PAHInA: Precision-Aware Hierarchical In-Network Aggregation for Edge Distributed Training

Yingpu Nian, *Graduate Student Member, IEEE*, Bo Yi, *Member, IEEE*, Qiang He, Xingwei Wang, Geyong Min, *Member, IEEE*, Keqin Li, *Fellow, IEEE*, and Sajal K. Das, *Fellow, IEEE*

***Abstract*—The rise of edge intelligence is driving distributed machine learning toward a new paradigm of edge-collaborative computing. To overcome the severe communication bottleneck in this paradigm, In-Network Aggregation is a critical enabling technology. However, its effectiveness is fundamentally undermined by the profound resource heterogeneity of edge networks. Specifically, edge devices, adapting to hardware constraints, operate at varying numerical precisions, leading to significant data inflation as gradients are aggregated. Compounding this, unevenly distributed network resources and traditional, precision-oblivious routing strategies often misallocate critical, high-precision gradients to low-quality paths. This mismatch creates severe network congestion, crippling the efficiency of distributed training. To address this, we propose the Precision-Aware Hierarchical In-Network Aggregation (PAHInA) framework, the first, to our knowledge, to perform routing optimization for in-network aggregation that explicitly considers precision heterogeneity. The core of PAHInA is an intelligent control-plane scheduler that co-optimizes for gradient priority and path cost, dynamically planning the most cost-effective aggregation strategy for each flow. This fine-grained scheduling guarantees that high-priority gradients are routed through premium, low-latency paths, minimizing global communication overhead. On the data plane, we leverage the eXpress Data Path (XDP) for high-performance packet processing to reduce aggregation-induced overhead. Extensive simulations show that, compared to state-of-the-art baselines, PAHInA significantly mitigates network congestion, reducing end-to-end communication time by up to 33% and boosting overall training throughput by approximately 30%.**

Received 14 August 2025; revised 2 November 2025; accepted 29 January 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor H. Shen. Date of publication 3 February 2026; date of current version 19 February 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62472078 and Grant U22A2004; in part by the Fundamental Research Funds for the Central Universities of China under Grant N25LPY013; in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy under Grant GML-KF-24-31; in part by the State Grid Corporation of China Science and Technology Project Funding under Grant 2024YF-95; and in part by the U.S. NSF under Grant OAC-2104078 and Grant ECCS-2319995. *(Corresponding authors: Bo Yi; Xingwei Wang.)*

Yingpu Nian, Qiang He, and Xingwei Wang are with the College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China (e-mail: nianpupu0124@gmail.com; heqiangcai@gmail.com; wangxw@mail.neu.edu.cn).

Bo Yi is with the College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China, and also with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen 518000, China (e-mail: yibobooscar@gmail.com).

Geyong Min is with the Department of Computer Science, University of Exeter, EX4 4QF Exeter, U.K. (e-mail: g.min@exeter.ac.uk).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Sajal K. Das is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 USA (e-mail: sdas@mst.edu).

Digital Object Identifier 10.1109/TON.2026.3660333

***Index Terms*—Edge computing, distributed training, heterogeneous precision, in-network aggregation, XDP.**

I. INTRODUCTION

THE explosive growth in the computational power of edge devices is driving a profound paradigm shift in distributed machine learning, from centralized cloud computing to collaborative edge-based training [1], [2], [3]. This emerging paradigm enhances data privacy and reduces latency, but it also introduces a severe communication bottleneck, often more acute than in traditional data centers. To address this, a natural and promising approach is to leverage powerful techniques proven effective in data center environments. Among these, In-Network Aggregation stands out as a particularly compelling candidate solution [4], [5], [6]. Its core principle is to intercept and sum gradient data from different workers at network switches, forwarding only the final aggregated result to the PS, thereby drastically reducing the data volume that reaches the central node.

However, applying in-network aggregation to complex edge environments reveals a deeper, more paradoxical challenge rooted in the inherent dual heterogeneity of these systems. On one hand, device heterogeneity, that is, the variance in compute and memory resources across edge nodes, compels workers to adopt heterogeneous precision formats (e.g., FP32, FP16, FP8) to meet local constraints [7], [8], [9]. This measure, taken to optimize local computation, inadvertently creates gradient flows of disparate sizes and numerical characteristics, complicating network aggregation. On the other hand, network heterogeneity manifests as highly imbalanced bandwidth and complex network topologies. The coupling of these heterogeneities creates two critical issues for conventional, precision-oblivious in-network aggregation mechanisms. First is the paradox of precision inflation, to maintain numerical fidelity during aggregation, a low-precision gradient (e.g., FP8) must be up-cast when combined with a high-precision one (e.g., FP16). This process can yield an aggregated packet larger than the original low-precision input, directly contradicting in-network aggregation's primary goal of reducing network traffic. Second is the severe challenge of an aggravated straggler effect [10], [11], without path quality awareness, larger and more latency-sensitive high-precision gradients can be routed onto low-bandwidth, congested paths, creating a "gradient backlog" that stalls the entire synchronous step and severely degrades training efficiency. Prior work attempts to solve these issues with either precision scaling [8], [12], [13] or network scheduling [14], [15], [16], [17] in isolation, but

fails to see the conflict, the former is network-oblivious and may worsen congestion by up-casting on taxed links, while the latter treats all gradients as uniform, failing to leverage the unique properties of heterogeneous precision data or protect critical flows.

Therefore, a holistic solution that can co-optimize computational precision with network resources remains a critical and unaddressed research gap. To bridge this gap, this paper introduces **P**recision-**A**ware **H**ierarchical **In**-Network **A**ggregation (**PAHInA**), an innovative framework for distributed training. To our knowledge, PAHInA is the first framework to design a precision-aware routing optimization mechanism for in-network aggregation. Its core idea is an intelligent control plane that jointly evaluates the "intrinsic data value," determined by gradient precision and size, against the "path transmission cost" across the network, thereby dynamically planning a cost-minimal routing and aggregation strategy for every gradient flow. The main contributions of this paper are as follows:

- Firstly, we are the first to identify and systematically formulate the core technical challenge in edge training arising from the coupling of device and network heterogeneity, namely, the intrinsic conflict between precision inflation and the aggravated straggler effect.
- Secondly, we propose PAHInA, a novel framework featuring a decoupled control and data plane architecture. Its control plane houses the optimization intelligence, while the data plane leverages eXpress Data Path (XDP) [18] for high-performance, in-kernel execution of the aggregation strategy.
- Thirdly, we design and implement an innovative, precision-aware routing and scheduling algorithm. The algorithm jointly optimizes gradient priorities and network path costs to generate optimal transport and aggregation plans for heterogeneous gradient flows, effectively mitigating the core conflict.
- Finally, we validate the effectiveness of PAHInA through extensive simulations across various representative network topologies. Our results demonstrate that, compared to state-of-the-art baselines, PAHInA reduces end-to-end communication time by up to 33% and boosts overall training throughput by 30%.

The remainder of this paper is organized as follows. Chapter II reviews related work. Chapter III provides a detailed discussion of our motivation and the core challenges. Chapters IV and V present the system architecture and the core algorithm design of PAHInA, respectively. Chapter VI showcases the comprehensive experimental setup and evaluation results. Finally, Chapter VII concludes the paper.

II. Related Works

This chapter situates our work in the context of existing research. We review dominant architectures for edge training and in-network aggregation techniques, analyzing the limitations of current schemes to highlight the novelty of our approach.

A. Edge Distributed Training

Distributed training of deep learning models originated as a solution to the computational and data-scaling challenges in data centers. Frameworks like Horovod and PyTorch DDP enabled efficient parallelism across high-speed, homogeneous cluster networks, significantly accelerating training cycles. Recently, the paradigm has shifted from centralized data centers to the network edge, giving rise to "Distributed Training at the Edge." [19], [20] This approach adapts distributed training principles for clusters of edge devices to satisfy modern demands for low-latency services and on-device data privacy. To coordinate this collaboration, two foundational architectures are prevalent. The first is the centralized Parameter Server (PS) architecture [21], [22]. Here, one or more central servers maintain the global model state. A set of workers computes local gradients and pushes them to the servers. The servers then aggregate these gradients, update the global model, and the workers pull the updated parameters for the next iteration. The second is the decentralized All-Reduce (AR) architecture [23], [24]. In this model, all nodes are peers and no central server exists. They collectively perform global gradient aggregation by communicating directly, often using structured patterns like a ring or tree topology to ensure all nodes receive the sum of all gradients.

However, migrating these architectures from the controlled environment of a data center to the network edge induces a fundamental shift in the system's performance bottleneck. Within a data center, training is typically compute-intensive, facilitated by high-performance computing resources and high-throughput, low-latency interconnects. In contrast, the edge is characterized by devices with limited computational power and communication over heterogeneous networks that often exhibit lower bandwidth, higher latency, and less stability [25], [26]. Consequently, the time required for transmitting model updates frequently dwarfs local computation time. This transforms edge distributed training from a compute-intensive to a communication-intensive challenge. This critical communication bottleneck has spurred the development of technologies such as in-network aggregation, a technique designed to reduce data transfers by processing updates within the network itself.

B. In-Network Aggregation

In-network aggregation is a technique that significantly reduces the amount of data transferred in distributed training by executing computation operations at intermediate network nodes, thereby alleviating communication bottlenecks. In recent years, many innovative schemes have emerged in this field, proposing corresponding solution strategies for different application scenarios and technical challenges. The PANAMA framework [27] proposed a scalable in-network aggregation method that efficiently processes distributed machine learning tasks in shared clusters using dedicated hardware accelerators; its design not only improves the training speed of large jobs but also significantly reduces the completion time of short jobs and latency-sensitive traffic. In the high-performance computing domain, SHArP technology [28] drastically reduces the latency of MPI Allreduce operations by implementing a scalable

in-network data reduction protocol in the Mellanox SwitchIB-2 ASIC; its optimized hierarchy and parallel mechanisms allow it to perform excellently in large-scale parallel computing. SwitchML [5], targeting distributed deep learning scenarios, proposed a method for gradient aggregation in programmable switches, significantly reducing the communication overhead of model updates by decomposing gradients into small chunks and processing them in parallel within the switch. ATP [4] further provided a dynamic in-network aggregation service in multi-tenant environments, supporting the simultaneous operation of multiple distributed training jobs by dynamically allocating aggregation resources in top-of-rack switches and achieving fair resource allocation during resource contention. Recent works by Xia et al. have further explored this area, proposing GISA [29] as a generic service, which notably highlights the impact of the bandwidth-delay product on switch resource limits, and GAIN [30] for stateless aggregation in federated learning. Additionally, the NetAgg [31] scheme utilizes software middleboxes for data aggregation on the network path, suitable for batch processing and online service applications in data centers, significantly reducing many-to-one traffic in the network by distributing aggregation computation across multiple levels of the network. However, these works mainly focus on the efficient implementation of aggregation operations on programmable switches, ignoring the impact of gradient routing choices, and are limited by the finite on-chip memory of switches and asynchronous gradient transmission issues.

