Accepted to appear in IEEE/CIC International Conference on Communications in China, 2025 ©IEEE

Maximizing the Throughput of Edge-Based In-Network Aggregation With Routing Optimization

Zuohan Qiao[1], Shouxi Luo[1,2], Ke Li[1], Huanlai Xing[1], Bo Peng[1]
[1]School of Computing and Artificial Intelligence, Southwest Jiaotong University
[2]Tangshan Institute, Southwest Jiaotong University

Abstract*—As is known, in-network aggregation (INA) is a promising solution to alleviate the communication bottlenecks faced by emerging distributed applications like data-parallel distributed machine learning (DML). Recent studies have shown that INA-aware routing optimization is the key to making efficient use of deployed aggregators for acceleration. In this paper, we consider a specific INA scenario, where aggregators are deployed at the edge of the data center network (DCN), like the Smart-NIC of servers and the programmable top-of-rack (ToR) switches. We find that existing INA-aware routing schemes like ATP and GRID are far from optimal for such a scenario, since they do not take the characteristics of edge-based INA into consideration. To address the problem, we propose EINA, a model-based routing optimization scheme for edge-based INA. In the core, EINA establishes a *Mixed-Integer Quadratically Constrained Programming* (MIQCP) mathematical model to precisely encode the requirements, constraints, and objective of edge-based INA. Then, EINA transforms the model into an *Integer Quadratically Constrained Programming* (IQCP) for solving acceleration. Based on the results of the model, EINA generates near-optimal (if not optimal) routing schemes to maximize the total aggregation throughput, using deployed edge aggregators. Performance studies indicate that EINA is able to achieve aggregation throughput up to about $3.02\times$ and $3.45\times$ of those of existing schemes, ATP and GRID, respectively, especially for the case where edge aggregators have limited network bandwidth.

***Index Terms*—Distributed machine learning, in-network aggregation, routing optimization.**

I. Introduction

Nowadays, machine learning (ML) techniques, such as deep neural networks, have been widely applied across various domains and have achieved remarkable success [1], [2]. With the development of ML, the scale of the deep neural network (DNN) model is increasing rapidly, making it impossible to train large-scale models using a single device [3], [4]. For such an issue, *parameter server* (PS) based data-parallel distributed machine learning (DML) design is a promising solution and has been widely used in practice [5]. In such a parallelism paradigm, the training dataset is shared among a group of workers for parallel training. During the training, all workers would periodically push their locally trained gradients to one or multiple PSs for aggregation, and then pull the synchronized results to start the next round of training. Obviously, with the scale of the model and/or the training cluster scaling, the network is prone to becoming the bottleneck of the entire system, and in-network aggregation (INA) has been proven to be a promising solution to alleviate the issue [3], [4], [6].

The key idea of INA is to enable some of the network devices such as programmable switches [4] and Smart-NIC applications [7] to pre-aggregate related gradient packets sent by different workers into a single one when they pass by, thus reducing both the traffic volume entering the network and the load of the PS. Obviously, the key to maximizing the benefits of INA lies in routing optimization, since related flows must go through the same aggregator for INA [3]. Recently, several proposals have been proposed for this purpose [3], [4], [8]. However, they are unaware of the heterogeneity of available network bandwidth among aggregators (e.g., ATP [4] and GRID [8]), or are specialized for specific network scenarios (e.g., ARO [3]), thus are far from addressing all the issues.

In this paper, we enrich the study of INA-aware routing optimization by designing EINA to maximize the aggregation throughput of edge-based INA. We focus on the case where aggregators are deployed at the edge of the data center network (DCN) like the Smart-NIC of servers or the programmable top-of-rack (ToR) switches. Such scenarios are very common in practice. For example, in public clouds, the workers run inside virtual machines or containers; then, the NIC-based network function application (e.g., SoftINA [7]) instances running on the host servers could work as edge aggregators. And in data centers built upon P4-compatible switches, the ToR switches might be configured to support INA, acting as edge aggregators [4]. In these scenarios, edge aggregators are connected by a flat DCN, and due to the impact of the other coexisting applications, different aggregators might have distinct available bandwidth for the DML application. Consequently, existing schemes such as ATP [4] and GRID [8] that ignore this heterogeneity would have performance issues, as Section IV shows. Also, by allowing the traffic to go through multiple aggregators on demand, it is possible to make more efficient usage of deployed edge aggregators. However, existing INA routing optimization schemes like ARO [3] have enforced flows to satisfy the up-down routing principle, and thus are inapplicable to this edge-based INA scenario.

