FL plan to them.

FL Runtime: Spread+ utilizes an *FL runtime* in every worker to carry out the training tasks. Each worker, upon receiving an FL plan and a cluster model, undertakes a set number of iterations as defined by the intra-cluster aggregation interval, using its local data. During this process, it calculates the edge loss and subsequently sends these findings back to its respective cluster head.

Example Store: Applications in the workers make their data available to FL runtime as an *example store*.

IV. Theoretical Analysis

A. Convergence Analysis

In this section, we prove the convergence of Spread+ system.

For the purpose of the analysis, we make the following assumption about the loss function.

Assumption 1: (Convex) For any edge device n_i^j: 1) $F_i^j(\omega)$ is convex; 2) $F_i^j(\omega)$ is ρ-Lipschitz; and 3) $F_i^j(\omega)$ is β-smooth.

From Assumption 1, It is straightforward $F(\omega)$ is convex, ρ-Lipschitz and β-smooth. We define gradient divergence:

Definition 1: (*Gradient Divergence*) For any device n_i^j and ω, we define δ_i as an upper bound of $\|\nabla F_i^j(\omega) - \nabla F^j(\omega)\|$, and Δ^j as an upper bound of $\|\nabla F^j(\omega) - \nabla F(\omega)\|$, i.e.,

$$\|\nabla F_i^j(\omega) - \nabla F^j(\omega)\| \leq \delta_i$$
$$\|\nabla F^j(\omega) - \nabla F(\omega)\| \leq \Delta^j \tag{18}$$

We arrive at the *overall gradient divergence*, denoted as $\delta = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{N}|\mathcal{D}_i^j|\delta_i$, and $\Delta = \frac{1}{|\mathcal{D}|}\sum_{j=1}^{R}|\mathcal{D}^j|\Delta^j$. The total number of iterations, represented by T, is subdivided into P *global intervals*, each encompassing k_1k_2 in length. At the conclusion of each *global interval*, *inter-cluster aggregation* is executed. For the analysis of convergence, we introduce a reference parameter sequence $\hat{\omega}_p(t)$ for each global interval $[p]$. These parameters sequences are updated using the centralized gradient descent method previously mentioned, meaning, for the global interval $[p]$, starting from $\omega((p-1)k_1k_2)$ at step $(p-1)k_1k_2$, the parameters are updated following this rule:

$$\hat{\omega}_p(t) = \hat{\omega}_p(t-1) - \eta\nabla F(\hat{\omega}_p(t-1)) \tag{19}$$

Lemma 1: (Proved in work [19]) For any global interval $[p]$, assuming $t \in [p]$, k_1 and k_2 have an upper bound K, i.e., $k_1 \leq K, k_2 \leq K$, we have

$$\|\omega(t) - \hat{\omega}_p(t)\| \leq h(K^2, \Delta) + \frac{1}{2}(K^2 + K - 1)(K+1)h(K, \delta) \tag{20}$$

where $h(x, \upsilon) = \frac{\upsilon}{\beta}((\eta\beta+1)^x - 1) - \eta\upsilon x$.

Lemma 2: (Proved in work [20]). With Assumption 1, define the upper bound of the weight gap as R, then the convergence upper bound of *FAVG* with an aggregation interval τ after T iterations is given by:

$$F(\mathbf{w}(T)) - F(\mathbf{w}^*) \leq \frac{1}{T(\eta\gamma - \frac{\rho R}{\tau\varepsilon^2})} \tag{21}$$

when the following conditions are satisfied: 1)$\eta \leq \frac{1}{\beta}$; 2) $\eta\gamma - \frac{\rho R}{\tau\varepsilon^2} > 0$; 3) $F(\hat{\omega}_p(t)) - F(\omega^*) \geq \varepsilon$ for all t and p for which $\hat{\omega}_p(t)$ is defined; and 4) $F(\omega(T)) - F(\omega^*) \geq \varepsilon$, where $\varepsilon > 0, \gamma = (1 - \frac{\beta\eta}{2})\min_p \frac{1}{\|F(\hat{\omega}_p((p-1)k_1k_2)) - F(\omega^*)\|}$.

Theorem 3: (Convex) The convergence upper bound for Spread+ after T iterations is given by

$$F(\omega(T)) - F(\omega^*) \leq \frac{1}{T(\eta\varphi - \frac{\rho G(K^2)}{K^2\varepsilon^2})} \tag{22}$$

when the following conditions are satisfied: 1)$\eta \leq \frac{1}{\beta}$; 2) $\eta\varphi - \frac{\rho G(K^2)}{K^2\varepsilon^2} > 0$; 3) $F(\hat{\omega}_p(t)) - F(\omega^*) \geq \varepsilon$ for all t and p for which $\hat{\omega}_p(t)$ is defined; and 4) $F(\omega(T)) - F(\omega^*) \geq \varepsilon$, where $\varepsilon > 0, \gamma = (1 - \frac{\beta\eta}{2})\min_p \frac{1}{\|F(\hat{\omega}_p((p-1)k_1k_2)) - F(\omega^*)\|}$.

Proof: Since $k_1 \leq K$ and $k_2 \leq K$, we have

$$\gamma \geq \min_p \frac{1}{\|F(\hat{\omega}_p((p-1)K^2)) - F(\omega^*)\|} = \varphi \tag{23}$$

Then, by substituting R in Lemma 2 with the upper bound of the weight gap for Spread+ $G(K^2)$, we prove this theorem.

We provide the convergence analysis of Spread+ and prove $O(\frac{1}{\sqrt{T}})$ convergence rate in T iterations under the assumption of non-convex.

Assumption 2: (Non-Convex) For any edge device n_i^j: 1) $F_i^j(\omega)$ is non-convex; 2)$F_i^j(\omega)$ is ρ-Lipschitz; and 3) $F_i^j(\omega)$ is β-smooth.

Assumption 3: (Non-IID data) Bounded gradient dissimilarity: there exist constants G $\geq$ 0 and B $\geq$ 1 such that $\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(\omega)\|^2 \leq G^2 + B^2\|\nabla f(\omega)\|^2, \forall x$.