To enhance the adaptability of in-network aggregation in dynamic networks, joint optimization of routing and aggregation has gradually become a research focus. Such works integrate topology-aware mechanisms into the aggregation process, dynamically planning gradient transmission paths to minimize end-to-end delay. PARING [32] proposed a Steiner tree-based approximation algorithm by jointly optimizing distributed training task placement and routing, effectively reducing communication time and traffic. XAgg [33] utilizes XDP technology to deploy in-network aggregators on servers, fully leveraging the idle CPU and memory resources of servers to address the challenges of heterogeneous computation and bandwidth resources. ALEPH [34] was the first to apply eBPF technology to gradient aggregation, improving the efficiency of distributed training through an efficient clustering algorithm in the control plane and low-overhead data processing in the data plane. GRID [35] and work [38] proposed an in-network aggregation framework based on gradient routing, optimizing communication efficiency through a randomized rounding algorithm in the control plane and a rate synchronization mechanism in the data plane. GOAT [36], [37] and work [39], through cooperative in-network aggregation and gradient scheduling, utilizes multiple programmable switches working in concert to effectively solve the efficiency problem of asynchronous gradient aggregation [40]. However, all the aforementioned works only consider single-precision gradient aggregation and have not conducted in-depth research on the problem of coexistence and efficient aggregation of heterogeneous precision gradients in heterogeneous precision scenarios. In such scenarios, different computing nodes may produce gradient data of different precisions. How to effectively integrate heterogeneous precision gradients to fully utilize the computational resources of each node while reducing communication overhead and aggregation delay has become a critical challenge to be solved. Therefore, this paper designs PAHInA to solve the aggregation optimization problem for heterogeneous precision.

III. Motivation

This chapter uses a motivational example to concretely analyze the "dual heterogeneity" challenge outlined in the introduction. We quantitatively demonstrate how existing schemes lead to precision inflation and the aggravated straggler effect, thereby revealing the core motivation for our precision-aware, hierarchical design.

A. Motivation Example

We considered a PS-based distributed training task with 1 PS and 8 workers. It should be noted that in an edge scenario, the bandwidth of each link, the processing capacity of programmable switches, and that of edge computing devices are heterogeneous and have a degree of randomness. Therefore, we set up different link bandwidths of 3, 6, and 9 Gbps, and programmable switch processing capacities were set to either 6 or 24 abstract computational units per second. Device heterogeneity was modeled by assigning workers to use either FP16 or FP8 precision for local training; we assume the data payload of an FP16 gradient is twice that of an FP8 gradient. Given that synchronous training is bottlenecked by the slowest contribution, we define the key performance metric as the system throughput, the data rate of the straggler worker, which dictates the pace of each training step. The specific topology and configuration are detailed in Figure 1.

We first analyze the performance of a baseline approach, ATP, with results shown in Figure 1(a). In the ATP scheme, gradients are aggregated at programmable switches if sufficient computational resources are available; otherwise, the switches act merely as forwarders. This dual constraint of network bandwidth and switch capacity limits the effective throughput for the FP8 and FP16 workers. The figure highlights a critical bottleneck at switch S_2, which aggregates gradients from an FP8 worker W_5 and an FP16 worker W_6. To maintain numerical fidelity, the FP8 gradient must be up-cast to FP16 before aggregation. This precision inflation results in an FP16-sized output that saturates the downstream link L_2, thereby constraining the effective throughput for all workers participating in that aggregation subtree.

A more advanced approach, represented by the work GRID, introduced joint optimization of in-network aggregation and routing. As illustrated in Figure 1(b), the GRID scheduler dynamically routes gradients to switches based on available processing capacity. This strategy improves overall resource utilization, achieving effective throughputs of 2.2 Gbps and 4.4 Gbps for the FP8 and FP16 workers, respectively. Due to the L_2 bandwidth limit, the maximum processing capacity for W_5 and W_6 is 1.5 Gbps and 3 Gbps, but S_2 has remaining computational capacity. Therefore, to further increase

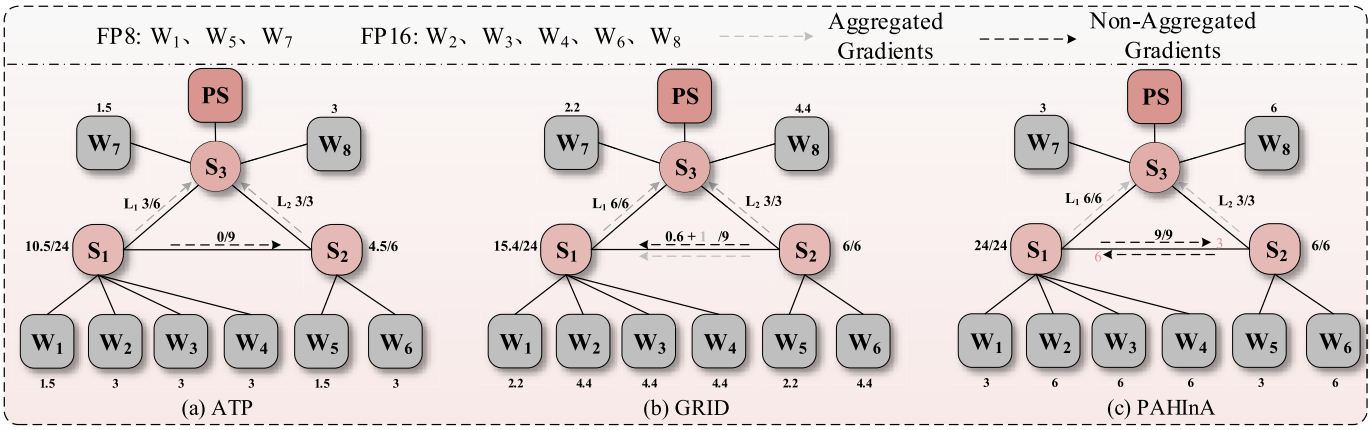


Fig. 1. A distributed training task, containing 1 PS, 8 workers (i.e., $W_1 - W_8$), and 2 programmable switches (i.e., $S_1 - S_2$). The gray dashed arrows represent aggregated gradients, while the black dashed arrows represent non-aggregated gradients. For simplicity, intra-rack transmission arrows are omitted. The term load/capacity denotes the ratio of usage to capacity for links and programmable switches. Figure (a) shows the network workload of ATP, with a minimum gradient sending rate of 1.5 Gbps. Figure (b) shows the network workload of GRID, with a minimum gradient sending rate of 2.2 Gbps. Figure (c) shows the network workload of PAHInA, with a minimum gradient sending rate of 3 Gbps. (It is assumed that for the same training batch, the data size of an FP16 gradient is approximately twice that of an FP8 gradient.).

the gradient sending rate, it forwards some aggregated and non-aggregated data to S_1. This includes 0.6 Gbps of non-aggregated data (0.2 Gbps + 0.4 Gbps) and 1 Gbps of aggregated data (from aggregating 0.5 Gbps and 1 Gbps gradients). This allows the training cluster to reach the maximum sending rates of 2.2 Gbps and 4.4 Gbps for the batch.

B. Our Intuition

As can be seen from the examples above, the technology of in-network aggregation is continuously evolving. ATP uses a default routing method for gradient aggregation, ignoring the data forwarding capabilities of programmable switches. Although GRID combines routing with in-network aggregation, it causes data expansion after aggregation because it does not consider in-network aggregation for heterogeneous precisions, leading to wasted bandwidth. Therefore, as shown in Figure 1(c), we choose to aggregate gradient data of different precisions hierarchically. We route the FP16 gradients from W_6 to S_1 and the FP8 gradients from W_1 to S_2.

This ensures maximum utilization of computation power and avoids the gradient expansion problem caused by heterogeneous precision aggregation. This allows workers to achieve gradient sending rates of 3 Gbps and 6 Gbps. Compared to ATP and GRID, this scheme improves the minimum gradient sending rate by 100% and 36.3%, respectively.

IV. PAHInA Architecture Design

To address the aforementioned challenges, this chapter details our proposed PAHInA framework. We focus on its decoupled control and data plane architecture, highlighting how the data plane leverages a hierarchical aggregation service and XDP technology to efficiently process heterogeneous gradient flows.

A. PAHInA Overview

The architecture of PAHInA, illustrated in Figure 2, is logically decoupled into a data plane and a control plane.

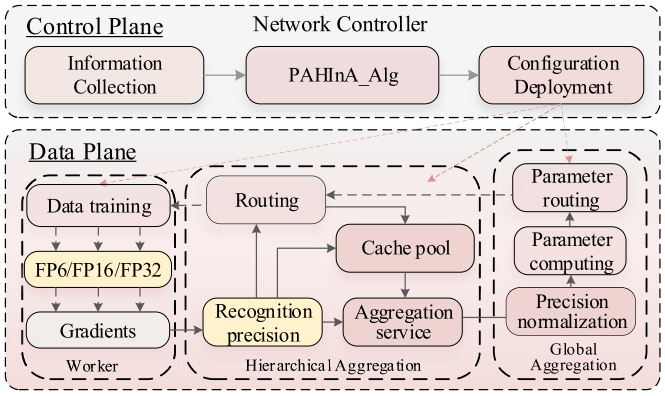


Fig. 2. PAHInA System Overview.