To address the problem, EINA establishes a *Mixed-Integer Quadratically Constrained Programming* (MIQCP) mathemat-

This work was supported in part by NSFSC under Grant 2025ZNSFSC0489, in part by the Hebei Natural Science Foundation under Grant F2025525008, and in part by the Fundamental Research Funds for the Central Universities under Grant 2682024ZTPY050. *(Corresponding author: Shouxi Luo.)*

ical model to precisely encode the requirements, constraints, and objective of edge-based INA. Then, EINA further transforms it to an *Integer Quadratically Constrained Programming* (IQCP) for solving acceleration. Based on the results of the model, EINA can generate a near-optimal (if not optimal) routing scheme to maximize the aggregation throughput of PS-based model synchronization using deployed edge aggregators. Extensive experiments show that EINA can achieve aggregation throughput up to about 3.02× and 3.45× of those of existing schemes, ATP and GRID, respectively, especially for the case where edge aggregators have limited link capacities.

In summary, we make two main contributions in this paper.

- We propose EINA, a math model driven routing optimization scheme that could achieve optimal or near-optimal aggregation throughput for edge-based INA (§III).
- Extensive evaluations verify that EINA can fully use the aggregator ability to maximize the aggregation throughput, outperforming existing schemes significantly (§IV).

In the rest of this paper, Section II first provides an overview of the background and motivation of INA. Then, Section III describes our model and the design of EINA. After that, Section IV reports our evaluation results. Lastly, Section V concludes the paper and discusses future directions.

II. Background and Motivation

In this section, we first overview the concepts of INA (§II-A), then analyze the limitations of existing INA routing optimization schemes (§II-B), and finally explain our motivation for proposing the algorithm EINA (§II-C).

A. Overview of INA

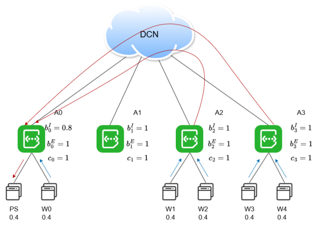
In-network aggregators are widely used in many fields, such as distributed machine learning, big data analysis, and so on [7]. For distributed machine learning, in the PS architecture, a distributed machine learning training task needs to split the model parameters across multiple worker nodes. After the calculation is completed, each worker node needs to push the data to the PS for aggregation operations like summation and weighted average, then the parameter server will distribute the data back to the worker nodes [3], [4]. However, the limit DCN egress capacity for the PS and link capacity may limit the training speed of the distributed machine learning model. In-network technology can offload the aggregation process to the network transmission process, thereby reducing the workload of data transmission. So far, in Parameter Hub [5], the authors design an efficient gradient aggregator, which achieves efficient overlap between gradient processing and communication through fine-grained key block division and core mapping strategy. In NetAgg [9], the authors propose a distributed aggregator based on middlebox. Executing application-specific aggregation functions on the paths of data center networks effectively reduces network flow, alleviates network bottlenecks, and significantly improves the performance and throughput of distributed applications. In SoftINA [7], the authors design a software-based network aggregator with high flexibility and scalability without relying on dedicated hardware.