Lemma 3: (Proved in work [19]) For any global interval $[p]$, assuming $t \in [p]$, k_1 and k_2 have an upper bound K, i.e., $k_1 \leq K, k_2 \leq K$, we have

$$\begin{aligned}\|\omega(t)-\hat{\omega}_p(t)\| &\leq h(K^2,\Delta)+(K^2(1+\eta\beta)^k+k^2+1)h(K,\delta)\\ &= \sigma^2\end{aligned} \tag{24}$$

where $h(x,\upsilon) = \frac{\upsilon}{\beta}((\eta\beta+1)^x - 1) - \eta\upsilon x$.

Lemma 4: (Proved in work [21]) For each communication round t, the gap between $\mathbb{E}F(\omega(t+1)) - F(\omega(t))$ is bounded as below, where $\mathbb{E}F(\omega(t+1))$ is the expectation of $F(\omega(T+1))$ with respect to the mini-batch.

$$\begin{aligned}\mathbb{E}F(\omega(t+1))-F(\omega(t)) \leq &\left(\frac{\beta\eta^2}{2}K^2B^2-\frac{\eta}{2}\right)\mathbb{E}\|\nabla F(\omega(t))\|^2\\ &+\frac{\beta\eta^2}{2}(\beta^2K^4\eta^2+K^2G^2+K^2\sigma^2)\\ &+\frac{\eta}{2}(1+K^2)\beta^2\eta^2K^2\end{aligned} \tag{25}$$

Theorem 4: In Non-IID settings, under Lemmas 1 and 2, given the number of local updates K, after T iterations, the expected average-squared gradients of $F(\omega)$ is upper bounded as:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\omega(T))\|^2 \leq &\frac{4[F(\omega_1)-F(\omega^*)]}{\eta T}\\ &+8\beta\eta K^2(G^2+\sigma^2+1)\end{aligned} \tag{26}$$

when the learning rate satisfies $\eta \leq \min\{\frac{1}{2\beta K^2B^2}, \frac{\sqrt{G^2+\sigma^2}}{\beta K}, \frac{2}{(1+K^2)\beta}\}$

Proof: We set the coefficient of the secondary item $\mathbb{E}\|\nabla F(\omega(T)\|^2$ to $-\frac{\eta}{4}$. The inequality of (26) still holds when η satisfying the following equations

$$\left(\frac{\beta\eta^2}{2}K^2B^2-\frac{\eta}{2}\right) \leq -\frac{\eta}{4} \tag{27}$$

Solving the above inequality yields the following range of η:

$$\eta \leq \frac{1}{2\beta K^2B^2} \tag{28}$$

Under the setting of the learning rate η according to (28), the inequality (25) can be transformed into:

$$\begin{aligned}\mathbb{E}F(\omega(t+1)) - F(\omega(t)) \leq &-\frac{\eta}{4}\mathbb{E}\|\nabla F(\omega(t))\|^2\\ &+\frac{\beta\eta^2}{2}(\beta^2K^4\eta^2+K^2G^2+K^2\sigma^2)\\ &+\frac{\eta^3}{2}(1+K^2)\beta^2K^2\end{aligned} \tag{29}$$

In order to establish the convergence rate of Spread+, we impose an additional constraint on the value of η:

$$\beta^2K^4\eta^2 \leq K^2G^2 + K^2\sigma^2 \tag{30}$$

Then we get another range of η:

$$\eta \leq \frac{\sqrt{G^2+\sigma^2}}{\beta K} \tag{31}$$

Equation (29) can be simplified as:

$$\begin{aligned}\mathbb{E}F(\omega(t+1)) - F(\omega(t)) \leq &-\frac{\eta}{4}\mathbb{E}\|\nabla F(\omega(t))\|^2\\ &+\beta\eta^2K^2(G^2+\sigma^2)\\ &+\frac{\eta^3}{2}(1+K^2)\beta^2K^2\end{aligned} \tag{32}$$

In order to achieve a better convergence rate, it is necessary to eliminate the term of η^3 by introducing a constrain on η, i.e.,

$$\frac{\eta^3}{2}(1+K^2)\beta^2K^2 \leq \beta\eta^2K^2 \tag{33}$$

which indicates that η should satisfy the following condition:

$$\eta \leq \frac{2}{(1+K^2)\beta} \tag{34}$$

Now, according to (32) and (34), we have:

$$\begin{aligned}\mathbb{E}F(\omega(t+1)) - F(\omega(t)) \leq &-\frac{\eta}{4}\mathbb{E}\|\nabla F(\omega(t))\|^2\\ &+2\beta\eta^2K^2(G^2+\sigma^2+1)\end{aligned} \tag{35}$$

Having rearranged the term $\mathbb{E}\|\nabla F(\omega(T)\|^2$ on the left-hand side of the inequality and dividing both sides of (35) by $\frac{\eta}{4}$, we can add the both sides over iteration t=1 to T. Then, the average value of $\mathbb{E}\|\nabla F(\omega(T)\|^2$ can be bounded, which directly reveals the convergence rate of the training, i.e.,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\omega(T))\|^2 \leq &\frac{4[F(\omega_1)-F(\omega^*)]}{\eta T}\\ &+8\beta\eta K^2(G^2+\sigma^2+1)\end{aligned} \tag{36}$$

The proof is completed.

Corollary 1: For sufficiently large T, with a fixed $\eta = \sqrt{\frac{F(\omega_1)-F(\omega^*)}{2T\beta K^2(G^2+\sigma^2+1)}}$, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\omega(T))\|^2 \preceq O\left(\frac{1}{\sqrt{T}}\right) \tag{37}$$

where $\preceq$ denotes order inequality, i.e., less than or equal to up to a constant factor.