The data plane comprises the physical training infrastructure. This includes the workers, which compute gradients at various numerical precisions; the programmable switches, which execute in-network routing and aggregation of gradient data; and the PS, which receives the final aggregated gradients to update and distribute the global model. The control plane acts as the system's intelligence, operating within a well-defined trust model designed for a single administrative domain. It gathers real-time state information, network link bandwidths and switch resources are monitored via legitimate administrative access, while crucial application details like worker precision are communicated explicitly through a management API during job submission. Using this trusted, global view, it dynamically computes an optimal, precision-aware routing strategy, triggered both periodically per epoch and reactively to significant network events like congestion, that directs gradients to suitable switches for hierarchical aggregation. This strategy forms the core of PAHInA's optimization. This chapter details the PAHInA architecture. We begin with the design of the data plane, followed by a description of the control plane's novel routing algorithm in the subsequent chapter.

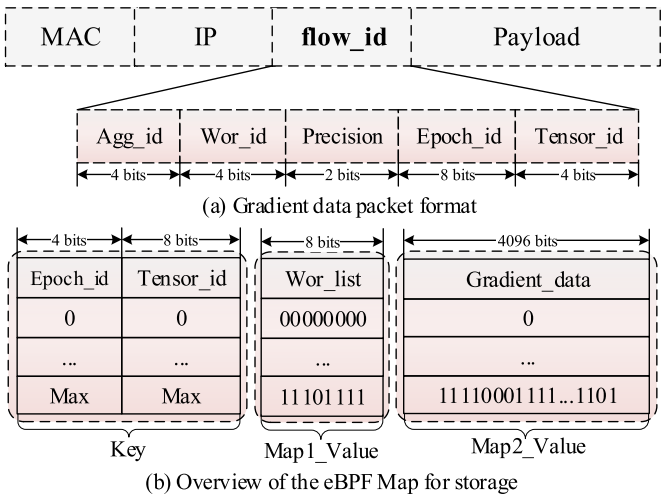


Fig. 3. (a) Shows the format of the gradient packet sent during training; we add a $flow\_id$ in the metadata region to distinguish the worker, batch, and group. (b) shows our two BPF-maps, which are used for gradient data lookup and storage during in-network aggregation. These two maps use the same type of key.

B. Hierarchical Aggregation

The data plane is responsible for the efficient and scalable execution of the complex routing and aggregation strategies computed by the control plane. To achieve this with high performance and resource efficiency suitable for the edge, we leverage the eXpress Data Path (XDP) framework, which enables fast, in-kernel packet processing without the operational overhead of kernel-bypass alternatives. Our hierarchical aggregation scheme is then implemented via two core components running atop this foundation: a Routing Service and an Aggregation Service.

Routing Service: The control plane establishes communication with each worker via an out-of-band channel. After calculating the optimal aggregation path for a gradient flow, the control plane does not directly configure intermediate network devices. Instead, it provides the source worker with the IP address of its destination aggregation node and a unique $flow\_id$, whose format is shown in Figure 3(a), is embedded within the payload of a standard UDP datagram. This encapsulation ensures that to any intermediate device not equipped with our XDP program, the packet appears as conventional traffic and is forwarded based on its destination IP address, guaranteeing interoperability. The $flow\_id$ itself includes five key fields: agg_id represents the destination aggregation switch; Wor_id represents the sending worker; $Precision$ identifies the gradient's precision; and $Epoch\_id$ and $Tensor\_id$ represent the training batch and the specific tensor group, respectively.

When a worker sends a gradient packet, it embeds this $flow\_id$ into a custom UDP payload header and uses standard IP routing to send the gradient packet to the destination aggregation node. After an intermediate aggregation node (or switch) in the network receives a heterogeneous precision gradient packet, the XDP program running on it performs a highly efficient policy match. The program first parses the agg_id to determine if the received gradient data should be aggregated at this switch. If it needs to be forwarded to another switch, it then finds the aggregation node corresponding to the agg_id by matching the agg_id with an IP address, thereby achieving the routing and forwarding of the data.

Aggregation Service: When a gradient packet's $flow\_id$ is parsed for the "local aggregation" policy, the aggregation service is efficiently triggered within the node's kernel space. We designed a fine-grained, two-level aggregation mechanism to handle two tasks of different granularities: "reassembly of tensor fragments within a single worker" and "global batch aggregation among multiple workers", which ensures the efficiency, robustness, and data integrity of the entire process. The core of the aggregation service is two bpf-maps: a State Tracking Map and a Data Storage Map (hereafter referred to as Map1 and Map2, respectively). Their fields are shown in Figure 3(b). It is important to note that the bit-widths shown in the figure are purely illustrative. These fields are configurable parameters designed to scale; for instance, the Wor_id field and the corresponding Wor_list bitmap can be expanded to support thousands of workers per aggregation task.

When a packet carrying the $Epoch\_id$, Wor_id, and $Tensor\_id$ tuple arrives at the switch, the aggregation logic is triggered. The program first uses a composite key, formed from the $Epoch\_id$ and $Tensor\_id$, to look up the state of the corresponding aggregation task. If no entry for this key is found in $Map1$ (the State Tracking Map), the packet is identified as the initiation signal for a new aggregation task. In this case, the program immediately creates a new entry in $Map2$ (the Data Storage Map), seeding its $Gradient\_data$ with the gradient data from the current packet as the initial accumulated value. Concurrently, a corresponding state entry is created in Map1, and the bit representing the source Wor_id is set to 1 in the Wor_list bitmap. After this initialization, the packet is dropped, as its data has been successfully stored. Conversely, when a subsequent (non-final) packet for an existing task arrives, the program finds the corresponding entry in $Map1$. It first checks the Wor_list bitmap to verify that the gradient from this packet's source worker has not already been processed, preventing duplicate aggregation. If the worker is new, the core accumulation operation is triggered: the program retrieves the current accumulated gradient from $Map2$, performs an element-wise summation with the newly arrived gradient, and writes the updated result back into $Map2$. Subsequently, the bit for the new worker is set to 1 in $Map1$'s Wor_list. Having been successfully incorporated into the aggregate, this intermediate packet is also dropped.

The arrival of the "last packet" is the critical turning point, identified when setting the current worker's bit causes the Wor_list bitmap to become full. At this moment, the program performs the final in-packet aggregation, combining the accumulated gradient from $Map2$ with the data from this last packet and overwriting the packet's payload with the final result. Immediately after, it deletes the task's entries from both $Map1$ and $Map2$ to free kernel resources. This modified packet, now carrying the globally aggregated gradient, is then passed to the kernel's standard networking stack via the

XDP_PASS action for routing to the PS, completing the entire in-kernel process.

To ensure the liveness of the aggregation process, a robust and timely timeout mechanism is critical. Our implementation discards the inefficient approach of user-space polling, instead utilizing a significantly more performant in-kernel, event-driven timer mechanism based on modern eBPF capabilities, specifically bpf_timer. This approach assumes capable edge nodes running suitable Linux kernels. When the first packet belonging to a new aggregation task arrives, the XDP program initializes and starts a per-task kernel timer using bpf_timer_start, atomically associating it with the aggregation state stored in the BPF map. The subsequent recovery process operates with high efficiency. If the aggregation task completes successfully before the timer expires, the XDP program simply cancels the timer via bpf_timer_cancel, incurring minimal overhead. Conversely, should the timer expire prior to task completion, the kernel automatically executes a predefined BPF callback function entirely within the kernel space. This callback function accesses the BPF-map, identifies the missing worker(s) by inspecting the Wor_list bitmap, and efficiently triggers the necessary recovery action, such as queuing a notification for a lightweight user-space agent to send targeted NACKs. This design confines time-sensitive detection logic entirely to the kernel, eliminating polling and context-switching overhead, thereby offering a scalable, low-latency solution for packet loss recovery on suitable edge infrastructure.

C. Global Aggregation

Global aggregation is the convergence stage of the distributed training process. Its core task is to gather all intermediate gradients that have undergone hierarchical aggregation at the PS to complete the final, globally consistent model parameter update. The performance and accuracy of this stage are critical to the overall training efficiency. When the XDP program deployed at the ingress port of the PS receives gradient data streams from different aggregation nodes, it first performs precision normalization, unifying potentially different-precision aggregated results by converting them to a high-precision format to ensure computational accuracy. Subsequently, the PS performs a final accumulation of all normalized gradients and, based on the selected optimizer algorithm, calculates the final model update value, completing one training iteration.

V. PAHInA Algorithm Design

This chapter focuses on the core intelligence of PAHInA, its control plane scheduling algorithm. We first mathematically formulate the problem, then detail our proposed two-stage heuristic algorithm, which comprises a priority-driven routing assignment and a global rate optimization model. Finally, we theoretically prove its effectiveness.

A. Problem Formulation

To precisely describe our algorithm, we first present a formal model of the physical environment of the heterogeneous edge network, the characteristics of the training tasks, and the optimization objective. We abstract the entire network as a graph comprising a set of nodes and links with varying roles and capabilities. Our model is designed to capture the core challenges within this environment, including the heterogeneity of node processing capabilities, the constraints of link bandwidth, and the diversity of task requirements. We denote the set of Workers in the network as N. These nodes are the sources of gradient computation. The physical backbone of the network is composed of a set S_{total}, which includes all physical switches, and a set E, representing the physical links connecting all nodes. In a real-world edge environment, not all switches are capable of performing computational tasks. Therefore, we define a fixed subset $S \subset S_{total}$ to represent the programmable switches that possess gradient aggregation capabilities. These programmable switches, together with the central PS α, form the set of all potential nodes that can execute aggregation tasks, which we denote as $V = S \cup \alpha$.

The core of our model lies in addressing heterogeneity. First, each worker $n \in N$ operates at a specific computation precision p_n from a set of available precisions P (e.g., FP8, FP16, FP32). Correspondingly, each potential aggregator $v \in V$ has a capability range, where we use the set p_v to denote the precision types it can support. For a programmable switch $s \in S$, its processing capacity is limited and differentiated; we use $C_{s,p}$ to represent the maximum processing capacity of programmable switch s for generating the aggregated output stream after performing aggregation on gradients of precision p. Second, the heterogeneity of network links is reflected in their bandwidth. We denote the bandwidth capacity of each physical link $e \in E$ as B_e In this model, the ingress bandwidth of the PS is treated as the bandwidth of its directly connected physical link. To make optimal decisions in the complex network environment, we introduce several key evaluation metrics and principles. We use a path cost $d(n, v)$ to measure the quality of the route from a worker n to an aggregation node v. To align with the overall goal of maximizing system throughput, this cost is specifically defined as the inverse of the bottleneck bandwidth on the path, which directly prioritizes higher-throughput routes. This cost can incorporate factors such as hop count and path bottleneck bandwidth, with the physical route being defined by the set of links $Path(n, v)$. To enable prioritized service for critical tasks, such as those requiring high precision, we design a priority function $\Pi(n)$ that assigns a unique priority to each worker. Finally, to implement a resource allocation scheme more sophisticated than simple fairness, we introduce a rate-weighting factor w_p for each precision p, which guides the final rate allocation according to predefined proportions.