B. Related Work

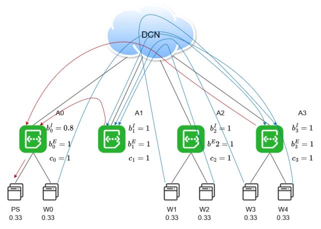
For the routing control of INA traffic, ATP [4] adopts the default ECMP routing scheme without considering the heterogeneity of the available network bandwidth among aggregators. Paring [10] proposes a framework to accelerate distributed training by jointly optimising task routing aggregation in the network, which minimises the cost of Steiner trees and finds spider structures to confirm routes. This framework can significantly reduce communication time and network flow overhead. However, the algorithm does not support multiple aggregations of traffic and relies on heuristic methods, which degrade solution quality. GRID [8] uses a mathematical modeling method and a gradient routing algorithm based on random rounding on the control plane to improve the efficiency of in-network aggregation. However, it permits only a single aggregation, and the randomized rounding method leads to unstable results. InArt [11] proposes using a Lagrange multiplier and a random rounding algorithm in route selection. It adopts randomized rounding, which leaves room for improvement in solution quality. Additionally, data aggregation is allowed only once. ARO [3] achieves the optimal routing selection of in-network aggregation on programmable switches for Clos topology by establishing an acceptable mathematical model, and designs a heuristic algorithm to ensure that the solution of large-scale topology is completed in a limited time. However, ARO's routing path must strictly adhere to the up-down principle. In Camdoop [12], aggregation is placed on the CPU of the server, which uses topology-aware heuristic rules combined with multi-objective optimization and greedy algorithm ideas to build aggregation trees for each to reduce usage, and realizes the aggregation at the intermediate server in the transmission path. This algorithm targets 3D torus topologies. In the work of GOAT [13], the authors propose a stochastic rounding method based on the knapsack problem to decide the aggregation position of the gradient, which primarily focuses on the limited memory of switches, which differs from what we focus in this paper.

C. Motivating Examples

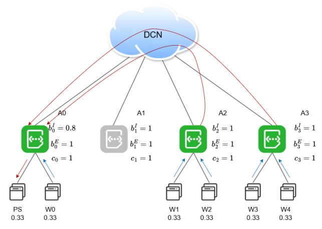
Currently, most aggregator routing algorithms are for fat-tree or spine-leaf network topologies. Most of them aim to deploy the aggregator on the programmable switch. In contrast, the application scenario studied in this paper is aimed at the aggregation routing optimization task of the aggregator deployed on the edge aggregators. First, the main difference between our aggregator and the work node is that they are on the same edge aggregator, so there are many differences in the DCN egress capacity and other aspects. Secondly, because the scenario considered by solutions like ATP [4], GRID [8], and ARO [3], is the switch, it needs to abide by the up-down principle, and the edge aggregators do not need to consider the up-down principle, so the number of routes that can be planned is greater. Thirdly, our solution allows data to be aggregated multiple times and combined with routing optimization to maximize the throughput. So far, research related to direct network topology has been conducted in CamDoop. This



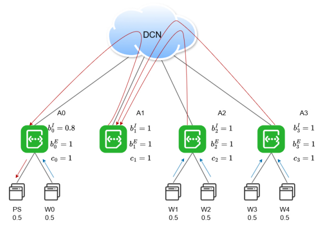
(a) ATP



(b) GRID



(c) Random Routing



(d) EINA

Fig. 1: Consider that there are five workers training a model with one PS and four edge aggregators. Aggregator 0 has 0.8 Gbps available DCN egress capacity, 1 Gbps available DCN ingress capacity, and 1 Gbps aggregation capacity, while all other aggregators have the same ingress, egress, and aggregation capacities of 1 Gbps. Here, blue lines denote data links without aggregation; red lines represent aggregated data links; green-highlighted aggregators indicate active participants in the aggregation process, and grey ones signify non-participating nodes. In short, the ATP scheme achieves a worker node sending throughput of 0.4 Gbps (Figure 1a,); both GRID and random routing (RR) schemes yield 0.333 Gbps (Figures 1b and 1c)); while our EINA delivers superior throughput at 0.5 Gbps (Figure 1d).

algorithm is for the 3D torus topology, which differs from ours.

We give a concrete example to show our motivation in Figure 1, which shows the different routing scenarios on the same configuration between GRID [8], ATP [4], random routing (RR), and our proposed EINA. The GRID does not consider the limit of ingress capacity and egress capacity, and it allows the data generated by the worker node to be aggregated at other edge aggregators. Compared to other algorithms, it consumes more ingress and egress capacity, and the random rounding method has a high probability of discarding the optimal solution. At the same time, the ATP scheme is not optimal because its data can only be aggregated on the nearest aggregator once. In comparison, EINA takes full advantage of the aggregator's aggregator capability and produces a higher throughput solution.