Proof: According to (26), we define a new objective $f(\eta) = \frac{4[F(\omega(T))-F(\omega^*)]}{\eta T} + 8\beta\eta K^2(G^2+\sigma^2+1)$. Let $f'(\eta) = 0$. We can get:

$$\eta = \sqrt{\frac{F(\omega_1)-F(\omega^*)}{2T\beta K^2(G^2+\sigma^2+1)}} \tag{38}$$

Considering all the above ranges of η in (28)(31)(34), we get final range of η is:

$$\eta \leq \min\left\{\frac{1}{2\beta K^2 B^2}, \frac{\sqrt{G^2+\sigma^2}}{\beta K}, \frac{2}{(1+K^2)\beta}\right\} \tag{39}$$

When T satisfies the following range:

$$T \geq \max\left\{\frac{2[F(\omega_1)-F(\omega^*)]\beta K^2 B^4}{G^2+\sigma^2+1}, \frac{[F(\omega_1)-F(\omega^*)]\beta}{2(G^2+\sigma^2+1)(G^2+\sigma^2)}, \frac{[F(\omega_1)-F(\omega^*)]\beta(1+K^2)}{8K^2(G^2+\sigma^2+1)}\right\} \tag{40}$$

We can get the upper bound of $f(\eta)$:

$$f(\eta) \geq \sqrt{\frac{32[F(\omega_1)-F(\omega^*)]\beta K^2(G^2+\sigma^2+1)}{T}} \tag{41}$$

We get the result of the convergence:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\omega(T))\|^2 \preceq O(\frac{1}{\sqrt{T}}) : \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\omega(T))\|^2 \leq \sqrt{\frac{32[F(\omega_1)-F(\omega^*)]\beta K^2(G^2+\sigma^2+1)}{T}} \tag{42}$$

The convergence analysis is topology independent for both convex and non-convex assumptions, due to the fact that updates from all workers are always collected when the global model is aggregated.

B. Privacy Analysis

As with traditional centralized federated learning, hierarchically trained workers only share parameter/gradient updates, and the data they occupy does not leave the local area. In centralized federated learning, the model update uploaded by worker i to the central server j is defined as $\Delta_{(i,j)}$, then the central server contains the information of the client is defined as $\bigcup_{i=1}^{N}\Delta_{(i,j)}$.

In the hierarchical architecture, for the sake of discussion, we consider the centralized central server j as the cluster head, which performs the model aggregation operation and then uploads the model update $\Delta_{(j,J)}$ to the server J. As described in Section III-B2, the aggregation operation can be regarded as a fast averaging process, in which case $\Delta_{(j,J)}$ does not disclose to J any attributes about the worker, instead this means that

$$\Delta_{(j,J)} \subset \bigcup_{i=1}^{N}\Delta_{(i,j)} \tag{43}$$

Therefore, the hierarchical architecture does not reduce the privacy-preserving nature of federated learning. However, the same shortcoming as conventional federated learning, it is difficult to cope with model inversion attacks [22], [23], i.e., there may be attackers who use model updates to steal privacy, and future research will be conducted in this area as well.

V. Implementation

We implement a Spread+ prototype based on the current FL system and TensorFlow Federated (TFF) (see Section II-C). We implement the Spread+ server on the Amazon cloud, and the cluster head and worker on the Raspberry Pi 4 Model B.

The Spread+ server: Upon receiving a training request, the Spread+ server utilizes the Selector module within the FL system to gather worker-specific details (like bandwidth and location). This information is then used to execute the cluster construction algorithm. Additionally, the Selector is adapted to manage network connections between the server and the cluster heads.

The Spread+ server's server aggregator (SA) module is an extension of the FL server's Master Aggregator, tasked with conducting inter-cluster aggregation. A unique feature of the SA, compared to the Master Aggregator, is its ability to accumulate training state data, encompassing edge loss, cluster loss, and global loss for each aggregation.

Executing the `report_local_outputs` method in TFF, this functionality is realized. While the FL training parameters are predetermined and remain constant throughout the training process, Spread+ dynamically computes these parameters during runtime. Spread+ includes additional training parameters, such as the intra/inter-cluster aggregation rate, which are not found in the standard FL system. These new parameters are incorporated into the `tf.variable` data structure in TFF, which stores the training parameters. The parameter controller in Spread+ adjusts the training parameters by altering the values in `tf.variable` based on the results from the adaptive algorithm. This approach facilitates the modification of training parameters in real-time.

The cluster head: The Aggregator module offers an interface that can accept a list of workers and training parameters from the master aggregator. We utilize this interface, inputting the cluster list and FL plan, to effectively implement the CA. Additionally, Spread+ enhances the selector module present in the FL server, extending its capabilities to the worker level. This modification is aimed at efficiently managing the network connections between the cluster head and the workers.

The worker: In Spread+, the roles fulfilled by workers mirror those in the FL system. To this end, Spread+ directly utilizes the FL Runtime and the Example Store modules, which are integral components of the FL system's worker framework.

VI. Evaluation

A. Evaluation Setup

Platform: In this experiment, we focus on performance comparisons of different systems, ignoring real-world latency for now. The outcomes are independent of the hardware and instead rely solely on the system itself. Therefore, we use a cluster of servers equipped with 16 NVIDIA GeForce RTX 3080 GPUs and create multiple VMs in the cluster for our experiments. (The same results are obtained even if real-world hardware is used in

the experiments). Experiments considering real-world application scenarios are discussed in Section VII. In configuring the network for workers, we utilize a publicly available smartphone trace dataset [6], which includes bandwidth data for hundreds of smartphones. The bandwidth ranges from 1 Mbps to 12 Mbps.

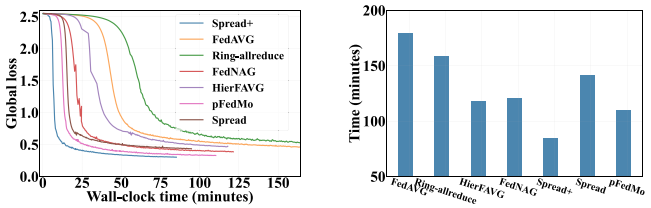
Dataset and Model: Regarding the training dataset, we chose CIFAR-10 [24], which comprises 60,000 color images, each 32x32 in size, distributed across 10 classes. It includes 50,000 images for training and 10,000 for testing. The system performance is evaluated under both IID and Non-IID data scenarios. In the IID case, images from all 10 CIFAR-10 classes are randomly distributed across workers, ensuring an IID dataset. For the Non-IID case, each worker is allocated data from a subset of classes, with $x < 10$ classes per worker to measure Non-IID distribution. A smaller x indicates a higher degree of Non-IID. We use 3-class, 6-class, and 9-class Non-IID to represent high, medium, and low Non-IID levels, respectively. In the Non-IID scenario, each worker receives images from 6 CIFAR-10 classes, resulting in a Non-IID distribution. The system's effectiveness is tested using a VGG16 [25], which comprises approximately 138 M parameters and has a size of 93 MB.