Based on the preceding definitions, our algorithm aims to solve for the following decision and optimization variables. In the first stage, the core output is a routing assignment map $A : N \to V$, which uniquely specifies the aggregation destination for each worker. In the second stage, the algorithm solves for a globally optimized base rate unit m, and subsequently calculates the final sending rate r_n for each worker, as well as the intermediate aggregate flow rate $y_{s,p}$ from each switch to the PS.

Algorithm 1 Resource Allocation

Require: N, V, S_{total}, E, $\{p_n\}$, $\{P_v\}$, $\{C_{s,p}\}$, $\{d(n,v)\}$, $\{\Pi(n)\}$, r_{est} (estimated rate).
Ensure: Routing allocation mapping A

Initialization: For all $s \in S$, $p \in P_s$, set used capacity $C'_{s,p} = 0$; set A as an empty mapping.
Sorting: Sort all workers in set N in descending order based on priority function $\Pi(n)$ to obtain list N_sorted.
for n in N_sorted **do**
  $p_n = n$'s precision
  // Filter and sort candidate nodes
  $V\_candidate = \{v \in V | p_n \in P_v\}$
  Sort $V\_candidate$ in ascending order based on path cost $d(n,v)$
  $is\_allocated = false$
  for v in $V\_candidate$ **do**
    if v is a switch and $C'_{s,p_n} + r_{est} \leq C_{s,p_n}$ **then**
      $A(n) \leftarrow s$; $C'_{s,p_n} \leftarrow C'_{s,p_n} + r_{est}$
      $is\_allocated = true$; **break**
    else if v is a PS **then**
      $A(n) \leftarrow \alpha$; $is\_allocated = true$; **break**
    end if
  end for
  If not $is\_allocated$: $A(n) \leftarrow \alpha$ // Fallback strategy
end for
return A

B. Algorithm Design

The PAHInA algorithm operates by decoupling the complex joint routing and rate optimization problem, which is executed in two sequential stages.

Priority-Driven Routing Assignment: This stage employs an efficient heuristic greedy algorithm to determine a unique aggregation node for each worker. The core principle of the algorithm is to allow high-priority tasks to preferentially select aggregation nodes that offer the lowest path cost and have sufficient resources.

As shown in Algorithm 1, the process begins by globally sorting all workers according to the predefined priority function $\Pi(n)$. To capture the dual heterogeneity of edge systems, we define a task's priority by coupling its data properties (device heterogeneity) with its network environment (network heterogeneity). The function is formally defined as:

$$\Pi(n) = w_{size} * NormSize(p_n) + w_{cost} * NormCost(n) \tag{1}$$

where $NormSize(p_n)$ is the normalized data size of the gradient based on its precision p_n, and $NormCost(p_n)$ is the normalized minimum path cost from worker n to any available aggregation node. The weights w_{size} and w_{cost} are configurable hyper-parameters that balance these two factors. This function ensures that tasks more sensitive to network latency and computational resources, such as high-precision gradient tasks, are given precedence. Following this, the algorithm iterates through the sorted list, assigning an aggregation node to each worker in sequence. For the current worker n to be assigned, the algorithm first filters all potential aggregation nodes to identify candidates capable of handling its precision p_n. These candidates are then sorted in ascending order based on the path cost d(n,v).

In the decision-making phase, the algorithm checks if the current best candidate node v, has sufficient remaining capacity to accept the worker (this is evaluated using a conservative estimated rate r_{est}). The term r_{est} is a configurable parameter, used as a placeholder to ensure a balanced initial assignment before the LP stage computes the precise rates. It is typically set to a fraction of the average worker's access link bandwidth. If the capacity is sufficient, the assignment is made immediately: n is allocated to v, and the used capacity of v is updated. The search for the current worker n is then terminated, and the algorithm proceeds to the next worker in the list. If the best candidate's capacity is insufficient, the algorithm discards it and considers the next-best option from the list. This process repeats until a suitable host is found for the current worker. Should all candidate nodes fail to meet its capacity demands, the algorithm employs a fallback strategy, such as forcibly assigning the worker to the PS, which is assumed to have more abundant resources to guarantee the completeness of the assignment. This process ensures that high-precision gradients, which are more sensitive to network latency, are preferentially assigned to optimal physical paths. Meanwhile, low-precision tasks utilize the remaining network resources that may have higher path costs, thus achieving differentiated service for heterogeneous tasks.

Global Weighted Rate Optimization: Once the routing assignment map A is determined, the original problem transforms into a resource allocation problem on a fixed topology. We then formulate and solve a Linear Programming (LP) model to maximize a global base rate unit m. The objective is to achieve Weighted Proportional Fairness, a model specifically chosen to handle the heterogeneous nature of gradient sizes. Unlike max-min fairness, which would inefficiently grant identical rates to all workers, this approach assigns bandwidth proportional to the data payload of each worker's precision. This is accomplished by setting the rate-weighting factor w_p to be proportional to the data size of precision p, a relationship directly enforced by the constraint $r_n = w_{p_n} * m$. Next, we will describe the constraints of the PAHInA model, and the problem can be formulated as follows:

$$\text{maximize} \quad m$$

$$\text{s.t.} \quad \begin{cases} r_n = w_{p_n} * m, & \forall n \in N \\ N_{s,p} = \{n \in N \mid A(n) = s \wedge p_n = p\} & \\ N_e = \{n \in N \mid e \in Path(n, A(n))\} & \\ Y_e = \{(s,p) \mid s \in S, p \in P_s, and\ e \in Path(s,\alpha)\} & \\ y_{s,p} \leq C_{s,p}, & \forall s \in S, \forall p \in P_s \\ \sum_{n \in N_e} r_n + \sum_{(s,p) \in Y_e} y_{s,p} \leq B_e, & \forall e \in E \\ r_n \leq y_{s,p}, & \forall n \in N_{s,p}, \forall s \in S, \forall p \in P_s \\ x_n^v \in \{0,1\}, & \forall n \in N, s \in S \cup \{\alpha\} \\ r_n \geq 0, & \forall n \in N, s \in S \cup \{\alpha\} \\ y_{s,p} \geq 0, & \forall s \in S, \forall p \in P \end{cases} \tag{2}$$

(1) Set of Workers Assigned to a Specific Switch-Precision Pair: To constrain the processing capacity of each switch s for each precision p, we define the set $N_{s,p}$. This set contains all workers that are assigned to switch s and have a precision of p.
(2) Set of Workers Whose Traffic Traverses Link e: To calculate the raw gradient traffic carried on each link e, we define the set N_e. This set includes all workers whose data transmission path traverses link e.
(3) Set of (Switch, Precision) Pairs whose Aggregated Flow Traverses Link e: In order to calculate the post-aggregation traffic on each link e, we define the set F_e. This set represents all aggregated flows that originate from a switch s with precision p, and whose path to the PS traverses link e.
(4) Proportional Rate Definition: This constraint is the cornerstone for achieving weighted fairness. It directly links the final sending rate $r_n$of each worker n with the global base rate unit m. By introducing the predefined precision-weighting factor $w_(p_n)$, we ensure that high-precision tasks are allocated higher bandwidth in proportion to their data volume and importance, rather than relying on a simple equal-sharing strategy. The constraint is defined as: $r_n = w_{p_n} * m$.
(5) Switch Per-Precision Capacity: This constraint ensures that computational resources within the network are not over-utilized. For each programmable switch s and for each precision p it supports, the rate of the aggregated output stream $y_{s,p}$ generated by it must not exceed the switch's processing capacity limit $C_{s,p}$ for that specific precision p.
(6) Synchronization: To ensure the proper execution of in-network aggregation, the rate of a worker r_n entering a switch s must not exceed the rate of the corresponding aggregated stream $y_{s,p}$ that leaves the switch. This constraint prevents the unbounded accumulation of data within switches due to rate mismatches and serves as a logical guarantee for maintaining steady-state operation of the system.
(7) Unified Bandwidth Constraint: This constraint is key to ensuring that the physical limitations of the network are met. For any given physical link e in the network, the total traffic it carries must not exceed its bandwidth capacity B_e. This total traffic is composed of two parts: first, the sum of rates Σr_n from all raw gradient flows that traverse this link; and second, the sum of rates $\Sigma y_{s,p}$ from all post-aggregation flows that also traverse this link.

By solving this LP model, we obtain the optimal base rate m^*. Based on this, the final sending rate for each worker is calculated as $r_n^* = w_{p_n} * m^*$.

C. Performance Analysis

Theorem 1: The PAHInA algorithm strictly enforces the constraints of single aggregation and precision matching.

Proof: The correctness of this theorem can be directly derived from the intrinsic logic of Algorithm 1. First, regarding the single-aggregation principle, the algorithm searches for an aggregation destination for each worker n in the main loop. Once a suitable node v is found and assigned in the inner loop (Line 9), the $break$ statement (Line 12 or 14) immediately terminates the search for the current node n. This means that for any given $n \in N$, the assignment operation $A(n) \leftarrow v$ occurs at most once during the entire execution of the algorithm. This guarantees that each gradient flow has a unique target aggregation node, the mathematical expression for which is:

$$|A(n)| = 1 \tag{3}$$

Second, for the precision-matching principle, the algorithm imposes an explicit filtering condition when constructing the pool of candidate nodes (Line 6). The formal mathematical expression for this condition is:

$$V_{candidate} = \{v \in V \mid p_n \in P_v\} \tag{4}$$

Since the finally assigned node $A(n)$ must be a member of this candidate set $V_{candidate}$, it is naturally guaranteed that the selected aggregation node has the capability to process the corresponding gradient's precision, ensuring the numerical consistency of the computation.

Theorem 2: PAHInA's routing assignment strategy achieves priority-based path optimization.