III. EINA

In this section, we first introduce the system model (§III-A), then give a formal definition of the optimization problem (§III-B), and finally introduce the solver that we use to solve the model for routing optimization (§III-C).

A. System Model

As shown in Figure 1d, we consider that a data-parallel distributed training job involving a group of workers W and a set of PSs P, is hosted in a cluster equipped with a group of edge-based aggregators A.We define A_W as the set of worker-assigned aggregators and $A_E = A \setminus A_W$ as the remaining aggregators. For load balancing, the trained model is sharded among all these $|P|$ PSs. To iterate a round of training, each worker splits its obtained gradient vector into $|P|$ messages and push them to PSs concurrently for gradient aggregation and synchronization.

For each PS and worker k, it is assigned to an aggregator denoted by $\pi[k]$, which has the INA capacity of $c_{\pi[k]}$. We use a_i to denote the number of workers assigned with aggregator i, Regarding implementation, these edge-based aggregators could be i) NIC-based network function application (SoftINA [7]) instances running on the hosting servers, where workers run in virtual machines or containers, or ii) P4-based ToR switches like ATP [4] that support INA, where workers run on servers under these switches. We assume that each worker has a high-performance and low-latency connection to its assigned aggregator, which would not be a communication bottleneck. Then, all aggregators are networked via a non-blocking DCN fabric in which congestion generally occurs at the ingress or egress of the network [14], [15]. For such a network, we assume that the aggregator i currently has the available capacity of b_i^I and b_i^E for links to the ingress and egress of the DCN fabric, respectively.

Following the design of ARO [3], we let all workers send data to PS p at the same throughput r_p, which is a variable, to unleash the power of INA. To capture their routing states among edge-based aggregators, given PS p, aggregators i and j, we further define a binary variable $x_{p,i,j}$ to denote whether aggregator i would send a flow with the final destination of PS p, to another aggregator j.

B. Problem Formulation

Now, we explain the constraints and objectives of our optimization problem in detail.

1) Flow throughput constraints: Since a task's data sending throughput is limited by the worker's network capacity, and data transmission would become ineffective when the throughput falls below a minimum threshold, EINA allows users to limit the allowed range of the data sending throughput via pre-defined constants r_p^L and r_p^U, as follows.

$$r_p^L \le r_p \le r_p^U, \quad \forall p \in P \tag{1}$$

2) Routing requirements: Each aggregator i assigned to training workers must send its aggregated flow aimed at PS p to another aggregator $j \in A \setminus \{i\}$.

$$\sum_{\forall j \in A \setminus \{i\}} x_{p,i,j} = 1, \quad \forall p \in P, \forall i \in A_W \tag{2}$$

Obviously, the routes for traffic sent by all workers to a PS p would form a tree rooted at aggregator $\pi[p]$. For each aggregator i, we use an integer variable $d_{p,j}$ to denote its depth (in terms of hops) to the root aggregator $\pi[p]$. Then, for each PS p, its root aggregator $\pi[p]$'s depth has

$$d_{p,\pi[p]} = 0, \quad \forall p \in P \tag{3}$$

To limit the length of the path for INA traffic, EINA allows users set a parameter of d^* to limit the maximum number of aggregators that INA traffic could encounter before reaching the PS. Recall that there are $|A|$ aggregators in total and let $\hat{d} = \min(d^*, |A|) - 1$. Then, for other aggregators, we have

$$d_{p,i} \in \{1, \cdots, \hat{d}\}, \quad \forall p \in P, \forall i \in A \setminus \{\pi[p]\} \tag{4}$$

Moreover, if aggregator j sends data to aggregator i, j's depth should be i's depth plus one. By introducing a large constant M, we have the following pair of constraints.

$$d_{p,i} \geq d_{p,j}+1-M(1-x_{p,i,j}), \quad \forall p \in P, \forall i \in A, \forall j \in A \tag{5}$$

$$d_{p,i} \leq d_{p,j}+1+M(1-x_{p,i,j}), \quad \forall p \in P, \forall i \in A, \forall j \in A \tag{6}$$

For an aggregator having not been assigned to any worker or PS, on receiving data forwarded by another aggregator, it must forward the aggregated traffic out again; otherwise, no forwarding is needed for it.