Baselines: In the analysis, we evaluated the performance of the newly proposed Spread+ architecture against various model training strategies. These include 1) a standard FL system that incorporates FedAvg [26] for its model aggregation algorithm; 2) The Ring-allreduce architecture [27], which is a decentralized system. In this setup, workers exchange updates directly using an all-reduce scheme. The participating workers are organized in a circular fashion, where each worker forwards its updated model to the next in a clockwise direction and receives updates from the preceding one. 3) FedNAG [28], a FL system with nesterov accelerated gradient. 4) HierFAVG [7], a three-tier system that employs FedAvg as the model aggregation algorithm. 5)pFedMo [29], a personalized Federated Learning with contrastive momentum algorithm. 6) Spread [1], previous work, lacks a directed cluster organization algorithm and optimization of the model aggregation algorithm.

Training and Control Parameters: In evaluations, we set the batch size $B = 16$ and the learning rate $\eta = 0.01$ (which we find works well through evaluations). Unless otherwise specified, we set the number of workers as 64, and the employed data distribution is 6-class Non-IID.

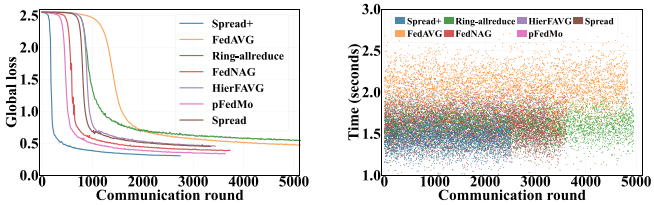
B. Overall Performance

1) Training Job Completion Time: Our initial investigation focuses on the enhancement Spread+ brings to the completion time of training jobs. We tracked the progression of global loss for various methods, including Spread+, FedAVG, Ring-allreduce, FedNAG, HierFAVG, pFedMo and Spread, throughout the training, as illustrated in Fig. 5(a). It's important to note that the model is considered converged, signifying the completion of the training job, once the global loss stabilizes. Observations show that Spread+ has significantly faster global loss reduction compared to the other methods, followed by Spread, pFedMo and HierFAVG, suggesting that the hierarchical framework has the ability for fast convergence, while



(a) Global loss as a function of time. (b) Training job completion time.

Fig. 5. Convergence speed versus the wall-clock time.



(a) Global loss as a function of communication round. (b) Time cost for each communication round.

Fig. 6. Convergence speed versus communication round.

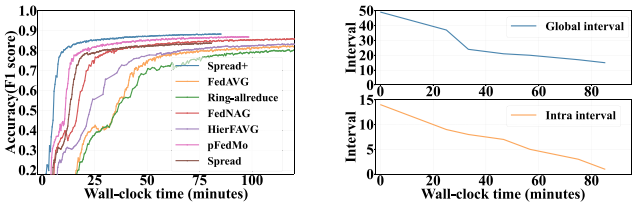
TABLE I
THE NEEDED COMMUNICATION ROUND

frameworks	Spread+	FedAVG	Ring-allreduce	FedNAG	HierFAVG	Spread	pFedMo
Round	2761	5066	5113	3740	3448	3355	3614

pFedMo is indeed effective in utilizing personalized momentum acceleration to process Non-IID data and thus accelerate convergence. Fig. 5(b) depicts the completion times of the training job. Spread+ outperforms other methods, completing training in 85.14 minutes, a 49.58% improvement over FedAVG's 168.85 minutes, and 22.78% faster than pFedMo's 110.25 minutes. While pFedMo benefits from momentum acceleration, its aggregation end faces congestion, leading to communication inefficiencies that slow overall training. In contrast, Spread+'s hierarchical structure resolves aggregation congestion and mitigates the negative effects of Non-IID data, achieving the fastest training completion time.

Fig. 6 shows the convergence state in terms of the communication round, providing more details. A faster convergence speed in terms of communication rounds means a better training quality for each iteration. In Fig. 6(a), considering the convergence speed in terms of communication round, HierFAVG, FedAVG, and FedNAG have a fast speed, after it Spread+, and Ring-allreduce is the slowest. Table I shows the number of communication rounds required for the convergence of the five systems, highlighting that systems with a ring structure need more rounds for model synchronization, indicating poorer training quality per iteration. Fig. 6(b) shows that most communication rounds in Spread+ take about 1.4 minutes, whereas FedAVG takes 2 to 2.5 minutes due to communication congestion. While HierFAVG and FedNAG with momentum acceleration are effective, there is still room for optimization. Spread+ efficiently balances workload distribution in model aggregation with synchronization frequency.

2) Training Accuracy: We shift focus to examine how Spread+ enhances training accuracy, a crucial indicator of an ML



(a) Accuracy. (b) Aggregation intervals.

Fig. 7. The accuracy of the training model.

system's training quality. The F1 score is utilized for assessing training accuracy. The complete training process over time is depicted in Fig. 7(a). Due to limitations in model aggregation algorithms and slower synchronization rates, both FedAVG and Ring-allreduce exhibit prolonged training durations without significantly improving accuracy. Relative to FedAVG, the hierarchical structure of HierFAVG improves the accuracy by about 1.8% and reduces the training time by 44%. Similarly, pFedMo improves accuracy by about 4% and reduces training time by 30.1% through personalized momentum acceleration optimization. spread+ does not have a significant accuracy advantage over pFedMo, but achieves and maintains maximal accuracy faster. Fig. 7(b) illustrates the global and intra-cluster aggregation intervals over time. Notably, both intervals shrink to a minimum as the model training ends. This confirms the conclusion that frequent aggregation contributes to lower function loss and enhanced training quality.