Proof: The theorem aims to prove that higher-priority tasks are assigned to paths with better (or at least not worse) path costs. We consider two arbitrary worker $i, j \in N$ and assume their priorities satisfy:

$$\Pi(i) > \Pi(j) \tag{5}$$

according to the sorting step of Algorithm 1 (Line 2), node i is necessarily processed before node j. We define $C'_{s,p}(t)$ as the allocated capacity on switch s for precision p when the algorithm is processing the t-th node. When the algorithm processes node i (assumed to be the k-th node processed), it faces the resource state $\{C'_{s,p}(k)\}$. Its candidate node set $V_{candidate}(i)$ is composed of all nodes that satisfy the precision and capacity constraints:

$$V_{candidate}(i) = \{v \in V \mid p_i \in P_v \wedge C'_{v,p_i}(k) + r_{est} \leq C_{v,p_i}\} \tag{6}$$

the algorithm's choice is based on minimizing path cost, so the assignment result is:

$$A(i) = arg \min_{v \in V_{cand}(i,R_i)} d(i,v) \tag{7}$$

Subsequently, when the algorithm processes node j (the l-th node processed, where $l > k$), the resource state becomes $\{C'_{s,p}(l)\}$. Since the assignment of node i consumed resources, it must be that:

$$\forall s, p, \quad C'_{s,p}(l) \geq C'_{s,p}(k) \tag{8}$$

this implies that the candidate set for node j is a subset of or equal to the candidate set for node i:

$$V_{candidate}(j, R_j) \subseteq V_{candidate}(j, R_i) \tag{9}$$

Therefore, the choice space for node i is always superior or equal to the choice space for node j, which mechanistically guarantees that higher-priority tasks receive better path assignments.

Theorem 3: The PAHInA algorithm strictly enforces all resource capacity constraints.

Proof: The correctness of this theorem is directly guaranteed by the Global Weighted Rate Optimization model, as any solution returned by an LP solver must be a feasible solution that satisfies all constraints.

First, for the switch capacity constraint, Constraint (6) in the LP model explicitly stipulates that for any switch $s \in S$ and any precision $p \in P_s$, the sum of the rates of all gradient flows routed to it for aggregation at precision p cannot exceed its processing limit:

$$\sum_{n \in N_{s,p}} r_n^* \leq C_{s,p}, \ where \ N_{s,p} = n \in N \mid A(n) = s \wedge p_n = p \tag{10}$$

second, for the link bandwidth constraint, Constraint (8) guarantees that for any physical link $e \in E$, the total traffic flowing through it cannot exceed its physical bandwidth B_e. This total traffic is defined as the sum of all raw gradient flow rates and all post-aggregation gradient flow rates whose paths traverse the link, formally expressed as:

$$\sum_{n \in N_e} r_n^* + \sum_{(s,p) \in F_e} y_{s,p}^* \leq B_e \tag{11}$$

Therefore, the final output of the algorithm must strictly adhere to all predefined physical resource limits.

Theorem 4: Pareto optimality of rate allocation, Given the fixed routing assignment A determined in Priority-Driven Routing Assignment, the subsequent rate allocation $\{r_n^*\}$ produced by Global Weighted Rate Optimization of the PAHInA algorithm is Pareto Optimal.

Proof: The proof is by contradiction. For a fixed routing assignment A, let m^*be the optimal global base rate that maximizes the LP objective, and let $R^* = r_n^*$ be the corresponding vector of optimal rates where $r_n^* = w_{p_n} * m^*$. We assume, for the sake of contradiction, that this rate allocation R^* is not Pareto Optimal.

By the definition of Pareto sub-optimality, this assumption implies the existence of another feasible rate vector $R' = r'_n$, which satisfies all system constraints and for which (a) $r'_n \geq r_n^*$ for all $n \in N$, and (b) there exists at least one worker $j \in N$ such that r'_j¿r_j^*. Since R' must also adhere to the proportional rate constraint, it must be generated by some base rate m', where $r'_n = w_{p_n}$. The strict inequality for worker j therefore leads to the conclusion that m'¿m^*, as shown by:

$$\begin{aligned} & r'_j > r_j^* \\ \Rightarrow & \ w_{P_j} * m' > w_{P_j} * m^* \\ \Rightarrow & \ m' > m^* \end{aligned} \tag{12}$$

The existence of a feasible solution R' generated by a base rate m' that is strictly greater than m^* contradicts the very premise that m^* is the optimal solution from the LP maximization. It is impossible for a feasible solution to be generated by a value greater than the defined maximum. Thus, the initial assumption must be false. We conclude that for the given routing assignment, no feasible allocation can improve one worker's rate without harming another's, and the rate allocation R^* is therefore Pareto Optimal.

VI. Evaluation

In this chapter, we compare PAHInA with current state-of-the-art research schemes. We present the performance metrics and baselines. We build a small-scale testbed using XDP programmable data plane technologies to evaluate the training performance of PAHInA. At the same time, to compensate for the limitations of small-scale experiments, we employ data-driven modeling for simulation-based training to demonstrate the feasibility of PAHInA in large-scale distributed training scenarios.

A. Performance Metrics and Benchmarks

Performance Metrics: To highlight the advantages of PAHInA in heterogeneous precision in-network aggregation, the following metrics were used in the micro-benchmarks: (1) Training Duration; (2) Training Throughput; (3) Training Efficiency; (4) In-network Aggregation Efficiency; and (5) PS Aggregation Efficiency.

We first calculate the total Training Duration (1) required to complete all training epochs, which is obtained by multiplying the total time per iteration by the total number of iterations, providing a macroscopic measure of the end-to-end task completion time. To evaluate the overall processing speed of the system, we derive the System Throughput (3) by dividing the Global Batch Size by the total time per iteration, which reflects the number of samples the entire system can process per second. To measure the raw communication pressure in the network, we calculate the Gradient Sending Rate (4) by recording the total amount of gradient data generated by all workers in a batch and dividing it by the total duration of the upload phase. We have redefined In-network Aggregation Efficiency (5) from a focus on the slowest path delay to a focus on the data aggregation rate. This efficiency value precisely represents the percentage reduction in data volume achieved through aggregation operations during the process of all gradient data from its source to the final PS; a higher value represents a more significant bandwidth saving effect. Finally, we quantify the PS Aggregation Efficiency (6) by using the time the PS actually spends on aggregation computation to evaluate its processing capacity as a central node.

Benchmarks: We compare our core algorithm, PAHInA, against three baseline algorithms. The first baseline is the Standard PS architecture. This is a traditional communication scheme that does not utilize in-network aggregation. In this model, after completing local computations, all workers find their best-effort path to the central PS and send gradients directly, with the PS performing all aggregation tasks. The second is ATP, which executes in-network aggregation on multiple programmable switches. Each worker sends gradients via a predefined routing path to the PS, and the gradient data is aggregated in the first switch on the path that has available

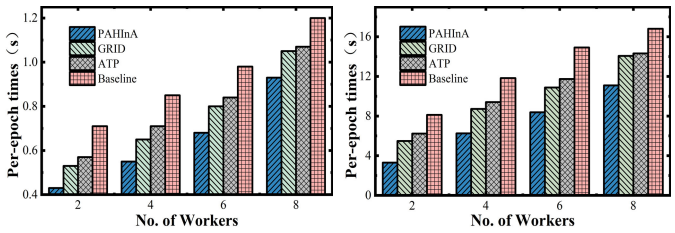
processing capacity, after which the aggregated result is sent to the PS. The final baseline is GRID, a centralized routing scheduling scheme based on Linear Programming. GRID aims to find a globally optimal gradient routing strategy to maximize system throughput, but it does not consider precision differences between gradients in its decision-making. We use it as an advanced, precision-unaware comparative scheme.

B. Testbed Evaluation

Setup: To evaluate and compare the performance of different gradient aggregation strategies in distributed training, we built a virtualized experimental environment based on a single physical server. This environment aims to simulate a typical data center computing cluster while ensuring the reproducibility and isolation of experiments. The experiment relies on a high-performance server equipped with an Intel Xeon Gold series multi-core processor and 64GB of DDR4 memory. We used Kernel-based Virtual Machine (KVM) [41] technology to create all workers and the PS. Each virtual node was allocated different vCPUs and memory and ran the *Ubuntu Server* 20.04 *LTS* operating system. In terms of network configuration, since all nodes are on the same host machine, we used a linux bridge to construct a tree-like network topology, simulating a scenario where all nodes converge to the PS. To more realistically reflect potential link quality differences and congestion fluctuations in a data center network, we set the link bandwidth from each virtual machine to the virtual switch to a random value that dynamically changes between 10 Gbps and 25 Gbps. Such a configuration not only supports the high gradient transmission rates observed in the experiments but also provides a basis for evaluating the robustness of aggregation algorithms in non-ideal network environments. We assume that the network latency between nodes is extremely low (<0.1ms), and the performance bottleneck is mainly reflected in bandwidth and data processing.

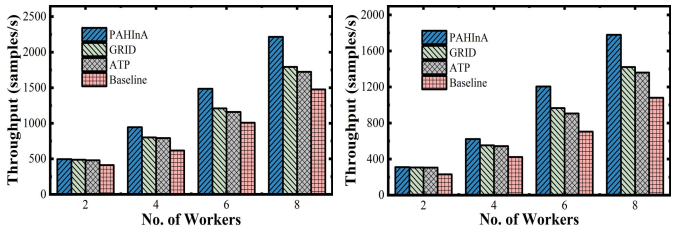
We conducted tests on two classic convolutional neural network models to evaluate the algorithm's performance under different computational loads and model sizes. The models we used are LeNet-5 [42] and ResNet-50 [43], both trained on the CIFAR-10 dataset [44]. We conducted experiments with configurations of 2, 4, 6, and 8 workers, using a separate node as the PS. The training used Stochastic Gradient Descent (SGD) [45] with momentum, and the global batch size was set to 512. To create a consistent yet heterogeneous environment, our testbed configurations maintained a fixed number of two workers operating at FP8 precision, while the remaining workers (1, 2, 4, or 6, corresponding to total worker counts of 2, 4, 6, and 8) operated at FP16.