$$\sum_{j \in A \setminus \{i\}} x_{p,i,j} \leq 1, \forall p \in P, \forall i \in A_E \tag{7}$$

$$\sum_{\forall k \in A} x_{p,j,k} \leq M \sum_{\forall i \in A} x_{p,i,j}, \forall j \in A_E \tag{8}$$

$$d_{p,i} \geq \sum_{\forall j \in A} x_{p,i,j}, \forall i \in A_E \tag{9}$$

3) Limited aggregation capacity and network capacity: Each edge-based aggregator i has a limited aggregation capacity c_i. So, the sum of each flow throughput r_p multiplied by the number of flows before aggregation on this aggregator should exceed this limitation.

$$\sum_{\forall p \in P} \sum_{\forall i \in A \setminus \{j\}} (x_{p,i,j} + a_j) r_p \leq c_j, \quad \forall j \in A \tag{10}$$

Regarding the limitation of capacity, the total flow throughput output from an aggregator i should not exceed b_i^I, i.e., the link capacity of the corresponding ingress of the DCN.

$$\sum_{\forall p \in P} r_p \leq b_i^I, \quad \forall i \in A \setminus \{\pi[p] : p \in P\} \tag{11}$$

The sum of all flows routed to an aggregator from other aggregators should not exceed b_i^E, the link capacity of the corresponding egress of the DCN.

$$\sum_{\forall p \in P} \sum_{\forall i \in A \setminus \{j\}} x_{p,i,j} r_p \leq b_j^E, \quad \forall i \in A \tag{12}$$

4) Objective: Finally, the optimization objective of EINA is to maximize the total aggregation throughput via edge-based aggregator routing scheduling, i.e.,

$$\text{Maximize} \sum_{\forall p \in P} r_p \tag{13}$$

C. Solver Designs

Using expression (13) as the optimization objective while subject to constraints (1)-(12), we have formulated the routing optimization problem as a MIQCP for EINA. By solving it, EINA obtains a routing scheme $\{x_{p,i,j} : \forall(p,i,j)\}$ along with a bandwidth allocation scheme $\{r_p : \forall p \in P\}$ to maximize the aggregation throughput with the acceleration of edge-based in-network aggregators. Following the design of MTREE [6], the PS p in EINA would obtain a $\frac{r_p}{\sum_{\forall p \in P} r_p}$ proportion of the total model for sharding.

In its core, EINA can solve the involved MIQCP by using off-the-shelf commercial available solvers like Gurobi [16]. Motivated by the work of ARO [3], we find that for large-scale scenarios, by enforcing the throughput r_p $(p \in P)$ to be integers, we can transform the MIQCP into an IQCP. Although the problem is NP-hard, as the results in Section IV shows, the solving of Gurobi can be accelerated remarkably. For huge-scale routing optimization, the design of an efficient heuristic algorithm is still needed, which we leave as future work.

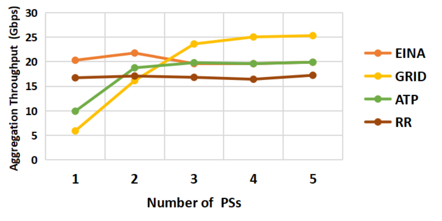
IV. Performance Evaluation

We evaluate the performance of EINA in various scenarios. Results indicate that EINA can achieve performance gains up to 3.02× and 3.45× over the baseline algorithms of ATP [4] and GRID [8], respectively. Moreover, compared to the MIQCP algorithm, the IQCP algorithm achieves up to 15.57× speedup in solution time while up to 8.11% performance reduction. The paper is structured as follows: Section IV-A presents the evaluation framework, and Section IV-B demonstrates the experimental outcomes.