3) System Scalability: Now we investigate the scalability of the system with Spread+. Fig. 8(a) plots the overall training time with different systems under different numbers of workers. We note that for FedAVG, the model training time decreases with the number of workers when the number of workers is less than 32, and the training time starts to increase significantly when the number of workers increases. pFedMo exhibits strong performance even with 128 workers, though convergence time increases beyond this point. Nonetheless, its improvements remain significant compared to FedAVG and Ring-allreduce. In contrast, hierarchical architectures like HierFAVG and Spread+ excel with larger worker counts but are less efficient with fewer workers due to the overhead of constructing the hierarchy. However, at 256 workers, Spread+ achieves a training time of 69.65 minutes, a 3.34× speedup over FedAVG, which takes 232.64 minutes.

As shown in Fig. 8(b), Spread+ outperforms other systems in terms of scalability. It presents a detailed comparison of the computational and communication times of each framework. With an increase in the number of workers, the computational time in FL systems decreases, as more workers participate in training the model. In contrast, the communication time increases, particularly for non-hierarchical frameworks like FedAVG and pFedMo, due to severe communication congestion as the number of workers grows.

Fig. 8(c) and (d) show convergence speed and accuracy of each framework. The convergence speed is defined as the time when the loss reduction rate significantly slows down, with a shorter time indicating faster convergence. Spread+ shows suboptimal performance with fewer workers but outperforms other hierarchical frameworks when the worker count exceeds 64, as they are hindered by Non-IID data, leading to slower convergence and lower accuracy. Momentum-based frameworks like pFedMo and FedNAG suffer from communication congestion and perform poorly. In contrast, Spread+ effectively mitigates both communication congestion and Non-IID issues, ensuring superior scalability.

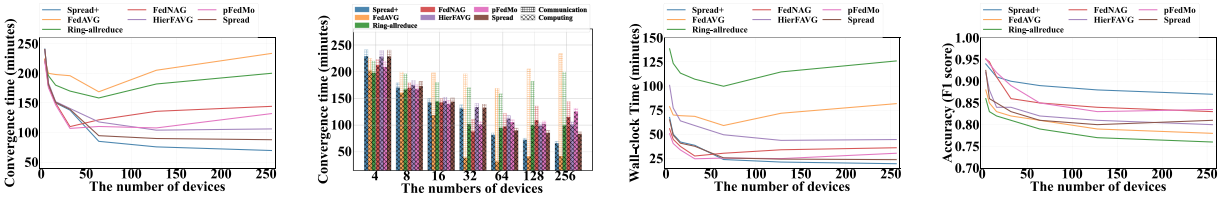
C. Sensitivity Analysis

1) Adaptability to Heterogeneity: We now evaluate the adaptability of Spread+ to different levels of worker heterogeneity. We define the degree of heterogeneity among workers as follows: $H = \frac{\sum_i^N v_i}{N \min\{v_1, v_2, ..., v_N\}}$, where v_i are the iterations that worker n_i can process per unit of time.

Fig. 9 illustrates the variation in convergence time across three policies, considering different degrees of worker heterogeneity. As depicted in Fig. 9, with an increase in the degree of heterogeneity from 1 to 3.2, the convergence time for FedAVG increases by 11.72%, 3.08 times greater than Ring-allreduce (3.8%) and 2.74 times higher than Spread+ (4.28%). FedNAG and HierFAVG see increases of 7.7% and 5.2%, respectively. FedAVG and FedNAG are significantly affected by high heterogeneity, as their protocols require all workers to wait for the slowest one during aggregation, making convergence time dependent on the slowest participant. In contrast, pFedMo mitigates this with momentum acceleration, achieving relatively good performance despite slow workers. Spread+, however, isolates the impact of slow workers to intra-cluster aggregation, allowing it to effectively handle heterogeneity and maintain the shortest convergence time.

2) Robustness Against Data Distribution: We focus on the impact of data distribution. Fig. 10 presents the test precision in relation to the number of communication rounds in both the IID and the non-IID data distribution scenarios. We observe that Spread+ achieves high training accuracy of up to 91% under both IID and low Non-IID conditions. Under IID data conditions, Spread+ converges after approximately 900 communication rounds, which represents a 64.8% reduction compared to high IID scenarios. This indicates that when applied to IID data, Spread+ significantly optimizes communication congestion.

3) Effectiveness of Cluster Construction: We assess the efficiency of cluster construction in Spread+, specifically analyzing how it compares to strategies employing a fixed size for clusters. Fig. 11 illustrates the breakdown of communication and computation times for various cluster construction approaches. Spread+ surpasses fixed-size cluster strategies with a performance improvement ranging from 2.53 to 3.04 times. This advantage is attributed to the fact that smaller clusters lead to a higher number of clusters, causing congestion at the server during inter-cluster aggregation. Conversely, larger clusters tend to create congestion at the cluster head during intra-cluster aggregation. Spread+ adeptly manages the communication capabilities of workers and strikes a balance between cluster size



(a) Model training time. (b) Time composition. (c) Model convergence speed. (d) Accuracy.

Fig. 8. The scalability of different ML systems.

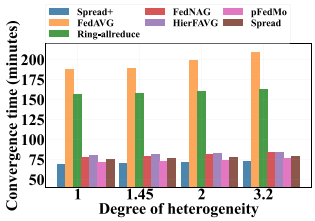


Fig. 9. Heterogeneity.

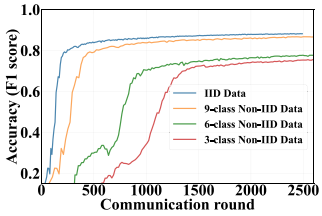


Fig. 10. Data distribution.

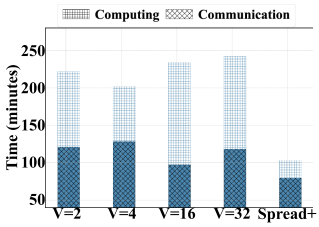


Fig. 11. Cluster strategy.

and the total number of clusters, thus significantly boosting performance.