Per-epoch time: Per-epoch time is the most direct metric for evaluating the end-to-end performance of a distributed training system. As shown in Figure 4, the training time for all the tested algorithms increases as the number of worker increases, which verifies that the continuous growth of workers in the cluster leads to a communication bottleneck. But in the comparative analysis of the algorithms, we observe a stable and clear performance ranking: PAHInA achieves the fastest training completion in all configurations, followed in order by GRID, ATP, and the Baseline scheme which takes the longest. This performance advantage is particularly prominent in larger-scale or more complex scenarios. For example, in the scenario of training ResNet-50 with 8 workers, PAHInA's training time is only 3.38 seconds, a 58.4% performance improvement compared to the Baseline's 8.12 seconds, demonstrating excellent efficiency.



(a) LeNet-5 (b) ResNet-50

Fig. 4. Per-epoch time vs. No. of workers.

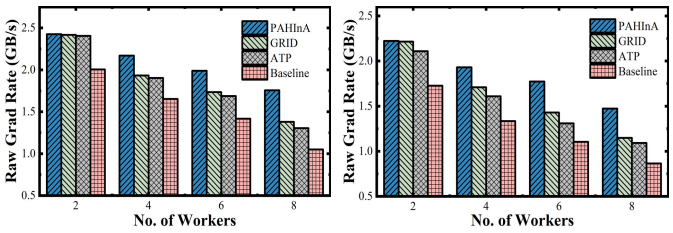


(a) LeNet-5 (b) ResNet-50

Fig. 5. Throughput vs. No. of workers.

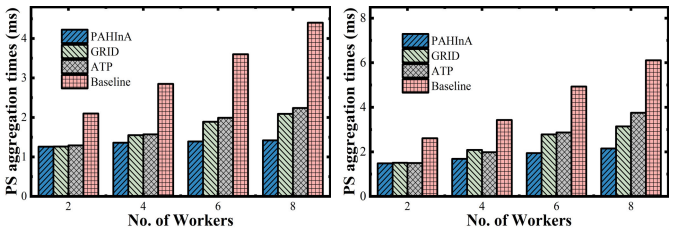
The fundamental reason for the difference in training time lies in communication overhead. In the Baseline scheme, all gradient flows are directed to a single PS, which easily causes network congestion. ATP and GRID alleviate the pressure on the central node by performing partial aggregation in the network. However, ATP is not sensitive to network state, and GRID ignores the heterogeneity of gradient sizes. PAHInA, by virtue of its dual-awareness of network paths and gradient precision, achieves optimized scheduling of communication resources, thereby minimizing communication delay and achieving the fastest training speed.

Throughput: As shown in Figure 5, system throughput exhibits an ideal inverse correlation with training time. As the number of workers increases, the throughput of all strategies improves significantly. In terms of performance ranking, PAHInA also demonstrates the strongest processing capability, followed again by GRID, ATP, and the Baseline. To more clearly showcase PAHInA's advantage, we compare it with the most advanced baseline algorithm, GRID. When training ResNet-50 with 8 workers, PAHInA's throughput reached 1779.6 samples/s, which is a 25.3% and 30.9% improvement compared to GRID's 1420.3 samples/s and ATP's 1359.9 samples/s, respectively. System throughput is a direct reflection of training speed. By minimizing communication waiting time, PAHInA enables workers to complete gradient synchronization faster and start the next round of computation, which greatly improves the effective utilization rate of computational units. In contrast, even an advanced algorithm like GRID, due to its insensitivity to the heterogeneity of gradient sizes, still



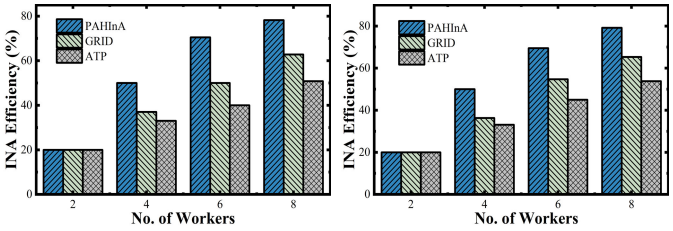
(a) LeNet-5 (b) ResNet-50

Fig. 6. Gradients sending rate vs. No. of workers.



(a) LeNet-5 (b) ResNet-50

Fig. 8. Aggregation time of PS vs. No. of workers.



(a) LeNet-5 (b) ResNet-50

Fig. 7. Efficiency of INA vs. No. of workers.

produces sub-optimal solutions in its routing decisions, leading to limited communication efficiency for some nodes, thereby restricting further increases in overall throughput.

Gradient Sending Rate: This evaluation metric reflects the ability of the entire system to effectively utilize network bandwidth during the gradient upload phase. As shown in Figure 6, PAHInA achieved the highest effective gradient sending rate in all tests, significantly outperforming the other comparison algorithms. For example, in the scenario of training ResNet-50 with 8 workers, PAHInA's gradient transmission rate reached 1.76 GB/s, which is a 27.5% and 35.0% improvement compared to GRID's 1.38 GB/s and ATP's 1.30 GB/s, respectively. This indicates that PAHInA can complete the gradient upload phase in the shortest amount of time, with the highest network resource utilization. For ResNet-50, which has a much larger number of model parameters, the amount of gradient data it produces far exceeds that of LeNet, leading to higher absolute values for the gradient sending rates of all algorithms. In this scenario, the rate difference between the different algorithms is also more pronounced, further highlighting the importance of PAHInA's efficient aggregation strategy. A higher effective sending rate means shorter communication waiting times.

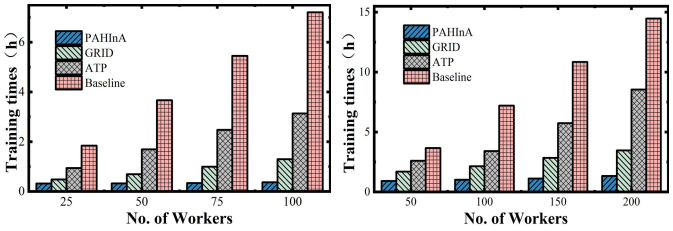
Efficiency of In-Network Aggregation: As shown in Figure 7, the efficiency of all three in-network aggregation algorithms improves as the number of workers increases. This is because more nodes provide more opportunities for aggregation, increasing the potential for data to be compressed before it reaches the PS. However, the algorithms differ significantly in their ability to capitalize on these opportunities. PAHInA demonstrates the highest aggregation efficiency in all test scenarios. For example, when training ResNet-50 with 8 workers, PAHInA's data compression rate reached 79.2%, while GRID and ATP were at 65.3% and 53.8%, respectively. This indicates that PAHInA successfully and effectively aggregated nearly 80% of the raw gradient data within the network, greatly alleviating the transmission pressure on subsequent links.

Aggregation time of PS: This metric measures the computation time required for the PS to complete the final aggregation operation and is an important indicator for evaluating the computational pressure on the central node. As is evident from Figure 8, all in-network aggregation strategies (PAHInA, GRID, ATP) can reduce the PS aggregation overhead by an order of magnitude compared to the Baseline, which fully demonstrates the core advantage of distributing aggregation computation into the network.

Building on this, when we further compare the three in-network aggregation strategies, we find that PAHInA performs the best in reducing PS overhead. When training ResNet-50 with 8 workers, PAHInA's PS aggregation time was 2.15 ms, which is a further reduction of 31.6% and 42.7% compared to GRID's 3.14 ms and ATP's 3.75 ms, respectively. This difference stems from the degree of intelligence in each algorithm's routing decisions. ATP uses static shortest paths, and while GRID can perform traffic optimization, it ignores precision heterogeneity. Both of these strategies can lead to sub-optimal aggregation plans, causing some insufficiently aggregated gradient flows to still converge on the PS. In contrast, PAHInA, through its co-optimization of network paths and aggregation tasks, achieves a more efficient hierarchical aggregation, minimizing the number of partial sums arriving at the PS, thereby resulting in the lightest computational burden for the PS.

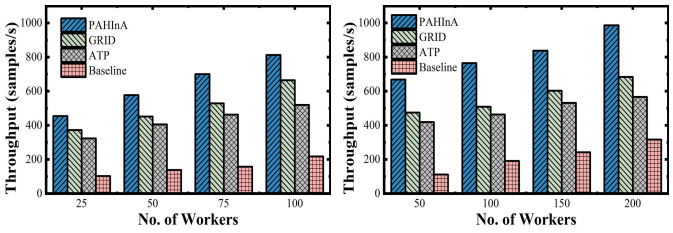
C. Simulation Evaluation

Setup: Our training simulations are implemented on a physical machine with an Intel Core i7-10700 processor and 32 GB of RAM. We evaluate performance across two topologies: an 80-switch, 128-host Fat-Tree [46] with 25 to 100 workers, and a 125-switch, 250-host random network with 50 to 200 workers. To model deep heterogeneity, workers are assigned FP8, FP16, and FP32 precisions in a 5:3:2 ratio. In-network aggregation switches are dynamically assigned at a ratio of approximately 1 per 8 workers. Host-to-switch links are fixed at 10 Gbps, while core link bandwidths between switches are randomly set from 5 to 20 Gbps to simulate network heterogeneity. Accordingly, the path cost $d(n, v)$ is defined as the inverse of the bottleneck bandwidth on the path from worker n to node v, thus prioritizing higher-throughput routes. The processing capacity limit for in-network aggregation on all programmable switches is uniformly set to 100 Gbps.



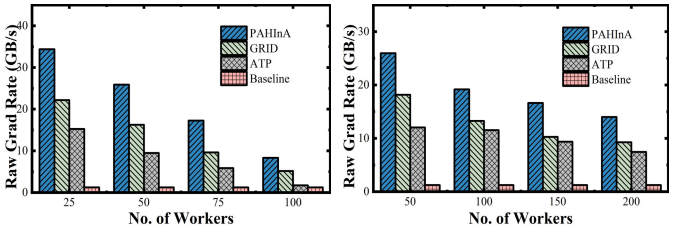
(a) Fattree (b) Random

Fig. 9. Estimated training time vs. No. of workers.



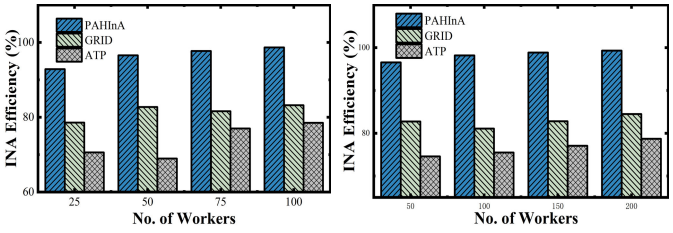
(a) Fattree (b) Random

Fig. 10. Estimated throughput vs. No. of workers.