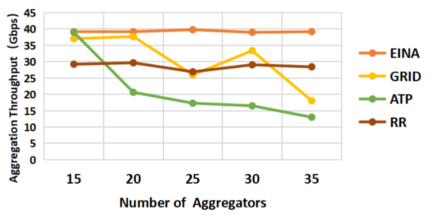
A. Methodology

We evaluate the performance of EINA by mainly use algorithms GRID [8] RR, and ATP [4], as baselines. Our experiments focus on the topology mentioned in Figure 1d. For the ATP algorithm, we primarily apply the idea of random allocation and early aggregation, setting the routing relationship between servers to be randomly generated. In contrast, the GRID algorithm applies the concept of its relaxation to solve random intake and uses the model of the original paper to solve. And the final solution is processed with constraint satisfaction to make the solution feasible. For both EINA and GRID, their involved models are solved by the commercial Gurobi solver [16]. By default, we mainly consider the condition in which there are 2 PSs, with 12 edge aggregators, and each aggregator has 5∼10 workers. All edge aggregators, including PS's ingress, egress, and aggregation capacity, range from 200 Gbps to 300 Gbps.

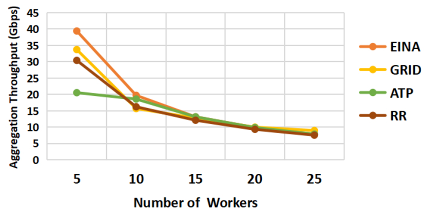
Regarding the metrics, we mainly consider $\sum_{\forall pinP} r_p$, the total aggregation throughput that workers could achieve when



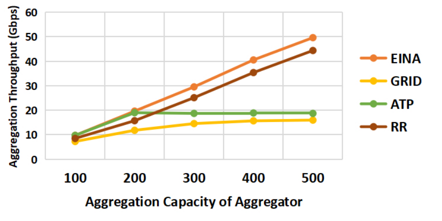
(a) Impacts of number of PSs



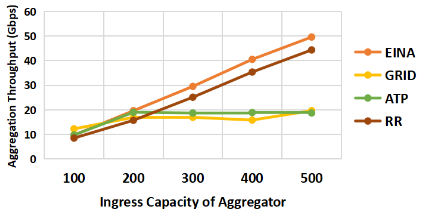
(b) Impacts of number of aggregators



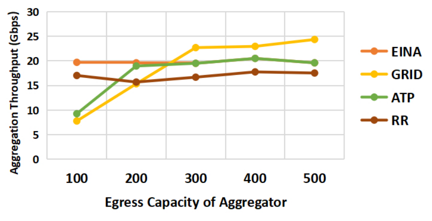
(c) Impacts of number of workers



(d) Impacts of aggregator capacity

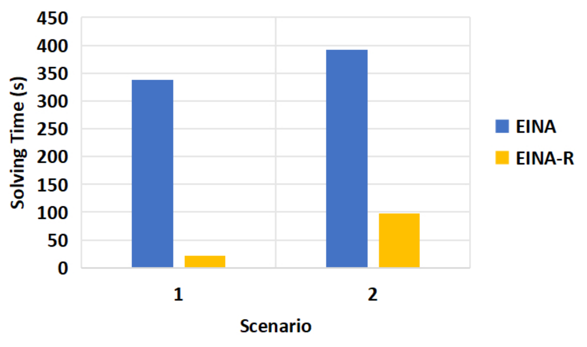


(e) Impacts of ingress capacity

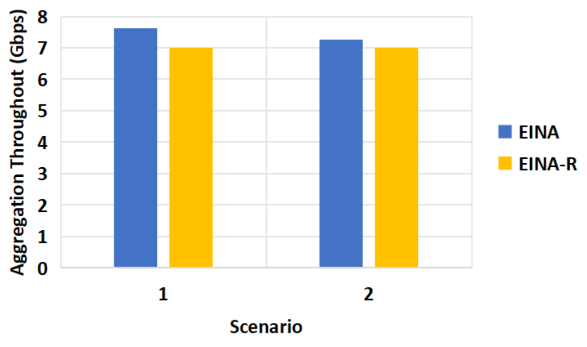


(f) Impacts of egress capacity

Fig. 2: Detailed experimental results.



(a) Solving time



(b) Aggregation throughput

Fig. 3: Experimental results demonstrate that transforming MIQCP into IQCP can significantly improve model-solving efficiency while maintaining virtually unchanged total aggregation throughput.

transmitting the gradients to all PSs for aggregation. We also investigate the time cost of solving the involved models using Gurobi. All experiments are performed on an Ubuntu server involving an i9-13900k CPU and 125G DDR5 memory.