4) Effectiveness of Adaptive Aggregation Interval: We scrutinize the efficacy of Spread+'s adaptive aggregation interval strategy. Spread+ is compared with systems that use the static intra-cluster aggregation interval k_1 and the inter-cluster aggregation interval k_2. Spread + demonstrates significantly higher accuracy compared to systems with larger global intervals and substantially reduces training time compared to those with smaller global intervals in Table II. These findings confirm the effectiveness of the adaptive aggregation interval strategy in Spread+.

VII. Case Studies

In an FL setting, workers may not contribute resources as consistently as distributed servers in cloud environments. Spread+ currently does not account for this variability. We aim to develop mechanisms to adapt cluster construction and training parameters in real time based on worker engagement. To address potential cluster head failures, we plan to implement a reselection strategy. Additionally, Spread+ servers face a single point of failure risk, which can be efficiently mitigated using solutions like Zookeeper [30].

TABLE II
Aggregation Interval

k_1, k_2	Time	Accuracy
5, 2	2.11	89.5%
2, 5	2.07	89.1%
50, 4	1.24	81.7%
20, 10	1.22	82.9%
4, 50	1.17	83.2%
Spread+	1.27	90.1%

A. Smart Object Detection for Self-Driving Cars

We present a case study using Spread+ for a smart object detection application in autonomous cars. Following the setup in [31], we use the BelgiumTSC [32] dataset, which includes 7,000+ images and 11,219 bounding boxes for over 2,000 traffic signs. The system simulates 12 Raspberry Pi devices, each generating 5 threads to emulate 60 vehicles, each assigned Non-IID preprocessed data. The A100 blade server acts as the cloud server, and the cluster comprises three GeForce RTX3080 GPUs with 10 GB memory each.

The system is deployed in a simulated vehicle network to train the object detection model. During training, clusters are formed based on data distribution similarity, followed by intra-cluster aggregation, and after several inter-cluster aggregation rounds, the model is sent to the server for global aggregation. As shown in Fig. 12, the system converges to 94.32% accuracy in 10.8 minutes with 10 ms E2E latency, and 88.9% accuracy in 16.54 minutes with 15 ms latency. While existing networks fail to meet real-time requirements for self-driving cars, 6 G could address this [31]. Despite this, our system shows rapid training and stable convergence. Post-training, randomly selected cars are tested with traffic signs, and performance is evaluated using precision, recall, and F1 score. The results in Table III demonstrate the system's robustness under varying latency conditions. For a delay of 10 ms, the F1 score is 0.928, indicating high detection accuracy. Even with a delay of 15 ms, the system maintains reliable performance with an F1 score of 0.871. This highlights

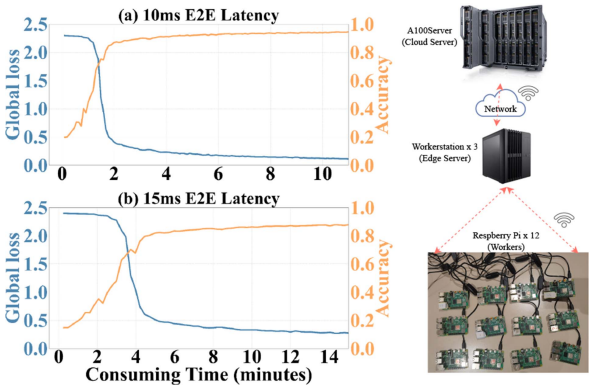


Fig. 12. End-to-end operations of Spread+ in the smart object detection system.

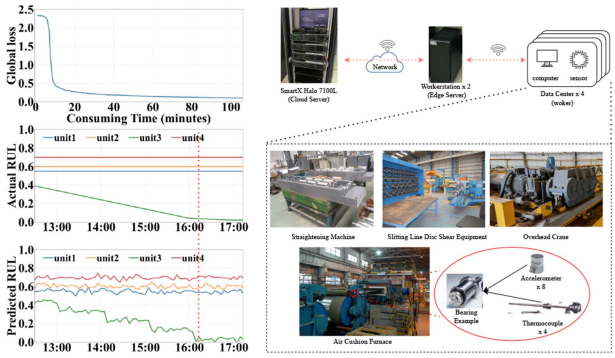


Fig. 13. End-to-end operations of the Spread+ system in an industrial equipment fault detection scenario.

TABLE III
RECOGNITION PERFORMANCE

Delay \ Indicator	Precision	Recall	F1 score
10ms	0.914	0.943	0.928
15ms	0.891	0.852	0.871

the adaptability of the system to real-time 6 G scenarios where low-latency and high-accuracy detection are essential.

B. Industrial Equipment Fault Detection Application

We deployed Spread+ in a fault prediction system at an aluminum plant, monitoring bearings from four industrial machines: Straightening Machine, Slitting Line Disc Shear, Overhead Crane, and Air Cushion Furnace. The system setup includes an A100 blade server as the cloud, two GeForce RTX3080 GPUs with 10 GB memory as the cluster head, and the data centers of the four machines as workers. Following the data collection setup in [33], sensors on each bearing collect vibration signals and temperature data. Vibration is measured by eight PCB 353B33 accelerometers (vertical and horizontal), while four thermocouples on the bearing's outer ring measure temperature. The data from the four workers is Non-IID distributed, making model accuracy in FL challenging.

Our system was tested over a two-month period to gather sufficient sensor data. The data was preprocessed by data centers, and a temporal convolutional network (TCN) was used for model training. Spread+ operates with a cluster comprising four workers and two cluster heads. Using adaptive model aggregation intervals, the cluster heads transmit the aggregated model to the server for global synchronization. The system predicts the remaining useful life (RUL) of bearings in real-time, supported by periodic data cleaning (every 5–10 minutes) for model tuning. This efficiency is enabled by the hierarchical architecture. As illustrated in Fig. 13, the prediction accuracy is high. After 1.5 months of operation, the RUL of a unit bearing decreased by 5% around 16:00, and the system successfully issued an alert 10 minutes in advance. Spread+ consistently maintains training accuracy, even with non-IID data.

VIII. RELATED WORK AND LIMITATION

To minimize model training time, there are a set of trade-offs for various reality constraints.