(a) Fattree (b) Random

Fig. 11. Gradients sending rate vs. No. of workers.



(a) Fattree (b) Random

Fig. 12. Efficiency of INA vs. No. of workers.

This parameter represents the maximum rate of the aggregated output stream that a switch can generate after completing the aggregation operation.

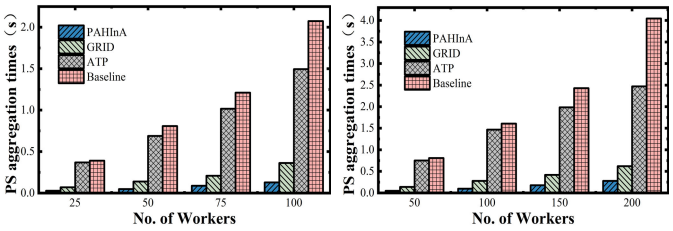
Training Duration Comparison: In this set of simulation experiments, we focus on the total duration of the simulated training, with the results shown in Figure 9. In both the Fat-Tree and random topologies, PAHInA achieved the shortest training time in all tested scenarios. For example, in the 100-worker Fat-Tree topology, PAHInA's training duration was reduced by approximately 87.9%, 72.2%, and 32.9% compared to the Baseline, ATP, and GRID, respectively. In the random topology, while increasing the number of workers from 50 to 200 led to longer training times for all methods due to the less structured communication, PAHInA maintained its lead, with its time increasing from 0.91h to 2.34h, still significantly outperforming the baselines.

Throughput: As shown in Figure 10, the throughput of PAHInA and GRID far exceeds that of ATP, which uses a static strategy, and the Baseline with no aggregation. This highlights that in large-scale environments, dynamic and intelligent routing decisions are a necessary condition for ensuring system performance. PAHInA's superiority is clearly reflected in its higher throughput, particularly evident in the random topology. In the 200-node random topology scenario, PAHInA's throughput is approximately 44.3% higher than GRID. This profoundly reveals the limitation of being purely network-aware in heterogeneous precision scenarios. Edge scenarios are often heterogeneous, not only in the network but also in the size of gradient data produced by end devices. PAHInA, through its unique network-precision dual awareness, can plan the most suitable paths for data packets of different sizes, avoiding the efficiency loss caused by resource mismatch, thereby maximizing the overall system throughput.

Gradient Sending Rate: In large-scale distributed scenarios, the access bandwidth of the central node is often the first bottleneck the system encounters. As shown in Figure 11, the Baseline strategy's gradient sending rate is strictly limited to 1.25 GB/s, which precisely corresponds to the PS's 10 Gbps physical link bandwidth. This result is a classic illustration of the "centralized bottleneck". In contrast, all in-network aggregation algorithms successfully break through this bottleneck by distributing the aggregation pressure across the entire network. PAHInA achieves the highest transmission efficiency in all scenarios, proving its most thorough utilization of the network-wide bandwidth resources. For example, in the 100-node Fat-Tree topology, PAHInA's transmission efficiency reached approximately 8.33 GB/s, a significant 61.9% improvement over GRID's 5.15 GB/s, and it far surpasses ATP's 1.73 GB/s. Compared to other algorithms like GRID, PAHInA's advantage is that it doesn't just disperse traffic, but rather performs intelligent traffic engineering based on data and network state.

Efficiency of INA: In wide-area and edge networks, bandwidth is often a scarce and expensive resource; therefore, in-network aggregation efficiency (data compression rate) is directly related to a scheme's economic viability and feasibility. As shown in Figure 12, PAHInA's aggregation efficiency performs exceptionally well in large-scale scenarios. As the number of nodes increases, its efficiency steadily approaches the theoretical limit of 99%. This means that almost all gradient data is effectively compressed before leaving the network edge and entering the core backbone. In comparison, GRID's efficiency stabilizes in the 80-85% range, while ATP's is even lower. The core reason PAHInA can achieve such high efficiency is its ability to construct deeper and more optimized aggregation hierarchies. By co-perceiving gradient sizes and path states, it can plan optimal multi-hop aggregation paths, ensuring data is maximally compressed at each hop, which is crucial for saving expensive backbone network bandwidth.

Aggregation time of PS: One of the ultimate goals of in-network aggregation is to make the central node "lightweight",



(a) Fattree (b) Random

Fig. 13. Estimated aggregation time of PS vs. No. of workers.

transforming it from a computational bottleneck into a simple coordinator. As shown in Figure 13, PAHInA is the most successful in achieving this goal. In the 200-node random topology scenario, PAHInA's PS aggregation overhead is 54.8% lower than GRID's and nearly 90% lower than ATP's. This indicates that the hierarchical aggregation system built by PAHInA is the most efficient; the vast majority of computational tasks are "digested" at the network edge or intermediate layers, and what finally arrives at the PS is only a very small number of highly aggregated partial sums. In future edge intelligence scenarios where the central server might itself be a resource-constrained edge cloud node, PAHInA's ability to maximally reduce the central load has inestimable value for ensuring system stability and scalability.

VII. Conclusion

In this paper, we addressed a critical challenge at the intersection of edge computing and distributed machine learning: the performance degradation caused by the dual heterogeneity of edge devices and network resources. We identified that conventional in-network aggregation mechanisms are ill-equipped to handle the heterogeneous precision gradients generated by resource-constrained edge devices, leading to precision inflation and an aggravated straggler effect that cripples training efficiency. To overcome these limitations, we introduced the PAHInA framework, the first of its kind to co-optimize gradient precision with network-aware routing for in-network aggregation. PAHInA features a logically decoupled control and data plane architecture. Its intelligent control-plane scheduler formulates the routing of heterogeneous gradients as a joint optimization problem, assigning high-priority, high-precision gradients to premium, low-cost network paths. This strategy is executed by a high-performance data plane that leverages XDP for efficient, hierarchical aggregation in the kernel, minimizing aggregation-induced overhead. Extensive evaluations, conducted on both a small-scale testbed and through large-scale simulations, confirm the significant advantages of our approach. Compared to state-of-the-art baselines, including standard PS, ATP, and GRID, PAHInA consistently demonstrated superior performance. Our framework significantly mitigates network congestion, reducing end-to-end communication time by up to 33% and boosting overall training throughput by approximately 30%.

The results validate that by treating precision as a first-class scheduling metric, PAHInA creates a more efficient and robust distributed training system for the edge. This work represents a significant step toward enabling complex AI model training in resource-constrained and diverse edge environments. Future research could extend this framework in several promising directions. To address the scalability limits of the current centralized control plane in massive, geographically dispersed environments, we plan to evolve PAHInA towards a hierarchical or federated model. Beyond scalability, our framework could be adapted to support fully decentralized architectures like All-Reduce, explore its application in asynchronous training paradigms, and investigate the integration of advanced security protocols for gradient exchange in untrusted edge networks. These advancements would collectively pave the way for robust, efficient, and secure distributed learning in diverse, real-world edge environments.

Acknowledgment

Yingpu Nian and Bo Yi are contributed to Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China.

References

[1] M. Sharma, A. Tomar, and A. Hazra, "Edge computing for Industry 5.0: Fundamental, applications, and research challenges," *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19070–19093, Jun. 2024.

[2] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, Sep. 2023.

[3] I. Ficili, M. Giacobbe, G. Tricomi, and A. Puliafito, "From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI," *Sensors*, vol. 25, no. 6, p. 1763, Mar. 2025.

[4] C. L. Lao et al., "ATP: In-network aggregation for multi-tenant learning," in *Proc. 18th USENIX Symp. Netw. Syst. Design Implement.*, 2021, pp. 741–761.

[5] A. Sapio et al., "Scaling distributed machine learning with in-network aggregation," in *Proc. 18th USENIX Symp. Netw. Syst. Design Implement.*, 2021, pp. 785–808.

[6] J. Fei, C.-Y. Ho, A. N. Sahu, M. Canini, and A. Sapio, "Efficient sparse collective communication and its application to accelerate distributed deep learning," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 676–691.

[7] P. Micikevicius et al., "Mixed precision training," 2017, *arXiv:1710.03740*.

[8] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8604–8612.

[9] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 4466–4475.

[10] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, "Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 197–205, Jun. 2022.

[11] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning with data and device heterogeneity," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 2, pp. 1487–1501, Jun. 2023.

[12] R. Han, M. Si, J. Demmel, and Y. You, "Dynamic scaling for low-precision learning," in *Proc. 26th ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, Feb. 2021, pp. 480–482.

[13] I. Gim and J. Ko, "Memory-efficient DNN training on mobile devices," in *Proc. 20th Annu. Int. Conf. Mobile Syst., Appl. Services*, Jun. 2022, pp. 464–476.

[14] Z. Xiang, Y. Zheng, Z. Zheng, S. Deng, M. Guo, and S. Dustdar, "Cost-effective traffic scheduling and resource allocation for edge service provisioning," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2934–2949, Dec. 2023.

[15] Z. Sun et al., "A resource allocation scheme for edge computing network in smart city based on attention mechanism," *ACM Trans. Sensor Netw.*, Mar. 2024, doi: 10.1145/3650031.