B. Results

We first evaluate the performance of different algorithms under different numbers of parameter servers, as shown in Figure 2a. As the number of PSs increases, the algorithms become more complex. When there is only 1 PS, the total throughput obtained by EINA, GRID, ATP, and RR are 20.3 Gbps, 5.88 Gbps, 9.9 Gbps, and 16.72 Gbps. With the number of PSs increases to 5, the total throughput obtained by EINA, GRID, ATP, and RR are 19.90 Gbps, 25.32 Gbps, 19.90 Gbps, and 17.23 Gbps, respectively. As the number of PS increases, GRID and ATP show a trend of first increasing and then stabilizing, while RR and EINA algorithms remain stable. The throughput of GRID keeps growing and exceeds EINA because at this point, each aggregator's aggregation capacity is limited, and EINA has fully utilized the aggregation capacity of each aggregator. At the same time, GRID allows data to be sent directly to PS nodes without aggregation. Therefore, a higher total throughput can be achieved by GRID when the aggregation capacity becomes the limiting factor, but the ingress and egress capacities are sufficient.

We further study the impact of different numbers of edge aggregators as shown in Figure 2b. When the number of edge aggregators is 15, the total throughput obtained by EINA, GRID, ATP, and RR are 39.4 Gbps, 33.72 Gbps, 20.53 Gbps, and 30.41 Gbps, respectively. As the number of edge aggregators increases to 20 and 25, the total aggregation throughput obtained by EINA, GRID, ATP, and RR are 7.84 Gbps, 8.95 Gbps, 7.84 Gbps, and 8.53 Gbps. As the number of aggregators increases, ATP and GRID show a trend of first decreasing and then stabilizing. At the same time, EINA and RR show a stable trend. This is because EINA and RR allow data to be aggregated multiple times, so they can fully use the aggregators' aggregation ability to accommodate more aggregators. However, GRID, and ATP only allow INA once, so the aggregation capacity and the egress point of the DCN for aggregator connected to the PS will quickly become the bottleneck, thus decreasing the throughput.

Under the same conditions, we evaluate the impact of different numbers of workers as shown in Figure 2b. When the number of edge aggregators ranges from 10 to 15, the total throughput obtained by EINA, GRID, ATP, and RR is 39.2 Gbps, 37.04 Gbps, 39 Gbps, and 29.23 Gbps, respectively. As the number of edge aggregators increases to range from 30 to 35, the total throughput obtained by EINA, GRID, ATP, and RR are 39.4 Gbps, 18.04 Gbps, 13 Gbps, and 28.42 Gbps. In general, as the number of worker nodes increases, all algorithms show a downward trend, and finally differ by five levels. This is because when there are fewer worker nodes, GRID produces larger throughput than EINA which allows it not to aggregate on its nearest aggregator. The aggregators connected by PS also have insufficient aggregation or DCN egress capacity, so the throughputs of the four algorithms are almost the same.

We also test the situation where the aggregation capacity, ingress and egress capacity in different edge aggregators differ. The results are shown in Figures 2d, 2e, and 2f. In summary, as aggregation capacity increases, the throughput of RR (Random Routing) and EINA continues to improve because multiple rounds of aggregation are permitted, preventing the aggregators connected to the PS from becoming bottlenecks. The trend and cause of throughput variation when the DCN ingress bandwidth increases are consistent with the different aggregation capacity experiments. However, as DCN egress bandwidth gradually increases, GRID and ATP exhibit an initial rise followed by stabilization because the primary bottleneck shifts from DCN egress bandwidth to the aggregation capacity of the PS-connected aggregators.

Our evaluation also incorporated the relaxation algorithm across two experimental scenarios: one scenario with 2 PSs and 50 edge aggregators, and the other scenario with 4 PSs and 50 edge aggregators. Each scenario's edge aggregator is connected to 5-10 worker nodes. The aggregation capacity, ingress, and egress capacities ranged from 200 to 400 Gbps across all scenarios, and there are 10 aggregators without workers in each scenario. The experimental results in Figure 3 reveal that while the relaxation algorithm showed a marginal performance degradation of up to 8.11% reduction compared to the original algorithm, it achieves up to 15.57× in solution time, demonstrating remarkable computational efficiency improvements. This significant time reduction is consistently observed across all evaluation scenarios.