To reduce model training workload: Compressing ML models to remove non-critical parameters has been extensively studied. Minimizing FL communication overhead is a major research topic. Transmitting compressed model parameters reduces update bits. Quantization and sparsification are common compression methods. Quantized algorithms reduce model updates to fewer bits. [34] modified FedAvg to compress uploaded parameters using distributed Adam optimization. Since sparse algorithms eliminate most elements, communication traffic is greatly reduced. Tang et al. [35] developed a sparse algorithm for client-peer interaction and a gossip matrix generation method to optimize bandwidth usage. Li et al. [36] enhanced Top-k compression with control algorithms, resulting in better compression ratio and more balanced computing and communication efforts. Additionally, Su et al. [37] improved communication efficiency and addressed unreliable FL transmissions by combining model compression, forward error correction, and retransmission. These model compression technologies can help Spread+ accelerate ML model training.

To mitigate the communication overhead: Some studies have pointed out that a centralized PS can become a bottleneck, as model updates lead to intense communication congestion. A prominent solution is adopting a ring structure to alleviate this congestion [38], with allreduce as its practical implementation [39]. Additional approaches include communication scheduling, model compression, and balancing trade-offs between accuracy and communication [40]. Spread+ addresses communication overhead effectively through the deployment of a tiered architectural design.

To reduce data heterogeneity: Personalized FL [41] shows its advantages in solving the problem of poor convergence of heterogeneous data. Among these studies, there are on how to construct personalized clusters in order to enhance the performance of collaborative training between clients. FedAMP [42] and FedFomo [43] utilize loss similarity and parameter similarity, respectively, to encourage pairwise collaboration between similar clients, but their communication efficiency is not as efficient as it should be due to the limitation

of one-to-one pairing. [44] proposes a learning-to-collaborate framework and aims at identifying the non-overlapping collaboration coalitions via a Pareto optimization approach. [45] allows dynamic coalition formation between organizations and constructs a simple distributed merge and split coalition formation algorithm.

To optimize FL in a hierarchical framework: Yoga [46] was designed as an efficient hierarchical framework, using the maximum matching algorithm to generate appropriate layer-by-layer model aggregation strategies for the client. Hiflash [47] utilizes reinforcement learning and integrates algorithms such as adaptive staleness control and heterogeneous-awareness, which improve model accuracy and reduce communication overhead. A recent work [48], on the other hand, alleviates overall computational and communication burdens by partitioning the global model into disjoint submodels. Spread+, however, addresses the issue starting from the cluster construction strategy and synchronization mechanism. Using a game-theoretic approach for cluster formation, Spread+ mitigates the impact of Non-IID data. It further improves communication efficiency by employing intra-cluster synchronization and inter-cluster asynchronous communication.

Limitation: Considering that many existing personalized federated learning approaches incorporate momentum acceleration and similar techniques, the aggregation algorithm proposed in this paper still has potential for further optimization. For instance, iterative algorithms with momentum acceleration could be integrated to enhance model convergence speed. Additionally, as discussed in Section IV-A, Spread+ currently lacks defense mechanisms against model inversion attacks. Addressing this limitation represents a promising direction for future work.

IX. Conclusion

Our study analyzed scalability in ML systems, revealing delays from model aggregation congestion. To address this, we developed Spread+, a tiered architecture that decentralizes aggregation to edge devices. It includes a novel tiered structuring algorithm, runtime modules, and an adaptive training parameter algorithm. Performance is enhanced by directed cluster organization and an efficient aggregation algorithm. A functional system based on the FL framework demonstrates its practical efficacy.

References

[1] C. Hu, H. H. Liang, X. M. Han, B. A. Liu, D. Z. Cheng, and D. Wang, "Spread: Decentralized model aggregation for scalable federated learning," in *Proc. 51st Int. Conf. Parallel Process.*, 2022, pp. 1–12.

[2] G. Nain, K. Pattanaik, and G. Sharma, "Towards edge computing in intelligent manufacturing: Past, present and future," *J. Manuf. Syst.*, vol. 62, pp. 588–611, 2022.

[3] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surv. Tuts.*, vol. 24, no. 1, pp. 1–30, 1st Quarter 2022.

[4] A. Hashemi, A. Acharya, R. Das, H. Vikalo, S. Sanghavi, and I. Dhillon, "On the benefits of multiple gossip steps in communication-constrained decentralized federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2727–2739, Nov. 2022.

[5] Z. Chen, L. Shi, X. Liu, J. Li, S. Liu, and Y. Xu, "OSP: Boosting distributed model training with 2-stage synchronization," in *Proc. 52nd Int. Conf. Parallel Process.*, New York, NY, USA, 2023, pp. 102–111.

[6] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.*, Kongens Lyngby, Denmark, 2019, pp. 74–90.

[7] Z. Yang, S. Fu, W. Bao, D. Yuan, and B. Zhou, "Hierarchical federated learning with adaptive momentum in multi-tier networks," in *Proc. IEEE 43rd Int. Conf. Distrib. Comput. Syst.*, 2023, pp. 499–510.

[8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," 2020, *arXiv:1907.02189*.

[9] J. Lu, Y. Chen, S. Cao, L. Chen, W. Wang, and Y. Xin, "LEAP: Optimization hierarchical federated learning on non-IID data with coalition formation game," 2024, *arXiv:2405.00579*.

[10] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated HD map updating through overlapping coalition formation game," *IEEE Trans. Mobile Comput.*, vol. 23, no. 2, pp. 1641–1654, Feb. 2024.

[11] K. Bonawitz et al., "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.

[12] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 6950–6960.

[13] Y. Xu, Z. Jiang, H. Xu, Z. Wang, C. Qian, and C. Qiao, "Federated learning with client selection and gradient compression in heterogeneous edge systems," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5446–5461, May 2024.

[14] T. Wang, Y. Liu, X. Zheng, H.-N. Dai, W. Jia, and M. Xie, "Edge-based communication optimization for distributed federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2015–2024, Jul./Aug. 2022.

[15] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 7611–7623.

[16] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[17] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[18] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," 2018, *arXiv:1810.08313*.

[19] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-IID data," 2019, *arXiv:1905.06641*.

[20] S. Wang et al., "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, 2018, pp. 63–71.