[16] J. Liu, C. Li, and Y. Luo, "Efficient resource allocation for IoT applications in mobile edge computing via dynamic request scheduling optimization," *Expert Syst. Appl.*, vol. 255, Dec. 2024, Art. no. 124716.
[17] A. Ali et al., "Energy-efficient resource allocation for urban traffic flow prediction in edge-cloud computing," *Int. J. Intell. Syst.*, vol. 2025, no. 1, Jan. 2025, Art. no. 1863025.
[18] T. Høiland-Jørgensen et al., "The express data path: Fast programmable packet processing in the operating system kernel," in *Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2018, pp. 54–66.
[19] Y. Chen, Q. Yang, S. He, Z. Shi, J. Chen, and M. Guizani, "FTPipeHD: A fault-tolerant pipeline-parallel distributed training approach for heterogeneous edge devices," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3200–3212, Apr. 2024.
[20] P. Joshi, M. Hasanuzzaman, C. Thapa, H. Afli, and T. Scully, "Enabling all in-edge deep learning: A literature review," *IEEE Access*, vol. 11, pp. 3431–3460, 2023.
[21] N. Provatas, I. Konstantinou, and N. Koziris, "A survey on parameter server architecture: Approaches for optimizing distributed centralized learning," *IEEE Access*, vol. 13, pp. 30993–31015, 2025.
[22] X. Chen, G. Xu, X. Xu, H. Jiang, Z. Tian, and T. Ma, "Multicenter hierarchical federated learning with fault-tolerance mechanisms for resilient edge computing networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 1, pp. 47–61, Jan. 2025.
[23] L. Hui, W. Yang, F. Wu, Y. Wang, F. Lyu, and Y. Zhang, "DirectReduce: A scalable ring AllReduce offloading architecture for torus topologies," *IEEE Internet Things J.*, vol. 12, no. 16, pp. 32951–32964, Aug. 2025.
[24] P. Liu, J. Peng, J. Liu, and L. Chi, "TH-allreduce: Optimizing small data allreduce operation on tianhe system," in *Proc. IEEE 29th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2023, pp. 1903–1911.
[25] F. Shirin Abkenar et al., "A survey on mobility of edge computing networks in IoT: State-of-the-art, architectures, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2329–2365, 4th Quart., 2022.
[26] G. Bao and P. Guo, "Federated learning in cloud-edge collaborative architecture: Key technologies, applications and challenges," *J. Cloud Comput.*, vol. 11, no. 1, p. 94, Dec. 2022.
[27] N. Gebara, M. Ghobadi, and P. Costa, "In-network aggregation for shared machine learning clusters," in *Proc. Mach. Learn. Syst.*, 2021, pp. 829–844.
[28] S. Dong et al., "MINA: Auto-scale in-network aggregation for machine learning service," in *Proc. 7th Asia–Pacific Workshop Netw.*, Jun. 2023, pp. 184–186.
[29] J. Xia, W. Wu, L. Luo, D. Guo, and G. Cheng, "In-network aggregation as a generic service for distributed applications," *IEEE Trans. Netw.*, vol. 33, no. 6, pp. 2977–2992, Dec. 2025, doi: 10.1109/TON.2025.3578180.
[30] J. Xia, G. Cheng, W. Wu, L. Luo, and D. Guo, "Exploring communication-efficient federated learning via stateless in-network aggregation," *IEEE Trans. Mobile Comput.*, vol. 24, no. 8, pp. 7423–7439, Aug. 2025, doi: 10.1109/TMC.2025.3551368.
[31] L. Mai et al., "NetAgg: Using middleboxes for application-specific on-path aggregation in data centres," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2014, pp. 249–262.
[32] Y. Qiu, G. Zhao, H. Xu, H. Huang, and C. Qiao, "PARING: Joint task placement and routing for distributed training with in-network aggregation," *IEEE/ACM Trans. Netw.*, vol. 32, no. 5, pp. 4317–4332, Oct. 2024.
[33] Q. Zhang, G. Zhao, H. Xu, and P. Yang, "XAgg: Accelerating heterogeneous distributed training through XDP-based gradient aggregation," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 2174–2188, Jun. 2024.
[34] P. Yang, H. Xu, G. Zhao, Q. Zhang, J. Liu, and C. Qiao, "ALEPH: Accelerating distributed training with eBPF-based hierarchical gradient aggregation," *IEEE/ACM Trans. Netw.*, vol. 32, no. 5, pp. 4128–4143, Oct. 2024.
[35] J. Fang, G. Zhao, H. Xu, C. Wu, and Z. Yu, "GRID: Gradient routing with in-network aggregation for distributed training," *IEEE/ACM Trans. Netw.*, vol. 31, no. 5, pp. 2267–2280, Oct. 2023.
[36] J. Fang, G. Zhao, H. Xu, Z. Yu, B. Shen, and L. Xie, "GOAT: Gradient scheduling with collaborative in-network aggregation for distributed training," in *Proc. IEEE/ACM 31st Int. Symp. Quality Service (IWQoS)*, Jun. 2023, pp. 1–10.
[37] J. Fang, H. Xu, G. Zhao, Z. Yu, B. Shen, and L. Xie, "Accelerating distributed training with collaborative in-network aggregation," *IEEE/ACM Trans. Netw.*, vol. 32, no. 4, pp. 3437–3452, Aug. 2024.
[38] J. Liu et al., "InArt: In-network aggregation with route selection for accelerating distributed training," in *Proc. ACM Web Conf.*, May 2024, pp. 2879–2889.
[39] L. Luo, X. Chen, S. Yang, W. Fan, and H. Yu, "Accelerating communication for distributed training with collaborative in-network aggregation," in *Proc. IEEE 24th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2024, pp. 681–687.
[40] Y. Li et al., "Straggler-aware gradient aggregation for large-scale distributed deep learning system," *IEEE/ACM Trans. Netw.*, vol. 32, no. 6, pp. 4917–4930, Dec. 2024.
[41] A. Kivity et al., "KVM: The Linux virtual machine monitor," in *Proc. Linux Symp.*, 2007.
[42] Z.-H. Zhang, Z. Yang, Y. Sun, Y.-F. Wu, and Y.-D. Xing, "Lenet-5 convolution neural network with mish activation function and fixed memory step gradient descent method," in *Proc. 16th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process.*, Dec. 2019, pp. 196–199.
[43] B. Koonce, "ResNet 50," in *Convolutional Neural Netw. With Swift for Tensorflow: Image Recognit. Dataset Categorization*, 2021, pp. 63–72.
[44] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, "CIFAR-10: KNN-based ensemble of classifiers," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2016, pp. 1192–1195.
[45] N. Ketkar, "Stochastic gradient descent," in *Deep Learning With Python: A Hands-on Introduction*. Berkeley, CA, USA: Apress, 2017.
[46] N. Jain et al., "Predicting the performance impact of different fat-tree configurations," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2017, pp. 1–13.

Yingpu Nian (Graduate Student Member, IEEE) received the B.S. degree in communication engineering from Hebei University of Technology, Tianjin, China, in 2022, and the M.S. degree in Northeastern University, Shenyang, China, in 2024, where he is currently pursuing the Ph.D. degree. His research interests include the next generation Internet, intelligent routing, and distributed optimization.

Bo Yi (Member, IEEE) is currently a Lecturer of computer science and engineering with Northeastern University, China. He has authored and co-authored more than 20 journal and conference articles on IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE COMMUNICATIONS LETTERS, and *Computer Networks*. His research interests include service computing, routing, virtualization, and cloud computing in SDN, NFV, and DetNet. He is a Reviewer of IEEE COMMUNICATIONS SURVEYS AND TUTORIAL, IEEE COMMUNICATIONS LETTERS, *Computer Networks*, and *Journal of Network and Computer Applications*.

Qiang He received the Ph.D. degree in computer application technology from Northeastern University, Shenyang, China, in 2020. From 2018 to 2019, he was with the School of Computer Science and Technology, Nanyang Technical University, Singapore, as a Visiting Ph.D. Researcher. He has published more than ten journal articles and conference papers. His research interests include social network analytic, machine learning, data mining, and software defined networking.

Xingwei Wang received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively. He is currently a Professor with the College of Computer Science and Engineering, Northeastern University. He has published more than 100 journal articles, books and book chapters, and refereed conference papers. His research interests include cloud computing and future Internet. He has received several best paper awards.

Geyong Min (Member, IEEE) received the B.Sc. degree in computer science from Huazhong University of Science and Technology, China, in 1995, and the Ph.D. degree in computing science from the University of Glasgow, U.K., in 2003. He is currently a Professor of high performance computing and networking with the Department of Computer Science, University of Exeter, U.K. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, and modeling and performance engineering.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University in 1985 and the Ph.D. degree in computer science from the University of Houston in 1990. He is a SUNY Distinguished Professor with the State University of New York and a National Distinguished Professor at Hunan University (China). He has authored or co-authored more than 1230 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the World's top few most influential scientists in parallel and distributed computing, regarding single-year impact (ranked #2) and career-long impact (ranked #3) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars (2023-2025) and is among the top 0.02% out of over 20 million scholars worldwide based on top-cited publications in the last ten years. He is listed in ScholarGPS Highly Ranked Scholars (2022-2025) and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the 2022-2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS Fellow, an AAIA Fellow, an ACIS Fellow, and an AIIA Fellow. He is a Member of the European Academy of Sciences and Arts. He is a Member of Academia Europaea (Academician of the Academy of Europe).

Sajal K. Das (Fellow, IEEE) is a Curator's Distinguished Professor of Computer Science and the Daniel St. Clair Endowed Chair at Missouri University of Science and Technology, USA, where he was the Chair of the Computer Science Department during 2013-2017. He served as a Program Director in the Division of Computer Networks and Systems Division at the National Science Foundation during 2008-2011. His current research interests include wireless and sensor networks, mobile and pervasive computing, UAVs, cyber-physical systems, IoT, machine learning, smart environments (e.g., smart city, smart grid, smart transportation, smart agriculture, and smart health), HPC and edge-cloud computing, cyber security, applied graph theory and game theory. He published more than 700 research articles in high quality IEEE and ACM transactions and refereed conference proceedings. He holds five US patents, coauthored 60 book chapters, and five books on Smart Environments: Technology, Protocols, and Applications; Handbook on Securing Cyber-Physical Critical Infrastructure: Foundations and Challenges; Mobile Agents in Distributed Computing and Networking; Principles of Cyber-Physical Systems; and Federated Learning: Foundations and Applications. His h-index is 105 with more than 46,700 citations, according to Google Scholar. He received 14 Best Paper Awards in prestigious conferences including ACM MobiCom and IEEE PerCom. He serves as the founding Editor-in-Chief of *Pervasive and Mobile Computing journal*, and an Associate Editor of IEEE *Transactions on Dependable and Secure Computing*, IEEE *Transactions on Networking*, IEEE *Transactions on Sustainable Computing*, and *ACM Transactions on Sensor Networks*. He is a Distinguished Alumni of the Indian Institute of Science, Bangalore and a Fellow of the National Academy of Inventors (NAI) and Asia-Pacific Artificial Intelligence Association (AAIA).