V. Conclusion and Future Work

This paper proposes a routing optimization scheme EINA for edge-based in-network aggregation where aggregators are deployed on edge nodes like the smart network cards of servers and ToR switches, for distributed applications like PS-based data-parallel distributed machine learning. By systematically modelling the routing problem as a mathematical optimization problem, EINA can maximize the total aggregation throughput of workers. Performance evaluations demonstrate that EINA delivers significantly higher aggregation throughput, outperforming conventional ATP and GRID schemes by approximately 3.02× and 3.45×, respectively. However, when the aggregation capacity of the aggregator is significantly lower than its bandwidth capacity, our algorithm does not perform as well as GRID, mainly due to the restriction that all worker nodes need to be aggregated on the corresponding aggregator. Therefore, in future work, we will further optimize the EINA algorithm by loosening this restriction and so on, so that it can obtain a better and efficient routing scheme.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of CVPR*, 2016, pp. 770–778.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL*, vol. 1. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[3] S. Luo, X. Yu, K. Li, and H. Xing, "Releasing the power of in-network aggregation with aggregator-aware routing optimization," *IEEE/ACM Transactions on Networking*, vol. 32, no. 5, pp. 4488–4502, 2024.

[4] C. Lao, Y. Le, K. Mahajan, Y. Chen, W. Wu, A. Akella, and M. Swift, "ATP: In-network aggregation for multi-tenant learning," in *Proceedings of NSDI*, Apr. 2021, pp. 741–761.

[5] L. Luo, J. Nelson, L. Ceze, A. Phanishayee, and A. Krishnamurthy, "Parameter hub: a rack-scale parameter server for distributed deep neural network training," in *Proceedings of SoCC*. ACM, 2018, pp. 41–54.

[6] S. Luo, R. Wang, and H. Xing, "Efficient inter-datacenter allreduce with multiple trees," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 5, pp. 4793–4806, 2024.

[7] X. Yu and S. Luo, "Softina: A softwarized in-network aggregator for distributed applications," in *Proceedings of 2024 10th International Conference on Computer and Communications (ICCC)*, 2024, pp. 351–355.

[8] J. Fang, G. Zhao, H. Xu, C. Wu, and Z. Yu, "Grid: Gradient routing with in-network aggregation for distributed training," *IEEE/ACM Transactions on Networking*, vol. 31, no. 5, pp. 2267–2280, 2023.

[9] L. Mai, L. Rupprecht, A. Alim, P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf, "NetAgg: Using middleboxes for application-specific on-path aggregation in data centres," in *Proceedings of CoNEXT*. ACM, 2014, pp. 249–262.

[10] 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 Transactions on Networking*, vol. 32, no. 5, pp. 4317–4332, 2024.

[11] J. Liu, Y. Zhai, G. Zhao, H. Xu, J. Fang, Z. Zeng, and Y. Zhu, "InArt: In-network aggregation with route selection for accelerating distributed training," in *Proceedings of the ACM Web Conference 2024*. ACM, 2024, pp. 2879–2889.

[12] P. Costa, A. Donnelly, A. Rowstron, and G. O'Shea, "Camdoop: Exploiting in-network aggregation for big data applications," in *Proceedings of NSDI*. USENIX Association, 2012, pp. 29–42.

[13] J. Fang, H. Xu, G. Zhao, Z. Yu, B. Shen, and L. Xie, "Accelerating distributed training with collaborative in-network aggregation," *IEEE/ACM Transactions on Networking*, vol. 32, no. 4, pp. 3437–3452, 2024.

[14] S. Luo, H. Yu, K. Li, and H. Xing, "Efficient file dissemination in data center networks with priority-based adaptive multicast," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1161–1175, 2020.

[15] S. Luo, K. Li, H. Xing, and P. Fan, "Efficient and flexible component placement for serverless computing," *IEEE Systems Journal*, vol. 18, no. 2, pp. 1104–1114, 2024.

[16] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com