[21] Z. Qu, N. Jia, B. Ye, S. Hu, and S. Guo, "FedQClip: Accelerating federated learning via quantized clipped SGD," *IEEE Trans. Comput.*, vol. 74, no. 2, pp. 717–730, Feb. 2025.

[22] C. Hu et al., "PriFairFed: A local differentially private federated learning algorithm for client-level fairness," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2024.3516813.

[23] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated anomaly analytics for local model poisoning attack," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 596–610, Feb. 2022.

[24] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*.

[26] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[27] A. Gibiansky, "Bringing HPC techniques to deep learning," Baidu Research, 2017.

[28] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya, "Federated learning with Nesterov accelerated gradient," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4863–4873, Dec. 2022.

[29] S. Fu, Z. Yang, C. Hu, and W. Bao, "Personalized federated learning with contrastive momentum," *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2024.3403387.

[30] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for internet-scale systems," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, Boston, MA, USA, 2010, Art. no. 11.

[31] X. Zhou, W. Liang, J. She, Z. Yan, I. Kevin, and K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6G supported internet of vehicles," *IEEE Trans. Veh. Technol*, vol. 70, no. 6, pp. 5308–5317, Jun. 2021.

[32] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3D localisation," in *Proc. Workshop Appl. Comput. Vis.*, 2009, pp. 1–8.
[33] H. Qiu, J. Lee, J. Lin, and G. Yu, "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *J. Sound Vib.*, vol. 289, pp. 1066–1090, 2006.
[34] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.
[35] Z. Tang, S. Shi, B. Li, and X. Chu, "GossipFL: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 3, pp. 909–922, Mar. 2023.
[36] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[37] X. Su, Y. Zhou, L. Cui, and J. Liu, "On model transmission strategies in federated learning with lossy communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1173–1185, Apr. 2023.
[38] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comput.*, vol. 69, no. 2, pp. 117–124, 2009.
[39] T. T. Nguyen, M. Wahib, and R. Takano, "Hierarchical distributed-memory multi-leader MPI-allreduce for deep learning workloads," in *Proc. 6th Int. Symp. Comput. Netw. Workshops*, Takayama, 2018, pp. 216–222.
[40] S. Nikolaou, C. Anagnostopoulos, and D. Pezaros, "Communication-aware edge-centric knowledge dissemination in edge computing environments," in *Real-Time Data Analytics for Large Scale Sensor Data*. New York, NY, USA: Academic, 2019, Art. no. 139.
[41] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023.
[42] Y. Huang et al., "Personalized cross-silo federated learning on non-IID data," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 7865–7873.
[43] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, "Personalized federated learning with first order model optimization," 2020, *arXiv:2012.08565*.
[44] S. Cui, J. Liang, W. Pan, K. Chen, C. Zhang, and F. Wang, "Collaboration equilibrium in federated learning," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, New York, NY, USA, 2022, pp. 241–251.
[45] S. Jiang and J. Wu, "Coalition formation game in the cross-silo federated learning system," in *Proc. IEEE 19th Int. Conf. Mobile Ad Hoc Smart Syst.*, 2022, pp. 49–57.
[46] J. Liu, J. Liu, H. Xu, Y. Liao, Z. Wang, and Q. Ma, "YOGA: Adaptive layer-wise model aggregation for decentralized federated learning," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1768–1780, Apr. 2024.
[47] Q. Wu et al., "HiFlash: Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 5, pp. 1560–1579, May 2023.
[48] W. Fang, D.-J. Han, and C. G. Brinton, "Submodel partitioning in hierarchical federated learning: Algorithm design and convergence analysis," in *Proc. IEEE Int. Conf. Commun.*, 2024, pp. 268–273.

Huanghuang Liang received the BS degree in automation engineering from the Anhui University of Technology, in 2016, and the MS degree in automation engineering from the University of Electronic Science and Technology of China, in 2019. He is currently working toward the PhD degree in computer science with Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.

Xin Yang received the BS degree from Wuhan University. He is currently working toward the MS degree in computer science with Wuhan University. His research interests include edge learning, federated learning, and distributed computing.

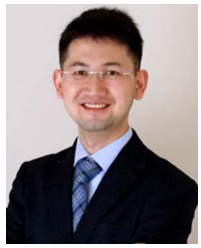
Xiaoming Han received the BS degree in software engineering from Inner Mongolia University, in 2013, and the MS degree in computer technology from Xiamen University, in 2019. He is currently working toward the PhD degree with Wuhan University. His research interests include distributed/parallel computing, federated learning, and AI systems.

Boan Liu received the BS degree from the South China University of Technology, in 2020, and the MS degree from Wuhan University, Wuhan, China, in 2023. He is currently working toward the PhD degree in computer engineering with the Hong Kong Polytechnic University, Hong Kong, China. His research interests include system architecture and distributed deep learning.

Chuang Hu received the BS and MS degrees from Wuhan University, in 2013 and 2016, respectively, and the PhD degree from the Hong Kong Polytechnic University, in 2019. His research interests include edge learning, federated learning/analytics, and distributed computing.

Dan Wang (Senior Member, IEEE) received the BS degree in computer science from Peking University, in 2000, the MS degree in computer science from Case Western Reserve University, in 2004, and the PhD degree in computer science from Simon Fraser University, in 2007. He is currently a professor with the Department of Computing, Hong Kong Polytechnic University. His research interests lie in networked systems and in the inter-discipline domains of smart energy systems.

Xiaobo Zhou (Senior Member, IEEE) received the BS, MS, and PhD degrees in computer science from Nanjing University, in 1994, 1997, and 2000, respectively. Currently, he is a distinguished professor of IOTSC and the Department of Computer and Information Science, University of Macau, Macau SAR. His research focuses broadly on distributed systems and cloud computing. He served as chair of the IEEE Technical Community in Distributed Processing for 2020–2023.



Dazhao Cheng (Senior Member, IEEE) received the BS degree in electrical engineering from the Hefei University of Technology, in 2006, the MS degrees in electrical engineering from the University of Science and Technology of China, in 2009, and the PhD degree from the University of Colorado at Colorado Springs, in 2016. He is currently a professor with the School of Computer Science, Wuhan University. His research interests include Big Data and cloud computing.